Data Self-Expansion and DoppelGANger-Based Time-Series Modeling for Realistic Steam Data Generation

1st Xinying Cai
Chu Kochen Honors College
Zhejiang University
Hangzhou, China
xinying.cai@zju.edu.cn

2nd Zheng Luo
Polytechnic Institute
Zhejiang University
Hangzhou, China
Zheng_Luo@zju.edu.cn

3rd Xueru Lin
College of Energy Engineering
Zhejiang University
Hangzhou, China
linxueru@zju.edu.cn

4th Ning Zhang
Polytechnic Institute
Zhejiang University
Hangzhou, China
zhang_n@zju.edu.cn

5th Yihui Mao
College of Energy Engineering
Zhejiang University
Hangzhou, China
Y.H.Mao@zju.edu.cn

6th Xiaojie Lin
Changzhou Industrial Technology
Research Institute
Zhejiang University
Hangzhou, China
xiaojie.lin@zju.edu.cn

7th Wei Zhong
Polytechnic Institute
Zhejiang University
Hangzhou, China
zhongw@zju.edu.cn

***Abstract*—Steam heating is an essential component of numerous industrial production processes that consume energy extensively. In particular, steam data are of significant commercial interest, and the privacy of the heat suppliers and steam users for whom this data is confidential is of paramount importance. However, the complexity and intricacy of steam data make it challenging to generate realistic samples using conventional data generation algorithms. This paper proposes a self-expansion data process method that multiplies samples with a coarser granularity. We then apply the time-series generation model, DoppelGANger, to steam data. Post-processing for the generated data is implemented to eliminate the effects of a certain level of noise. To validate the proposed method and model, we conduct experiments using field data obtained from a thermopower company. The results indicate that DoppelGANger can capture the steam operating characteristics and generate realistic samples reflecting steam data patterns. The difference between the statistic metrics of the true and generated samples is around 2.87%. The mean DTW distance of features' principal components is around 16.32. Overall, this study offers a promising workflow for data generation across many steam system scenarios. This approach is valuable for energy management, privacy protection, and data sharing in industrial domains.**

***Index Terms*—industrial steam data, self-expansion, GAN, DoppelGANger, data generation, post-processing**

I. Introduction

Industrial parks bring together many heat users, with huge heat consumption and more centralized and large-scale steam demand, making it easier to form economies of scale and environmental benefits, and various policies have been introduced across the country to promote industrial parks to eliminate decentralized coal-fired boilers and implement large-scale centralized heat supply [1]. It is estimated that around 70 percent of production energy consumption in industrial parks relies on steam heating [2]. The industrial steam heating system (ISHS) contains a large number of pipes and heat exchange equipment [3].

In order to enhance the coordination and management of the production and consumption of industrial steam, various research has been conducted, such as dynamic modeling of storage characteristics based on industrial steam [3]. However, collecting and sharing the required data can be a challenge, due to commercial interests and user privacy concerns [4]. As a result, the publicly available industrial steam data is limited in both scale and duration, which creates a research gap in this area. Moreover, traditional data generation models such as autoregressive models and Markov models are not well-suited for generating realistic industrial steam data [5].

In 2014, Goodfellow et al. proposed a promising data generation framework called generative adversarial networks (GANs) [6]. GANs are powerful deep generative models that can implicitly capture any differentiable probability distribution and generate samples accordingly.

Assuming some prior distribution $z \sim P_z(z)$, the goal is to learn a generator function G, such that $G(z) \sim P_{data}(x)$. To achieve this goal, a discriminator D and a generator G are introduced to play the following two-player minimax game with a loss function:

$$L(G, D) : \ min_G \, max_D \, V(G, D) \tag{1}$$

Here, $V(G, D) = E_x \sim p_{data}(x)[logD(x)] + E_z \sim p_z(z)[log(1 - D(G(z)))]$, P_{data}, P_z represent the distribution

National Key R&D Program of China (Grant No. 2019YFE0126000).
National Natural Science Foundation of China (Grant No. 51806190).

of real data and noise data.

To tailor GANs towards specific data types, several adaptations, and variants have been proposed, such as Wasserstein GAN (WGAN) [7], conditional GAN (cGAN) [8], and TimeGAN [9] which is aimed at generating accurate time series data.

However, generating realistic industrial steam data using traditional GAN models poses significant challenges. Industrial steam is strongly correlated with time, lacks classification and labeling, possesses a complex, stochastic probability distribution that is related to the unpredictable demands and behaviors of heat users and lacks correlation with external factors such as weather and temperature. To address these challenges, we propose a variant of the GAN-based framework, DoppelGANger (DG), which is specifically optimized for generating synthetic industrial steam time series data. Through three evaluation methods, we show that DG model performs well in generating realistic synthetic steam time series data. Our contribution is the introduction of a novel approach for generating synthetic industrial steam time series data using DG, which can be used to enhance operational control and privacy protection.

II. Methodology

A. Model Structure

DoppelGANger (DG) is a state-of-the-art generative adversarial network (GAN) that generates time series datasets with metadata. It has shown superior performance compared to baseline models across diverse real-world datasets and use cases [10]. In this study, we adapted DG to generate steam data due to its excellent performance and adaptability.

The modified DG used in this paper consists of a generator part and a discriminator part.

1) Generator Part: The generator has three parts as follows:

- **Label generator**: This part generates attributes that do not vary over time, also known as metadata [10]. As the label is not temporal data, a simple multilayer perceptron (MLP) model is used to generate it. We set up this dedicated generator to generate the label separately because generating samples containing both attributes and measurements in one generator is difficult and often results in suboptimal performance.
- **Min/Max generator**: This part supports per-example scaling of continuous variables to handle data with a large dynamic range. It can also alleviate the problem of mode collapse. Similar to the label generator, this part also uses an MLP model.
- **Measurement generator**: This part contains a long short-term memory (LSTM) network to produce sequence data, but with a batch setup where each LSTM cell outputs multiple time points to improve temporal correlations. To preserve the hidden relationships between the metadata and the measurements, the generated labels $(L_1, \cdots, L_m)$ are added as an input to the cell at every step.

Combined, these three generators comprise the probability multiplication rule: $P(L_i, R_i) = P(L_i) \cdot P(R_i|L_i)$.

2) Discriminator Part: On the other hand, the discriminator part of DG has three discriminators. One is the auxiliary discriminator, which only judges labels and ranges, and the other discriminator judges complete data, including labels, ranges, and measurements. The loss functions of these two discriminators are added with weights and input to a probability discriminator which outputs the probability of being real or synthetic. Discriminators assess the fidelity of the generated samples and provide feedback to the generator to improve the quality of the generated data.

Given the modified DG model structure we introduced, we proceed to the next section to describe the dataset preprocessing and experimental setup.

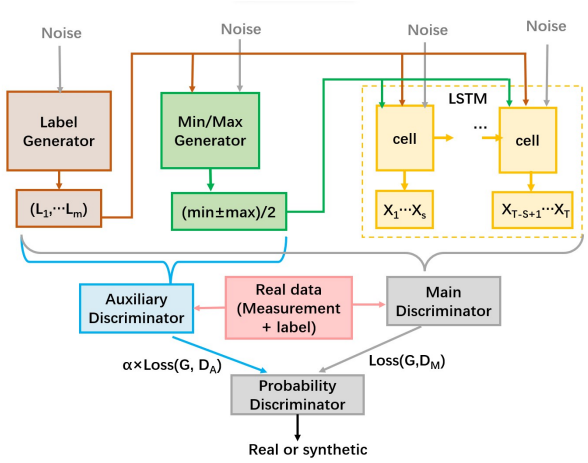

Fig. 1. DG Model Structure

3) Loss Function: DG uses Wasserstein loss with gradient penalty to reduce mode collapse and improve training. The Wasserstein loss is shown in Eq. 2.In (2), $f_w(x)$ is a discriminator network with w.

$$\min_G \max_D L(G, D) = E_{X \sim P_r}[f_w(x)] - E_{X \sim P_g}[f_w(x)] \quad (2)$$

The loss function of DG is the weighted sum of the Wasserstein loss of the auxiliary discriminator and main discriminator, i.e.,

$$\min_G, \max_{D_1, D_2}; L_D = L_1(G, D_1) + \alpha L_2(G, D_2) \quad (3)$$

where α is the weighting term. The discriminator loss L_D is the difference between the expected value of the discriminator output for generated data P_g and the expected value of the discriminator output for real data P_r. The generator loss L_G is the negative expected value of the discriminator output for

generated data P_g. Together, minimizing L_D and maximizing L_G improve the quality of the generated data.

Given the loss function of our modified DG model, we proceed to the next section to describe the dataset preprocessing and experimental setup.

B. Data Setup

Data preparation is crucial to generating high-quality synthetic data. Our study found that the raw data available for training were of poor quality and insufficient scale, which could impede the learning of the model and the generation of successful samples. Thus, we employed a data preparation pipeline consisting of three stages: preprocessing, self-expansion, and clustering and labeling.

1) Preprocessing: Raw data from energy systems can be of poor quality, with issues such as duplicate values, outliers, missing values, redundant fields, uneven time intervals, and irregular shapes. To address these issues, we implement four preprocessing steps as shown in Fig. 2.

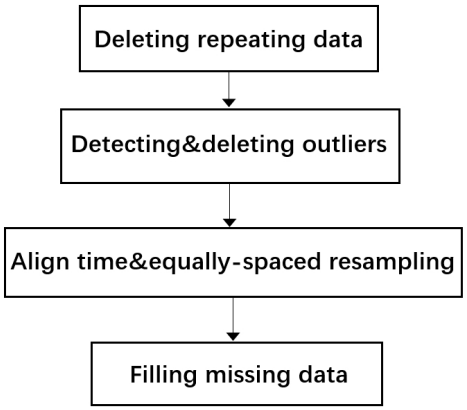

Fig. 2. Steps of Preprocessing

- **Deleting meaningless repeating samples**: To remove duplicated samples that do not provide additional information.
- **Outlier detection and removal**: To remove samples with extreme values that are not representative of the underlying data distribution. We use methods such as the three-sigma rule, box-plot, and k-nearest neighbors (K-NN) to detect and remove outliers.
- **Time alignment**: If there is more than one time column in raw data, the time columns are merged to ensure alignment. To obtain equally-spaced time intervals, we use methods such as the resample function in the pandas library, a popular library in Python for data manipulation and analysis.
- **Missing value imputation**: To deal with missing samples, we apply appropriate imputation methods based on the missing ratio and the specific situation. These methods include mean or median imputation, stochastic regression imputation, carry forward/carry backward, interpolation/extrapolation, among others.

Through these preprocessing steps, we can ensure that the data is in a suitable format for subsequent stages in data preparation. We will now discuss the self-expansion, clustering and labeling stages.

2) Self-Expansion: Before this step, it is important to determine what a standard sample should look like based on the specific task and goals. In the steam dataset, we observed a clear daily pattern and defined the minimum unit generated in our study as one day's steam data.

Afterward, we found that the available steam data was insufficient for training a model. For instance, in a dataset covering one year, there are only around 365 samples, which is not sufficient for the model to learn from. Furthermore, we found that steam has relatively high inertia, while data acquisition instruments tend to have a high sampling rate. Therefore, we need more one-day samples that do not require such high time granularity.

To solve this problem in a cost-effective way, we propose a self-expansion method. Similar to how the Monkey King's hair can transform into clones, our method expands one-day samples by sacrificing some granularity [11]while retaining important information. The degree of granularity reduction is determined by calculating the volatility of each time granularity for each feature and then drawing a function of each feature's mean volatility and time granularity. We can then use empirical knowledge and the figure to determine an appropriate time granularity level.

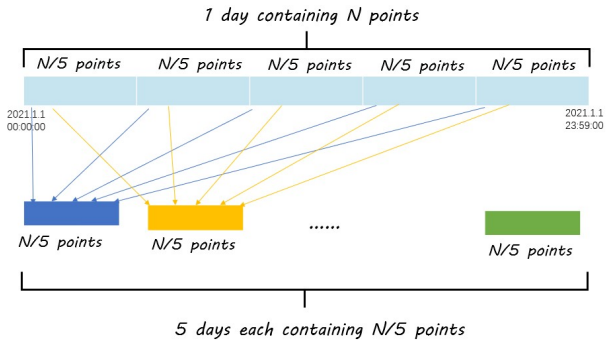

Fig. 3. An example of self-expansion

For instance, if the preprocessed dataset has a time interval represented as A minutes with a period of one year, then the number of points captured in one day would be $N = 24 \times 60/A$. Suppose the optimal granularity to be 5A minutes. Figure 3 depicts the data expansion process. Consequently, we obtain a dataset of dimension $(5 \times 365, (N/5, f))$, as shown in the figure, where each sample has $N/5$ time points and f features. The multiple the optimal granularity to the smallest time unit is the multiple the number of samples can be expanded.

3) Clustering and Labeling: Moreover, we utilize unsupervised learning to tag our samples and provide

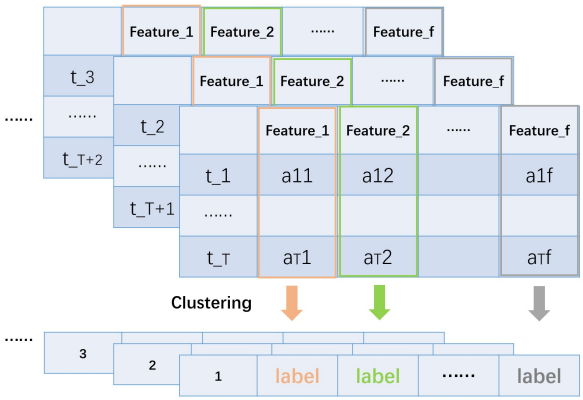

Fig. 4. Clustering of the same features for each sample

valuable insights into their operating modes. Specifically, we cluster the samples' features using k-means clustering and obtain a labeled dataset of dimension (number of samples, (1, number of features)), as shown in Figure 4. This labelled dataset is stored separately and does not require merging with the measurement data.

III. Experiment

A. Data Generation

Our dataset for this experiment was obtained from a thermopower company situated in Suzhou, China, which has a primary heat supply pipeline network spanning 35 km and an annual heat supply of approximately 2.1 million tons. The real data reflect the operating status of one user, causing the need for realistic and high-fidelity datasets with strong privacy protection measures.

We collected the data between July 1, 2021, and August 31, 2022, for a total of 414 days. The data points were taken at regular intervals of one minute, with some time intervals being irregular. The raw dataset comprised steam flow rates (Q), steam pressure (P), steam temperature (T), and corresponding recording times. Notably, the dataset did not contain any labels or attributes. Since Q, P, and T are essential parameters for pricing and customer requirements, we treated them as the key targets for data generation in our experiment. Moreover, for superheated steam, the combination of steam pressure and temperature suffices to define its state, while Q represents a critical process parameter. Other information such as superheat degree and enthalpy can be easily calculated based on these three parameters.

To prepare the data for analysis, we used the K-NN method to identify outliers, which were replaced with adjacent values. Missing data were handled using Hermit interpolation. As shown in Figure 5, data volatility was calculated on three normalized features - steam flow, steam pressure, and steam temperature. The results of the volatility analysis indicated that Q has the highest volatility, followed by P and T, respectively. Considering the average volatility and granularity, we selected a volatility of 1% as optimal and correspondingly chose a time granularity of 10 minutes.

As a result, we expanded the number of samples by ten times, generating a dataset of 4140 samples, which can be used for further analysis.

The experiment was conducted on a single NVIDIA GeForce RTX 3060 Ti GPU with 8GB memory and an 11th Gen Intel(R) Core(TM) i7-11700K @ 3.60GHz CPU with 16GB RAM. The training time was around 5 hours.

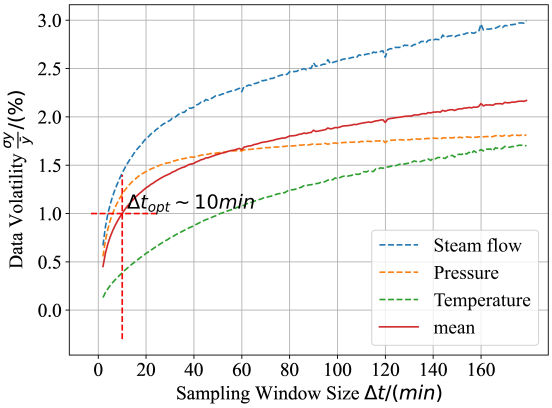

Fig. 5. Optimal Granularity

B. Data Post-processing

One feature in the result is noted to be affected by noise, specifically temperature, which exhibits a small fluctuation range with strong stochasticity. Similar observations have been documented in previous studies [13]. Excessive noise can have several causes, and adjusting model parameters is a common but tough solution. In light of this, we have employed post-processing techniques, such as exponential smoothing [14], to address the noise issue in the affected feature. As demonstrated in Fig. 6, the model has already captured the distribution of real-world data, but noise-reduction through post-processing results in an improved visual effect of synthetic data. Data post-processing provides a novel approach to enhance the effectiveness of models under the constraint of limited training time.

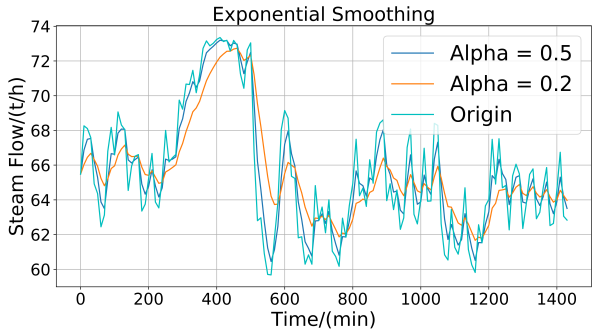

Fig. 6. Exponential Smoothing

TABLE I
STATISTIC METRICS

Statistic Metrics	Real Q	Real P	Real T	Synthetic Q	Synthetic P	Synthetic T
mean	69.38	3.52	388.48	65.55	3.57	378.98
std	4.60	0.0045	3.42	3.58	0.0062	4.82
max	83.01	3.53	395.37	73.35	3.59	390.02
min	60.49	3.51	382.64	59.67	3.56	370.43

C. Results and Discussions

To present our results, we generated a random sample which illuminates the performance as shown in Fig.7. The synthetic data, along with the real data, fall in almost the same range, appear to have similar trends, but with a certain degree of bias.

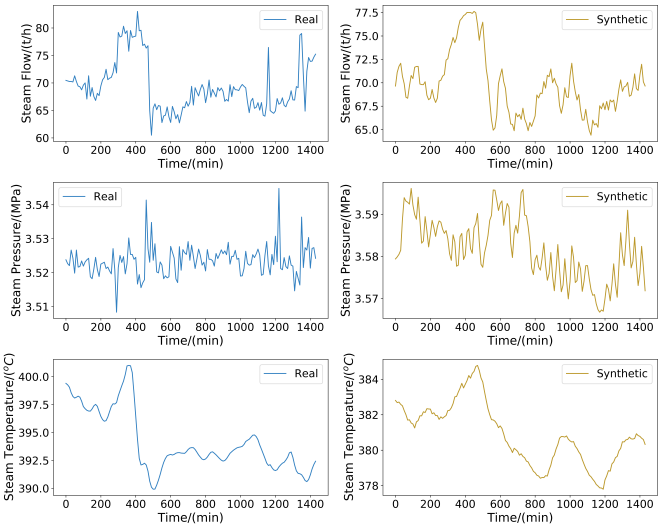

Fig. 7. Steam Synthetic Data

We further evaluated our results through statistical methods, empirical mode decomposition, and dimensionality reduction techniques to analyze the quality of the generated output on the single sample distribution and the overall sample distribution. The results of this analysis are summarized below.

- To begin, we randomly selected a label mode, generated ten synthetic samples under this mode and calculated several statistical metrics, as presented in Table I. The quantitative indicators in the table clearly demonstrate the synthetic data mimics the real-world data's range and fluctuation conditions.The average difference between the generated samples and the true values is within 3%, which meets the accuracy requirements for the generated samples.
- To evaluate the changing trend of each intrinsic mode function (IMF), we decomposed the real and generated data based on empirical mode decomposition (EMD) [15], as shown in Fig. 8. The figure shows that for each mode, the generated data and the original data have the same trend with respect to time which means the underlying patterns, trends, and information contained in the generated data is similar to the original data.

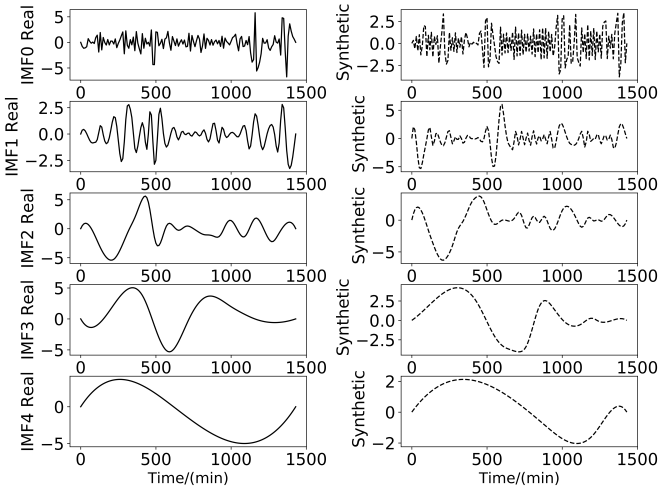

Fig. 8. the IMFS of Real Data and Synthetic Data after EMD

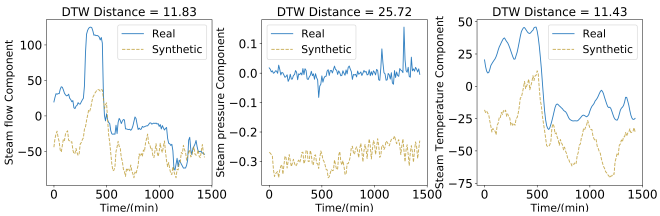

Fig. 9. PCA Transform of Synthetic Data

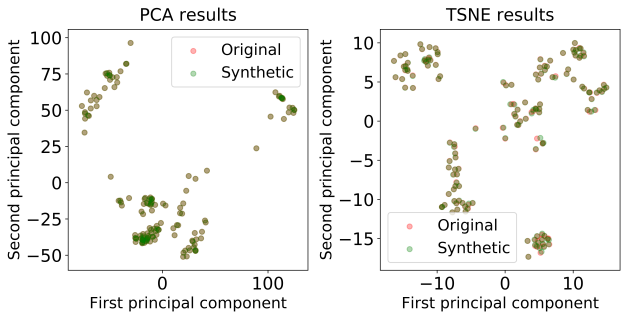

Fig. 10. Synthetic and Real Data Diversity and Distributions

- Ultimately we obtain the common features of the data set by means of data dimensionality reduction and then measure the similarity of the overall distribution of the data by comparing the similarity between the data after dimensionality reduction. The first method used is principal component analysis (PCA) , a widely used method of data dimension reduction, to transform the multiple variables into a few uncorrelated composite variables for more comprehensive insights into the data set [16]. In Fig. 9, we extract components of flow, pressure, and temperature, compare them with the real components and calculate the dynamic time warping (DTW) [17] distance

between them. The DTW distance measures the similarity of the time series in terms of distribution. As can be seen from the results in the figure, the first principal component of the synthetic data and the real data remain highly similar in time, indicating that most of the samples of the synthetic data and the real data have the same time series characteristics.

PCA and t-SNE [18] dimension reduction analysis results show that the synthetic data precisely cover the real data, indicating that they have a similar distribution.

On the other hand, we took the steam flow data as an example and performed PCA and t-SNE [18] dimension reduction analysis on the data and reduced the whole dataset into two components. Fig. 10 uses the first components as the x-axis, and the second components as the y-axis to represent the whole data distribution. The Figure shows that the points of the synthetic data overlapped highly with the points of the real data in these two principal component dimensions, showing the similarity between the synthetic sample and the real data in the overall data feature distribution.

IV. Conclusion

This study aimed to generate complex steam data in heating systems with limited real data. First, we proposed preprocessing and self-expansion methods to obtain sufficient standard samples for model training. Then, samples were clustered and labeled to obtain separate label and measurement datasets. A DG model was constructed and trained to generate samples with specific labels. Lastly, we reduced excessive noise using the exponential smoothing method to improve the results. The steam data reflecting a petrochemical corporation's steam usage was used as experimental data. Our results demonstrate that the three key features of real and synthetic samples show similar ranges and statistical characteristics with a deviation of 2.7%. The IMFs of real and generated data after EMD are also remarkably similar. Through dimensionality reduction methods like PCA, we observed that the distribution of one feature of real and synthetic data significantly overlap. The mean DTW distance is 16.32, which is relatively small for such time series. Due to a lack of comparative studies on steam data generation, our paper falls short of comparisons with similar studies. Hence, our future work will investigate more evaluation metrics with a comparison between different models. We aim to inspire further research that can help break the obstacles in industrial data sharing and protect business privacy.

Acknowledgment

This work is supported by National Key R&D Program of China (Grant No. 2019YFE0126000). This work is in part supported by National Natural Science Foundation of China (Grant No. 51806190)

References

[1] Yu, C., Bo, Y., 2016. Promoting the clean and efficient use of energy and promoting the healthy and orderly development of cogeneration - interpretation of the management measures of cogeneration. Pet. Chem. Energy Conserv. 000, 4-7.

[2] Guo, Y., Tian, J. Chen, L. Managing energy infrastructure to decarbonize industrial parks in China. Nat Commun 11, 981 (2020). https://doi.org/10.1038/s41467-020-14805-z

[3] Liteng, W., Yu, S., Kong, F., Xinnan, S., Zhou, Y., Zhong, W., & Lin, X. (2020). A study on energy storage characteristics of industrial steam heating system based on dynamic modeling. Energy Reports.

[4] Chen, Zhiqiang et al. "Federated-WDCGAN: A federated smart meter data sharing framework for privacy preservation." Applied Energy (2023): n. pag.

[5] Aslam, S., Herodotou, H., Mohsin, S.M., Javaid, N., Ashraf, N., Aslam, S. (2021). A survey on deep learning methods for power load and renewable energy forecasting in smart microgrids. Renewable and Sustainable Energy Reviews.

[6] Goodfellow, Ian J. et al. "Generative Adversarial Nets." NIPS (2014).

[7] Li, Jianbin et al. "Energy data generation with Wasserstein Deep Convolutional Generative Adversarial Networks." Energy (2022): n. pag.

[8] Odena, Augustus et al. "Conditional Image Synthesis with Auxiliary Classifier GANs." International Conference on Machine Learning (2016).

[9] Yoon, Jinsung et al. "Time-series Generative Adversarial Networks." Neural Information Processing Systems (2019).

[10] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In Proceedings of the ACM Internet Measurement Conference (IMC '20). Association for Computing Machinery, New York, NY, USA, 464–483. https://doi.org/10.1145/3419394.3423643

[11] Luo, Z, Feng, E, Lin, X, & Zhong, W. "Research on Coarse Granularity Data Sample Completion Method for District Heating System." Proceedings of the ASME 2022 International Mechanical Engineering Congress and Exposition. Volume 6: Energy. Columbus, Ohio, USA. October 30–November 3, 2022. V006T08A002.

[12] Asre, Shashank and Adnan Anwar. "Synthetic Energy Data Generation Using Time Variant Generative Adversarial Network." Electronics (2022): n. pag.

[13] Zhang, Chi et al. "Generative Adversarial Network for Synthetic Time Series Data Generation in Smart Grids." 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm) (2018): 1-6.

[14] Sulandari, W., Suhartono, Subanar, & Rodrigues, P.C. (2021). Exponential Smoothing on Modeling and Forecasting Multiple Seasonal Time Series: An Overview. Fluctuation and Noise Letters, 2130003.

[15] Rilling, G., Flandrin, P., Gonçalves, P. (2003). On empirical mode decomposition and its algorithms.

[16] Zhang, Tonglin and B. Yang. "Big Data Dimension Reduction Using PCA." 2016 IEEE International Conference on Smart Cloud (SmartCloud) (2016): 152-157.

[17] Senin, Pavel. "Dynamic Time Warping Algorithm Review." (2008).

[18] Maaten, Laurens van der and Geoffrey E. Hinton. "Visualizing Data using t-SNE." Journal of Machine Learning Research 9 (2008): 2579-2605.