**Aalborg Universitet**

AALBORG UNIVERSITY
DENMARK

# Scenario prediction for power loads using a pixel convolutional neural network and an optimization strategy

Liao, Wenlong; Ge, Leijiao; Bak-Jensen, Birgitte; Pillai, Jayakrishnan Radhakrishna; Yang, Zhe

*Published in:*
Energy Reports

*DOI (link to publication from Publisher):*
10.1016/j.egyr.2022.05.028

*Creative Commons License*
CC BY 4.0

*Publication date:*
2022

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

Link to publication from Aalborg University

*Citation for published version (APA):*
Liao, W., Ge, L., Bak-Jensen, B., Pillai, J. R., & Yang, Z. (2022). Scenario prediction for power loads using a pixel convolutional neural network and an optimization strategy. *Energy Reports*, *8*, 6659-6671. https://doi.org/10.1016/j.egyr.2022.05.028

**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 -

**Take down policy**
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Energy Reports 8 (2022) 6659–6671

Contents lists available at ScienceDirect

Energy Reports

journal homepage: www.elsevier.com/locate/egyr

Research paper

# Scenario prediction for power loads using a pixel convolutional neural network and an optimization strategy☆

Wenlong Liao [a], Leijiao Ge [b], Birgitte Bak-Jensen [a], Jayakrishnan Radhakrishna Pillai [a], Zhe Yang [a,*]

[a] AAU Energy, Aalborg University, Aalborg, Denmark
[b] The Key Laboratory of Smart Grid of Ministry of Education, Tianjin University, Tianjin, China

ARTICLE INFO

*Article history:*
Received 30 January 2022
Received in revised form 20 April 2022
Accepted 10 May 2022
Available online xxxx

*Keywords:*
Scenario prediction
Power load
Pixel convolutional neural network
Deep learning
Stochastic behavior

ABSTRACT

Accurate and reliable prediction of power load is critical to ensure the economy and stability of power systems. However, deterministic point prediction can scarcely be accurate due to the fluctuating and stochastic behavior of power load series, resulting in high risks for the system operation. Scenario prediction is a widely used method to model stochastic behavior by generating a group of possible power load scenarios rather than deterministic point predictions, so that system operators can account for the uncertainty of power loads. In this paper, a new deep generative network-based method is proposed for scenario prediction of power loads, in which structure and parameters are redesigned on the original pixel convolutional neural network (PixelCNN). An optimization model is presented to search for a range of power load scenarios with similar shapes, temporal dependency, and probability distribution as the real ones. Numerical simulations on a real-world power load dataset show that the PixelCNN outperforms other generative networks for the scenario prediction of power loads.

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

## 1. Introduction

Regarding the time horizon, load prediction can be divided into very short-term load prediction for the online monitoring of power equipment, short-term load prediction for daily scheduling plan and weekly scheduling plan, medium-term load prediction for the maintenance plan for equipment, and long-term load prediction for the transformation and expansion of power grid, with the time horizon of minutes, hours, months, and years, respectively (Zhu et al., 2020). This paper focuses on the short-term load prediction, since accurate short-term load prediction is the basis for daily safe and economic operation of power systems.

Traditionally, deterministic point prediction only provides the most likely prediction values as a single estimate for the future power load (Powell et al., 2014). The main methods of deterministic point prediction for the short-term load can be listed as follows: statistical methods and machine learning-based methods. In particular, common statistical methods (Azeem et al., 2021) include the grey prediction model, auto-regressive integrated moving average, auto-regressive moving average, persistence method, and auto-regressive which predict future values only based on historical loads. Their inability to take into account the correlation between loads and other factors (e.g., weather and calendar information) leads to limited prediction accuracy, especially for highly volatile load profiles. To consider the effect of weather and calendar information on prediction accuracy, some machine learning-based methods (e.g., support vector machine, multi-layer perceptron, and random forest) were developed in the early stage (Lindberg et al., 2019), but they failed to model the time temporal of the load curve accurately. To address this issue, various deep neural networks have been proposed recently. Especially, the long short-term memory (LSTM) has shown outstanding performance in capturing the temporal dependence of load curves (Tan et al., 2020; Kong et al., 2019; Li et al., 2021; Lin et al., 2022).

Deterministic point prediction ignores the prediction error caused by the volatility of power loads, which poses potential risks to the operation of power systems. Over the past few decades, a large number of methods have been developed to represent the uncertainty of power loads. Mainstream methods generally include (Zhang et al., 2016): interval prediction (Zhao et al., 2020b), probabilistic prediction (Wen et al., 2021), and scenario prediction (Liao et al., 2021a). Specifically, the interval prediction, and probabilistic prediction are developed based on

☆ This work was supported by the State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology, China [grant number: EERI_KF20200014].

* Corresponding author.
*E-mail addresses:* weli@energy.aau.dk (W. Liao), legendglj99@tju.edu.cn (L. Ge), bbj@energy.aau.dk (B. Bak-Jensen), jrp@energy.aau.dk (J.R. Pillai), zya@energy.aau.dk (Z. Yang).

https://doi.org/10.1016/j.egyr.2022.05.028
2352-4847/© 2022 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

interval and density, respectively. In addition, their predicted values are usually used for probabilistic optimization or interval optimization, which cannot ensure that the dispatch decision is feasible to all extreme load conditions. Relatively, the reliable dispatch decision of robust optimization is always applicable for any common and extreme load conditions, and their solutions are usually derived from the robust optimization model and scenario prediction (Zhao et al., 2020a), which is one of the widely used methods to model the uncertainty of future power load curves by generating a series of possible power load scenarios. Scenario prediction not only captures the temporal dependence of power loads, but also reflects future uncertainty by generating a group of plausible power load curves (Wang et al., 2020b). Especially, scenario prediction plays a role in risk-based decision-making problems (e.g., robust optimization), since these generated power load scenarios can be considered inputs of a robust optimization model to obtain a conservative dispatch decision.

In respect to scenario prediction of power load curves, traditional methods can be divided into two categories: noise-based methods and feature condition-based methods. Specifically, noise-based methods obtain a group of possible scenarios by adding noises to the future power load curves that are coming from conventional deterministic point prediction models. For example, the work in Shepero et al. (2018) assumes that the errors between the real and predicted power loads belong to Gaussian noises, which are added back to the original predictions to obtain a stochastic scenario. In Chen et al. (2019), the Monte Carlo method is employed to calculate prediction errors given the expectation and variations from a deep residual network. In Wang et al. (2018), a quantile regression method is used to fit prediction errors that are conditional on both the input data and point predictions. A key weak point of this category is that it generates scenarios centered on point prediction values, which may not capture the diversity of load behavior, especially when there are multiple patterns in the power load curves. In addition, they require artificial assumptions about the probability distribution that prediction errors obey. (2) For the feature condition-based methods, they make full use of the relationship between power loads and feature conditions (e.g., temperature, humidity, wind speed, weekdays, holidays), and obtain predicted scenarios of power loads according to simulated features. For instance, the work in Dordonnat et al. (2016) produces a range of power loads by inputting simulated feature scenarios to a conventional deterministic point prediction model. In McSharry et al. (2005), a statistical model is designed to fit the main sources of variations in the power load demand and generate possible prediction values of future peak loads. Further, an empirical formula is proposed to quantitatively evaluate temperature scenario methods for scenario prediction of loads in Xie and Hong (2018). Compared with noise-based methods, feature condition-based methods produce more diverse scenarios, but they do not solve the fundamental problem, since they push the problem to how to obtain a good deterministic point prediction model and feature scenarios.

In recent years, many deep generative neural networks have been applied to scenario generations and scenario prediction of power loads and renewable energy sources to overcome the above-mentioned challenges in traditional methods. For example, the works in Zhang and Zhang (2020), Pan et al. (2019) and Ge et al. (2020) use the Variational auto-encoder (VAE) and flow-based generative network to generate scenarios for power loads, while the works in Chen et al. (2018) and Wang et al. (2020a) design different variants of generative adversarial networks (GAN) to model power curves of renewable energy sources, such as wind farms and photovoltaic (PV) plants. Further, an improved GAN model and a non-linear independent component estimation (NICE) model are extended to produce scenarios for wind power and power loads (Jiang et al., 2021; Hu et al., 2021). The simulation results show that the GAN and NICE have stronger performance than traditional methods. However, these generative networks have inaccurate loss functions or training issues, limiting the quality of predicted scenarios significantly. For example, the VAE involves an intractable inference step, while non-convergence and mode collapse problems are still challenges in the GAN (Wu et al., 2020). On these grounds, there is a need to develop a new model with a stable training process and strong performance.

The pixel convolutional neural network (PixelCNN) is a powerful deep generative network with a tractable likelihood. Compared with other generative networks (e.g., GANs), the PixelCNN shows a more stable training process, since it can accurately calculate the likelihood of samples through the chain rule (Oord et al., 2016a). So far, the PixelCNN has shown convincing performance in various fields, such as image generation, missing data imputation, and speech processing (Guo et al., 2017). The successful applications of the PixelCNN in computer vision prove that they can accurately mine the complex intrinsic nature of high-dimensional data (e.g., image and speech signal) through unsupervised training. These unique characteristics of the PixelCNN make it an ideal candidate for the scenario prediction task of power loads. Normally, power load curves are stored as time series, which can be treated as a special image with different sizes of rows and columns. Therefore, the PixelCNN should have the potential to model the uncertainty of power load curves. However, the structures of existing PixelCNN cannot be directly used for the scenario prediction, because the dimension of images significantly varies from power load curves. It is necessary to redesign the structure of PixelCNN for scenario prediction according to the characteristics of power load curves.

In this paper, the PixelCNN is migrated into the scenario prediction task of power loads. Compared with traditional scenario prediction methods (e.g., noise-based methods and feature condition-based methods), the proposed method is free from statistical hypotheses about the probability density function of power load curves or prediction errors. After unsupervised training, it can generate a lot of realistic power load scenarios with similar shapes, temporal dependence, and probability distribution as real ones. Besides, there is no restriction on the number of generated scenarios. Compared with other generative networks, the PixelCNN shows a more stable training process, since it can accurately calculate the likelihood of samples through the chain rule. The key contributions of this paper are:

(1) The PixelCNN is generalized from image generations into the scenario prediction task of power loads. The structure of the original PixelCNN is redesigned to model the uncertainty of load curves.

(2) An optimization strategy is proposed to select a group of possible power load scenarios given deterministic point prediction values and new scenarios from the pre-trained PixelCNN. This optimization strategy can generate a group of possible scenarios for future power loads without any restrictions of time horizons (e.g., hours or days) by simply fine-tuning parameters.

(3) The influence of key hyper-parameters (e.g., the training epoch, optimizer, and learning rate) on the performance is analyzed, and some suggestions about how to select these hyper-parameters are presented.

The rest of the paper is organized as follows. Section 2 introduces the structure and loss function of the PixelCNN, and Section 3 formulates the proposed optimization strategy. Section 4 presents the process of the proposed method. Extensive experiments are performed in Section 5. Finally, Section 6 summarizes the paper and shows possible future works.

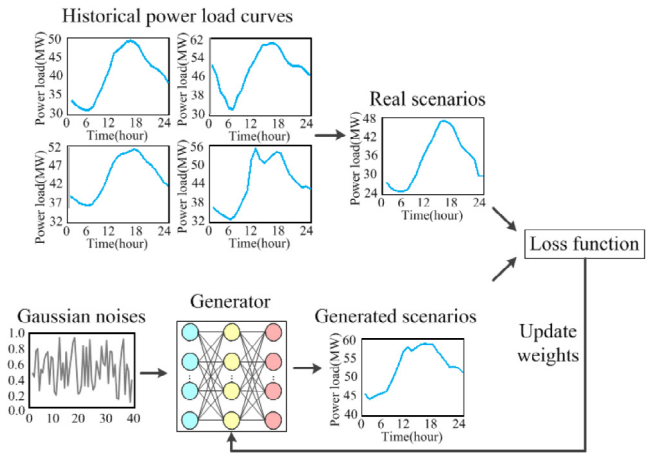


**Fig. 1.** The framework of the PixelCNN.

## 2. Scenario generation using the PixelCNN

### 2.1. The framework of the PixelCNN

The framework of the PixelCNN is shown in Fig. 1. Its main idea is to train a powerful deep neural network based on the historical power load curves, so as to project noise vectors obeying a priori distribution (e.g., Gaussian distribution in this paper) into stochastic scenarios with similar patterns.

$$Z \sim N(0, 1), X = G(Z) \tag{1}$$

where $X$ is the stochastic scenarios; $Z$ is the Gaussian noise; and $G$ is the deep neural network(i.e., the generator in the PixelCNN).

Normally, the distance between the generated stochastic scenarios and the real scenarios is defined as the loss function, which is used to update the parameters (e.g., weights) of the generator in the PixelCNN. Note that noise vectors in the PixelCNN are used to generate new scenarios rather than represent prediction errors like traditional noise-based methods. In other words, the traditional noise-based methods assume that the prediction errors obey the Gaussian distribution, which may be inaccurate. Relatively, the PixelCNN does not require statistical hypotheses about the probability density function of power load curves or prediction errors. After unsupervised training, the PixelCNN can generate a lot of realistic power load scenarios, which will be filtered to a set of suitable scenarios by an optimization model in Section 3.

In the following sections, each module in the PixelCNN will be introduced in detail, including the principles of generators, the structure of generators, and the design of loss functions.

### 2.2. The basic principle of the generator

Without loss of generality, each stochastic scenario can be regarded as a time series $X = \{x_1, x_2, \ldots, x_n\}$, which is composed of $n$ power loads (called pixels in the image generation). Then, the joint probability distribution $P(X)$ of these power loads can be represented as:

$$p(X) = p(x_1, x_2, \ldots, x_n) \tag{2}$$

Further, this joint probability distribution can be factorized, i.e., the time series $X$ can be decomposed into a product of multiple 1-dimensional probability distributions using the chain rule:

$$p(X) = p(x_1)p(x_2)\ldots p(x_n) \tag{3}$$

where $P(x_n)$ is the probability distribution of the power load $x_n$.

Previous publications have shown that power load curves have strong time dependence (Ge et al., 2020). Therefore, for the stochastic scenario generation task of power load curves, this paper employs the first $i-1$ power loads as the input data to estimate the $i$th power load. The stochastic scenario generation proceeds row by row and pixel by pixel. The specific steps are as follows:

(1) Taking $x_1$ as the input data, the first conditional probability $p(x_1, x_2) = p(x_1)p\,(x_2|x_1)$ is utilized to get $x_2$.

(2) The $x_1$ and $x_2$ are input to the second conditional probability $p(x_1, x_2, x_3) = p(x_1)p\,(x_2|x_1)\,p\,(x_3|x_1, x_2)$ to obtain $x_3$.

(3) Similarly, the power loads $x_1, x_2, \ldots, x_{n-1}$ are input to the $(i-1)$th conditional probability to obtain the last power load $x_n$.

(4) Therefore, the joint probability distribution of power loads can be expressed by a product of conditional distributions:

$$p(x_1, \ldots, x_i) = \prod_{i=2}^{n} p\,(x_i|x_1, \ldots, x_{i-1}) \tag{4}$$

where $\Pi$ is the multiplication operation.

After defining the probability distribution of stochastic scenarios, the PixelCNN of power loads can be obtained by maximizing the likelihood of the training samples. Considering that the conditional probabilities of load curves are generally difficult to be accurately represented by mathematical formulas, this paper employs deep neural networks to replace conditional probabilities.

### 2.3. Structural design of the generator

As one of the classical algorithms of deep learning, the convolutional neural network (CNN) is a feed-forward neural network that uses convolutional operations to extract features from high-dimensional data. The CNN has greatly promoted the development of a new generation of artificial intelligence. Compared with recurrent neural networks (RNN), CNN has a more powerful feature extraction ability, and its parallel operation greatly reduces the time cost of model training (Nguyen et al., 2020). At present, CNN has been widely used in drug discovery, fault diagnosis, time series prediction, and style transfer (Liao et al., 2021b). In this context, this paper reshapes the original power load curves into a two-dimensional matrix, and then uses CNN to fit the above-mentioned conditional probability in Section 2.2. In other words, the CNN is employed to construct the generator of the PixelCNN.

As described in Section 2.1, the first $i-1$ power loads are fed to the conditional probability to estimate the $i$th power load, while the standard convolutional operation extracts all the time-series information to estimate the $i$th power load, which means that the standard CNN cannot be directly used to replace the conditional probability. Specifically, Fig. 2(a) shows the standard convolutional operation whose mathematical formula is as follows:

$$X_{\text{cov}}^{l+1} = \sigma_{\text{cov}}\left(X_{\text{cov}}^{l} * W_{\text{cov}} + B_{\text{cov}}\right) \tag{5}$$

where $X_{\text{cov}}^{l}$ is the input data of the $l$th convolutional layer; $\sigma_{\text{cov}}(\cdot)$ is the activation function, such as the linear rectified unit (ReLU) function; $*$ is the convolutional operation; $W_{\text{cov}}$ and $B_{\text{cov}}$ are the weights and offset vectors (i.e., parameters to be trained) of convolutional layers, respectively.

Further, a mask matrix is employed to block the later information before performing the standard convolutional operation, as shown in Fig. 2(b). Specifically, the first $i-1$ value in the mask matrix is 1, and the remaining values are all 0. The mathematical formula of masked convolution is as follows:

$$X_{\text{cov}}^{l+1} = \sigma_{\text{cov}}\left(\left(X_{\text{cov}}^{l} \odot M_s\right) * W_{\text{cov}} + B_{\text{cov}}\right) \tag{6}$$

where $\odot$ is the Hadamard product; and $M_s$ is the mask matrix.

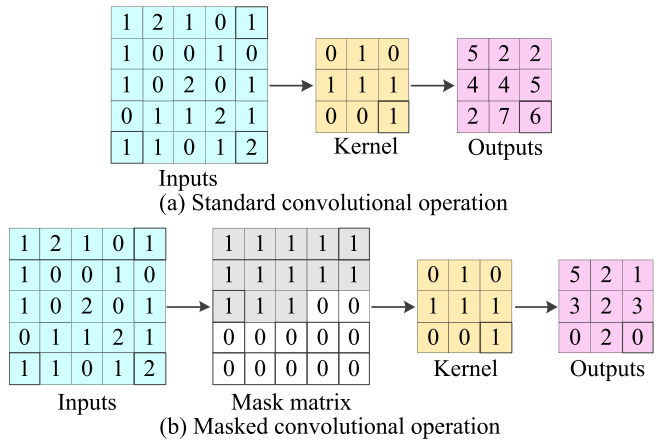


**Fig. 2.** Visualization of the convolutional operations.

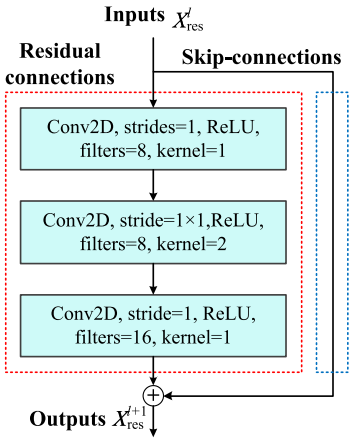


**Fig. 3.** Structure of the residual block.

Generally, the mask convolution is regarded as the input layer of the generator, and its outputs will be fed to the middle layers of the generator. Previous works have shown that appropriately increasing the number of convolutional layers helps to improve the performance of feature learning and generalization (Zhang et al., 2018). However, too many convolutional layers may also lead to over-fitting and network degradation problems. This is because the original data information contained in the extracted features decreases as the number of layers deepens during the forward transmission.

In order to improve the performance of the model and alleviate over-fitting problems, this paper uses residual blocks, which are commonly used in image generations, to form the middle layer of the generator. The structure of the residual block is shown in Fig. 3, and its operation process includes two parts: skip-connections and residual connections. Specifically, the main function of residual connections is to extract latent features through multiple 2-dimensional convolutional (Conv2D) layers, while the skip-connections are to add the original information of input data to the output data. In this case, later layers contain more feature information than the previous layers, and then network degradation can be avoided. The calculation process of residual block can be represented as follows:

$$X_{\text{res}}^{l+1} = X_{\text{res}}^{l} + F_{\text{cov}}\left(X_{\text{res}}^{l}\right) \tag{7}$$

where $F_{\text{cov}}(\cdot)$ denote multiple convolutional layers; and $X_{\text{res}}^{l}$ is the input data of the $l$th residual block. Note that the input data of the first residual block is the output of the masked convolutional layer.

To sum up the structure of the generator, the input layer of the generator is a masked convolutional layer, and the middle layer consists of residual blocks (the number of residual blocks will be explored in the simulation session). The output layer is a standard convolutional layer.

### 2.4. Design of the loss function

For most of existing generative networks (e.g., VAE and GAN), they project $1\times m$ noise vectors into $1\times n$ stochastic scenarios of power loads, and then calculate the statistical distance (e.g., mean square error, Kullback–Leibler divergence, and maximum mean discrepancy) between the real scenarios and generated scenarios as the loss function to update the weights in the model (Gm et al., 2020). Obviously, these generative networks use the real encoding method to represent stochastic scenarios of power load with real numbers ranging from 0 to 1 (Turhan and Bilge, 2018). Normally, the activation function in the last layer can be chosen from the ReLU, sigmoid, and hyperbolic tangent (Tanh), which are suitable for dealing with continuous real numbers.

Previous publications have shown that the one-hot coding method can also be used to represent stochastic scenarios for the calculation of loss functions (Oord et al., 2016b). Compared with the traditional real encoding method, the loss function encoded by the one-hot coding method is not only easy to train, but also has better performance. Therefore, this paper chooses the one-hot coding method to encode stochastic scenarios of power loads.

Specifically, the PixelCNN projects a noise vector into a stochastic scenario of power loads, which is represented as using an $n\times k$ 2-dimensional matrix by the one-hot coding method (Liao et al., 2021b). The categorical cross-entropy is used to measure the statistical distance between the real and generated scenarios:

$$L = \sum_{i=1}^{k} y_i \lg\left(p_i\right) \tag{8}$$

where $L$ is the categorical cross-entropy; $p_i$ is the output of neural networks, i.e., the probability that the sample belongs to class $i$; $k$ is an integer that represents the length of the outputs; and $y_i$ is a label. $y_i$ is equal to 1, if the sample belongs to class $i$, otherwise $y_i$ is equal to 0.

Note that this $n\times k$ 2-dimensional matrix can be decoded into a $1\times n$ integer vector, where the element is divided by $k$ to obtain a stochastic scenario of power loads. Normally, the activation function of the last layer in the PixelCNN selects the softmax function, which is suitable for dealing with binary numbers.

## 3. An optimization model for scenario prediction

Obviously, a large number of stochastic scenarios produced by feeding noise vectors into the pre-trained PixelCNN are disordered, i.e., generated power load curves are not correlated with the estimated values from deterministic point prediction models. To make full use of the historical loads from the past time $t-h$ to time $t$ and the estimated loads from the future time $t+1$ to time $t+m$ from the deterministic point prediction models, an optimization problem is formulated to achieve this goal. $m$ denotes time horizons of predictions.

Fig. 4 visualizes the framework of the optimization strategy. First of all, Gaussian noises are input to the pre-trained PixelCNN to generate a large number of disordered stochastic scenarios. Then, an optimization model with the objective function and constraints are defined given deterministic point prediction values from time $t+1$ to time $t+m$, historical power load from time

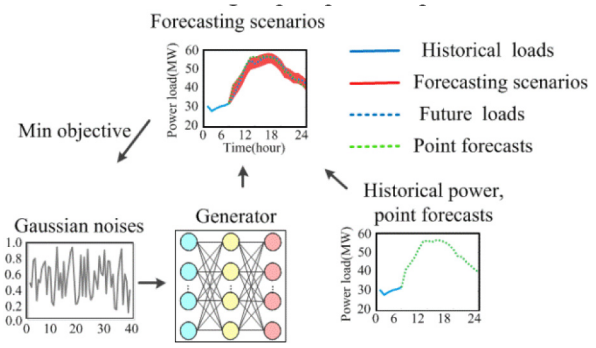


**Fig. 4.** The framework of the optimization strategy.

$t-h$ to time $t$, and generated stochastic scenarios from time $t-h$ to time $t+m$. Lastly, the optimization model is solved to obtain a group of possible power load scenarios that may occur.

Suppose $X_{\text{hist}} = (x_t, x_{t-1}, \ldots, x_{t-h})$ be power loads from time $t-h$ to time $t$, and $X_{\text{pred}} = (x_{t+1}, x_{t+2}, \ldots, x_{t+m})$ be deterministic point prediction values from time $t+1$ to time $t+m$. The pre-trained PixelCNN is used to model the uncertainty of power load curves by generating a lot of possible stochastic scenarios $S = (X'_1, X'_2, \ldots, X'_N)$. For ease of discussion, the generated stochastic scenario $X'_i$ is divided into two components, which include the former $X'_{\text{former}}$ from time $t-h$ to time $t$ and the latter $X'_{\text{latter}}$ from time $t+1$ to time $t+m$:

$$X'_i = \left[X'_{\text{former}}, X'_{\text{latter}}\right] = G(Z) \tag{9}$$

$$X'_{\text{former}} = (x'_t, x'_{t-1}, \ldots, x'_{t-h}); X'_{\text{latter}} = (x'_{t+1}, x'_{t+2}, \ldots, x'_{t+m}) \tag{10}$$

Normally, the generated power load curves $S$ are disordered. In order to screen out some stochastic scenarios related to the deterministic point prediction values, the following two conditions should be satisfied to capture the uncertainty of power load curves:

(1) The former $X'_{\text{former}}$ of generated power load curves from time $t-h$ to time $t$ should be close to the historical power load $X_{\text{hist}}$ at the same moment.

(2) The latter $X'_{\text{latter}}$ of generated power load curves from time $t+1$ to time $t+m$ should be realistic and around the deterministic point prediction values $X_{\text{pred}}$.

Note that the above-mentioned stochastic scenarios $S$ are not derived from historical power load curves of the database, but are generated by a generative network, which is trained with historical power load curves. The reason for this is that the data of historical load curves are generally insufficient, which makes it difficult to fully cover the different changes that may occur in the future power loads. In contrast, the generative network can not only produce power curves with similar characteristics to the training samples, but also have strong generalization ability (Pan et al., 2019; Ge et al., 2020; Chen et al., 2018; Wang et al., 2020a), i.e., it can generate power load scenarios that are not in the training set but may appear.

To ensure that generated stochastic scenarios do not conflict with the above-mentioned two conditions, a constrained optimization model is defined as follows:

$$\begin{aligned} &\min_z \left\| X_{\text{hist}} - X'_{\text{former}} \right\|_2 \\ &s.t.\ z \in Z \\ &L_\alpha(X_{\text{pred}}) \le X'_{\text{latter}} \le U_\alpha(X_{\text{pred}}) \end{aligned} \tag{11}$$

where $L_\alpha(X_{\text{pred}})$ is the lower bound of prediction intervals (PIs); $U_\alpha(X_{\text{pred}})$ is the upper bound of PIs; and the parameter $\alpha$ (also called the prediction confidence) is used to control the width of PIs.

There exist a large number of methods for constructing PIs, such as Gaussian (Wan et al., 2017), Delta (Khosravi et al., 2010), mean–variance estimation (Khosravi et al., 2012), and bootstrap techniques (Khosravi et al., 2015). It is hard to determine which method is the best, because each method has its own advantages and characteristics. Further, the PIs from the previous works (Liao et al., 2021a; Chen et al., 2018) are employed as a simple example to facilitate the comparison between baselines and the proposed method:

$$\begin{aligned} L_\alpha(X_{\text{pred}}) &= X_{\text{pred}} - \alpha \times \max(X_{\text{pred}}) \\ U_\alpha(X_{\text{pred}}) &= X_{\text{pred}} + \alpha \times \max(X_{\text{pred}}) \end{aligned} \tag{12}$$

Compared with constrained optimization problems, unconstrained optimization problems are easier to solve. In light of this, the proposed constrained optimization model is transformed into an unconstrained optimization model by replacing constraints with penalty terms in the objective function:

$$\begin{aligned} &\min_z \left\| X_{\text{hist}} - X'_{\text{former}} \right\|_2 + \\ &M\left[\varepsilon\left(X'_{\text{latter}} - U_\alpha(X_{\text{pred}})\right) + \varepsilon\left(L_\alpha(X_{\text{pred}}) - X'_{\text{latter}}\right)\right] \end{aligned} \tag{13}$$

where $M$ is a penalty coefficient, which is much larger than the first term of the objective function; and $\varepsilon(\cdot)$ is the Heaviside function.

## 4. Process of the proposed method

The framework of scenario prediction for power loads based on PixelCNN is shown in Fig. 5. Firstly, historical power load curves are utilized to train a PixelCNN. Secondly, historical power load curves and corresponding feature conditions (e.g., temperature, humidity, wind speed, weekdays, weekends, holidays) are considered as the input data of a deterministic point prediction model (e.g., RNN) to obtain the estimated power loads from time $t+1$ to time $t+m$. To obtain a set of possible scenarios, an optimization algorithm (e.g., genetic algorithm) is employed to solve the proposed unconstrained optimization model. Finally, some indicators are proposed to analyze results. The detailed steps are as follows:

(1) Normalize and divide datasets.

The 80% and 10% of measured power load time series curves are randomly selected as the training set and validation set, respectively. The remaining part is used as a test set to evaluate the performance of the proposed method. Before inputting power load curves to the PixelCNN and deterministic point prediction model, the power loads and the corresponding feature conditions need to be normalized, and otherwise the loss functions of models may fail to converge. Therefore, this paper employs the min–max normalization method to transform the original data into values that vary from 0 to 1.

(2) Train the PixelCNN.

After initializing the structure and parameters of PixelCNN, the weights in the model are updated by unsupervised training. As a most widely used algorithm for training neural networks, the back propagation can calculate how much each weight contributes to this error and the amount that needs to be adjusted. Note that heuristic algorithms (e.g., genetic algorithms) can also be used to optimize the weights of neural networks for early-stage shallow neural networks (e.g., multi-layer perceptron). However, deep neural networks have so many parameters (e.g., millions of parameters) that it is difficult for heuristic algorithms to optimize such a large number of parameters, while back propagation algorithms can quickly optimize the weights by employing an optimizer such as gradient descent. Therefore,

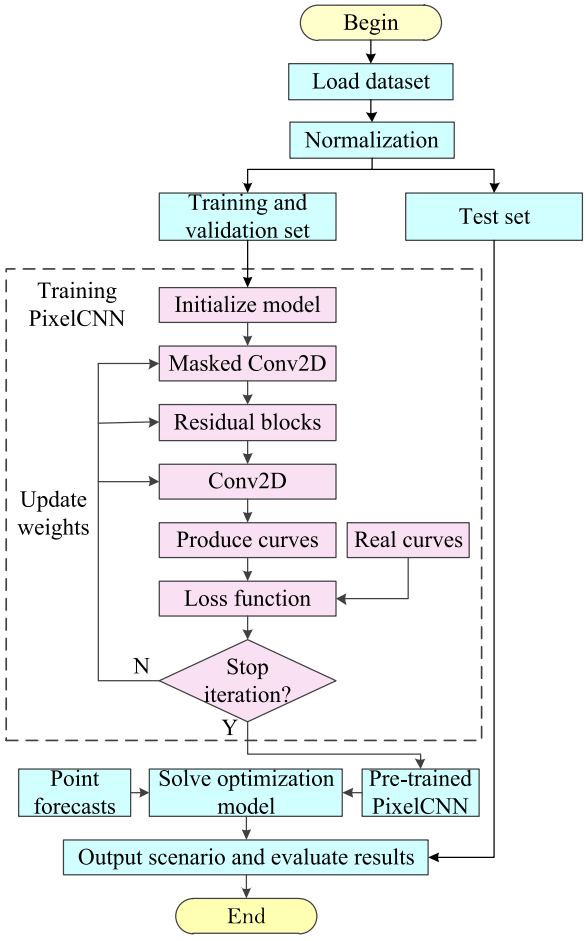


**Fig. 5.** Process of the proposed method.

the back propagation is employed to train the proposed model through an optimizer, which will be discussed in the simulation section below. Specifically, the Gaussian distribution (The mean is 0 and the variance is 1) is sampled to obtain a set of noise vectors as the input data of the PixelCNN. The noise vectors are processed by a masked convolutional layer and residual blocks to form stochastic scenarios with the same dimension as real ones. Then, the categorical cross-entropy loss function between the generated scenarios and the real scenarios is calculated to update the weights of the PixelCNN through the back propagation algorithm. When the training epoch exceeds the pre-set size, the trained PixelCNN is used for the stochastic scenario generation task of power loads.

(3) Train a deterministic point prediction model.

The conventional deterministic point prediction model is utilized to estimate power loads from time $t+1$ to time $t+m$, which is part of the input data of the unconstrained optimization model. To make predicted scenarios cover real power loads as much as possible, a point prediction model with strong performance is necessary, since generated power load curves should be around deterministic point prediction values. In other words, a good point prediction model helps the predicted scenarios to cover the real power loads with a small PI. Previous works have shown that LSTM can solve the long-term dependence problem in traditional RNNs, and achieve state-of-art performance with superior accuracy in short-term power load prediction (Tan et al., 2020; Kong et al., 2019; Li et al., 2021; Lin et al., 2022). Therefore, this paper selects LSTM as the deterministic point prediction model to estimate the short-term power loads. Normally, the control variable method (Ge et al., 2020) is used to adjust the structure and parameters of LSTM, which are related to the prediction accuracy. After many tests, a suitable structure of LSTM is determined as follows:

The middle layer consists of 4 LSTM layers, and their units are 20, 15, 10, and 10, respectively. The dense layer with 1 unit is used as the output layer. The activation function in the output layer is the sigmoid function, and the activation functions of other layers are ReLU functions.

(4) Solve an unconstrained optimization model.

The genetic algorithm (GA) is a kind of population evolutionary algorithm which simulates the process of biological evolution. At present, GA has been widely used in different fields (e.g., signal processing, combinatorial optimization, machine learning) because of its strong global optimization ability (Abdelhady et al., 2020). Therefore, this paper employs GA to solve the proposed unconstrained optimization model, so as to get a group of possible scenarios to represent the uncertainty of power load curves. After adjusting the parameters by the control variable method (Ge et al., 2020), the structure and parameters of the GA are determined as follows:

The variable to be optimized is the noise vector z. The fitness function is Eq. Eq. (13). The population size of the chromosome is 50, the probability of crossover operation is 0.9, and the probability of variation operation is 0.1. The training epoch is 400.

(5) Evaluate results.

Normally, the prediction interval normalized average width (PINAW) and prediction interval coverage percentage (PICP) are used as indicators to evaluate the results of predicted scenarios.

Specifically, the PICP is the probability that the real power loads fall within prediction intervals(PIs) of the generated scenario sets. When parameter $\alpha$ is fixed, the larger the PICP, the better the performance of the model.

$$PICP = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} c_i \tag{14}$$

where $N_{\text{test}}$ is the number of sample in the test set; and $c_i$ is a Boolean values. If the prediction value is out of bounds, $c_i = 0$, and otherwise $c_i = 1$.

Obviously, the PICP is positively correlated with parameter $\alpha$. To avoid the single pursuit of interval coverage and unlimited increase of parameter $\alpha$, the PINAW should be considered:

$$PINAW = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (U_i - L_i) \tag{15}$$

where $U_i$ is the upper bound of PIs for the $i$th sample; $L_i$ is the lower bound of PIs for the $i$th sample. When other parameters are the same, the smaller the PINAW, the better the performance of the model.

## 5. Case study

### 5.1. Data description and simulation platforms

To verify the predictive superiority of the proposed method for the scenario prediction of power loads, numerical simulations are performed on a real-world power load dataset from the University of Texas at Austin Powell et al. (2014). This dataset records the hourly cooling, heating, and power load, and corresponding feature conditions (e.g., wind speed, humidity, pressure, and dry bulb temperature) from July 2011 to September 2012. The lack of weekdays, weekends, and holidays in the feature conditions of this dataset may slightly limit the performance of the LSTM, but it does not matter. Any point prediction model cannot perfectly

**Table 1**
Average results of models with different structures.

| Number of residual blocks | Loss function values of the validation set | Training time (min) | Number of parameters |
|---|---|---|---|
| 2 | 1.238 | 17.29 | 162 944 |
| 4 | 0.465 | 28.28 | 269 952 |
| 6 | 0.314 | 37.48 | 376 960 |
| 8 | 0.272 | 47.44 | 483 968 |
| 10 | 0.259 | 57.12 | 590 976 |
| 12 | 0.224 | 70.81 | 697 984 |
| 14 | 0.215 | 91.55 | 804 992 |
| 16 | 0.200 | 107.19 | 912 000 |
| 18 | 0.218 | 122.73 | 1 019 008 |
| 20 | 0.216 | 133.23 | 1 126 016 |

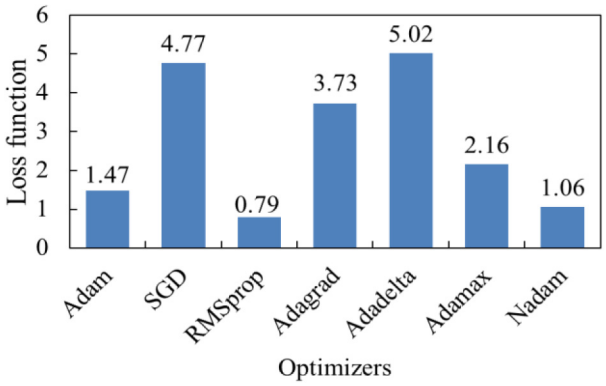


**Fig. 6.** The average loss functions of models with different optimizers.

predict the real future load without errors, which is why the proposed method is needed to represent the uncertainty of power loads. The case study focuses on whether the generated scenario can cover real power load curves and capture the time correlation and probability density functions of power load curves.

To ensure the diversity of power load curves used for training, verifying, and testing the model, the samples are randomly split into the training set, validation set, and test set without overlap. The proportion of the training set is 80%, and the proportion of the validation set and test set is 10%. Further, historical loads and historical meteorological features are fed into the pre-trained LSTM to obtain deterministic point prediction values of power loads from time $t+1$ to time $t+m$. Note that the focus of this paper is not on point prediction models. In case the reader is interested in point prediction models, the inputs and outputs of point prediction models can be found in Powell et al. (2014), and the principles of LSTM can be found in Tan et al. (2020), Kong et al. (2019), Li et al. (2021) and Lin et al. (2022).

All programs in this paper are tested in an integrated development environment (IDE) named Spyder 3.6. The programming language is Python 3.7, and the deep learning frameworks used are Keras 2.2.4 and Tensorflow 1.12.0.

### 5.2. Discussion on hyper-parameters

Before training the PixelCNN, it is necessary to initialize the hyper-parameters, which have a great impact on the performance of the model. The hyper-parameters of the PixelCNN mainly include the numbers of residual blocks, parameter $k$ in the loss function, optimizer and its learning rate (LR), and training epoch. It is difficult to give perfect parameters suitable for all data sets, so this section shows how to use the control variable method (Ge et al., 2020) to fine-tune these hyper-parameters on a specific dataset. Lastly, a good starting point for each parameter is given. Specifically, when one of the parameters is adjusted, the other parameters use the initial values:

The middle layer includes 3 residual blocks. The optimizer is the root mean square propagation (RMSprop) algorithm, and its learning rate is 0.001. The training epoch is 1000, and parameter $k$ is 256.

To find a suitable structure for the middle layer of the generator, the number of residual blocks is varied from 2 to 20, and then the models with different structures are trained 30 times. Table 1 shows the average loss functions of the validation set, training time, and the number of parameters.

As the number of residual blocks increases, the loss function value first decreases and then increases, while the training time and the parameters to be trained increase linearly. When there are a few numbers of residual blocks in the middle layer, the PixelCNN can only learn a few useful features from power load curves, and lose some important information due to the limited capacity of the model. Relatively, if the middle layer consists of a large number of residual blocks, the PixelCNN will learn too many representations which are specific to the training set, and do not generalize to other data, such as the validation set and test set. 16 is a good starting point for the number of residual blocks in the middle layer, and higher values or lower values may be fine for some datasets.

There is a need to choose an optimizer to update the weights of the PixelCNN after initializing the middle layer. The popular optimizers include the root mean square prop (RMSProp), Nesterov-accelerated adaptive moment estimation (Nadam), adaptive delta (Adadelta), adaptive moment estimation extension based on infinity norm (Adamax), adaptive moment estimation (Adam), adaptive gradient descent algorithm (Adagrad), and stochastic gradient descent (SGD). The models with different optimizers are trained 30 times, respectively. Their average loss functions of the validation set are shown in Fig. 6.

The PixelCNN can achieve good performance when Nadam, Adam, and RMSprop algorithms are used as optimizers. Among them, the RMSprop algorithm is the optimal optimizer for the PixelCNN, because it has the smallest loss function. In addition, the average loss functions of Adadelta, Adamax, SGD, and Adagrad are significantly larger than those of other optimizers, indicating that they are not suitable for the PixelCNN in scenario generations of power loads.

Before using the selected optimizer, the LR needs to be set to control the speed at which the model learns. Normally, LR is a small positive value, which ranges from 0 to 1. In order to find an appropriate LR, the models with different learning rates are trained 30 times, respectively. The training epoch is extended to 2000. The average loss functions of the validation set are shown in Figs. 7 and 8.

The following conclusions can be drawn from Figs. 7 and 8: (1) A too large LR (e.g., LR $=0.1$) will cause the gradient to remain almost constant rather than reducing the training error. When the LR is equal to 0.01, the loss function of the PixelCNN has converged after 500 training epochs, but it oscillates severely during the training period. (2) On the contrary, if the LR is too small (e.g., LR $=0.0001$ and LR $=0.00001$), convergence speed is not only very slow, but stuck with a high training error. Therefore, the PixelCNN should not use an LR that is too large or too small. The suitable range of LR is less than 0.01 and greater than 0.0001, and a default value of 0.001 may work well for other datasets. (3) On the premise of selecting an appropriate LR, the loss function value of the PixelCNN converges to a constant after 1600 training epochs, which indicates that the PixelCNN has been trained well.

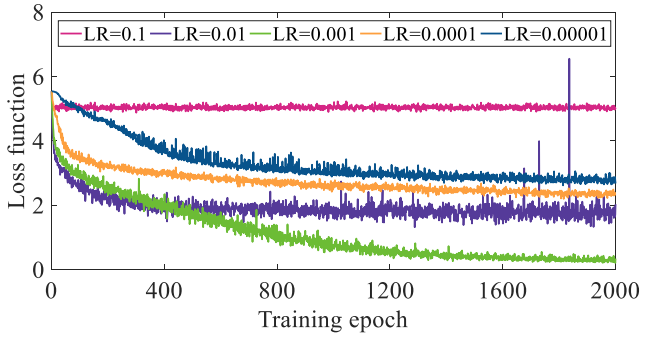


**Fig. 7.** The training process of different learning rates.

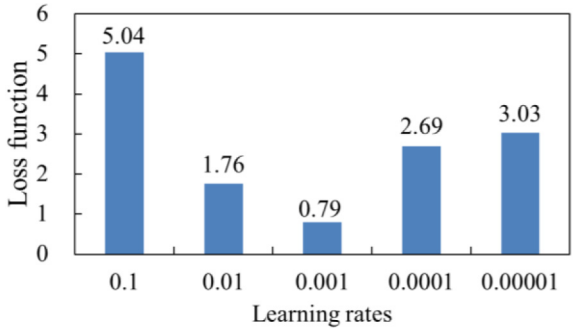


**Fig. 8.** The average loss functions of models with different learning rates.

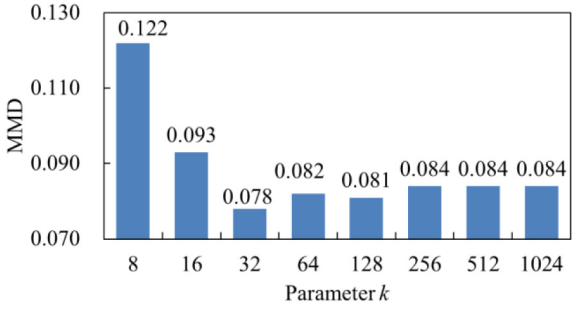


**Fig. 9.** MMD between generated samples and real samples.

Unlike other generative networks (e.g., GAN) where the loss functions involve unstable training problems (e.g., non-convergence and mode collapse), the training process of the PixelCNN is very stable.

The parameter $k$ is an integer that denotes the length of the outputs. In the field of image generations, the parameter $k$ is 256, representing different RGB color codes. However, 256 may not be the best choice of the parameter $k$ for scenario generation of power load curves, which makes it necessary to explore a suitable parameter $k$ for the PixelCNN.

Considering that the change of parameter $k$ will affect the loss function, it is unfair to compare the loss function values of models with different sizes of parameter $k$. Therefore, maximum mean discrepancy (MMD) is employed to evaluate the performance of models by comparing the statistical distance between the generated samples and the real samples (Liao et al., 2021a). The models with different parameters are trained 30 times, respectively. The average MMD between generated samples and real samples are shown in Fig. 9.

Similar to LR, the parameter $k$ should not be too large or too small, and a suitable parameter $k$ for scenario generation of power load curves varies from 32 to 128.

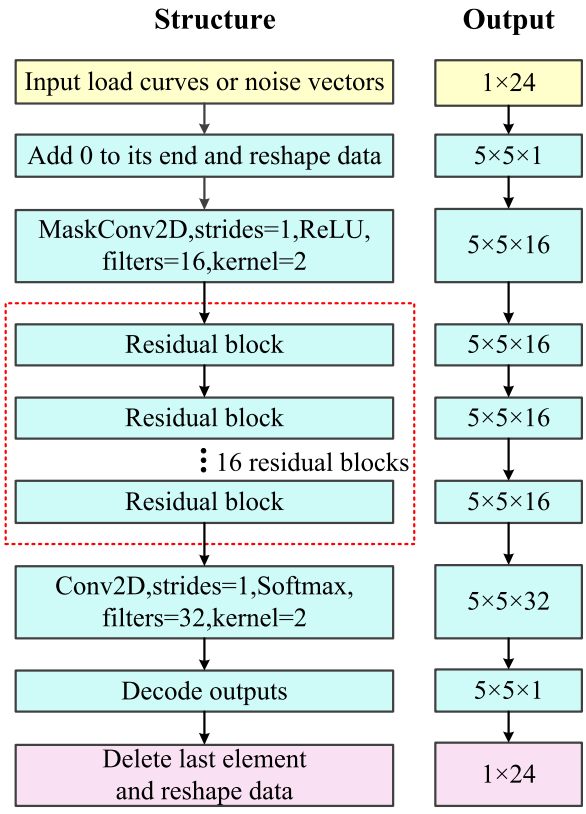


**Fig. 10.** A simple PixelCNN for scenario generation.

To sum up, hyper-parameters of the PixelCNN are set as follows: The middle layer includes 16 residual blocks. The optimizer is the RMSprop algorithm, and its learning rate is 0.001. The training epoch is 2000, and parameter $k$ is 32.

As a simple example, Fig. 10 shows the structure of the PixelCNN for scenario generation of daily power load curves with 1-hour time resolution. The structure and parameters of the PixelCNN only need to be simply fine-tuned for datasets with different time resolutions. Specifically, a 0 element is added to the end of power load curves or noise vector to form a $1 \times 25$ vector, which is reshaped into a $5 \times 5 \times 1$ tensor as the input data of the PixelCNN. The input layer is a 2-dimensional masked convolutional (MaskConv2D) layer, which extracts features and blocks unnecessary temporal information. The middle layer consists of 16 residual blocks whose parameters and structure are shown in Fig. 3. A 2-dimensional convolution is employed as the output layer to output $5 \times 5 \times 32$ tensor, which is decoded into a $5 \times 5 \times 1$ tensor using one-hot coding method. Finally, the last element is discarded, and the output vector is reshaped into a $1 \times 24$ generated load curve. The output layer uses the Softmax function as the activation function, and the other layers use the ReLU function as the activation function.

### 5.3. Results of scenario generation using PixelCNN

To visualize the adaptability of the proposed method to stochastic scenarios with different prediction time horizons, a sample in the test set is randomly selected, and then the GA is employed to optimize noise vectors to obtain a group of scenarios ranging from 6 h to 24 h. Among them, 100 scenarios are randomly selected, as shown in Fig. 11.

By simply changing the parameter m (a parameter to control time horizon), PixelCNN can easily generate reliable power load

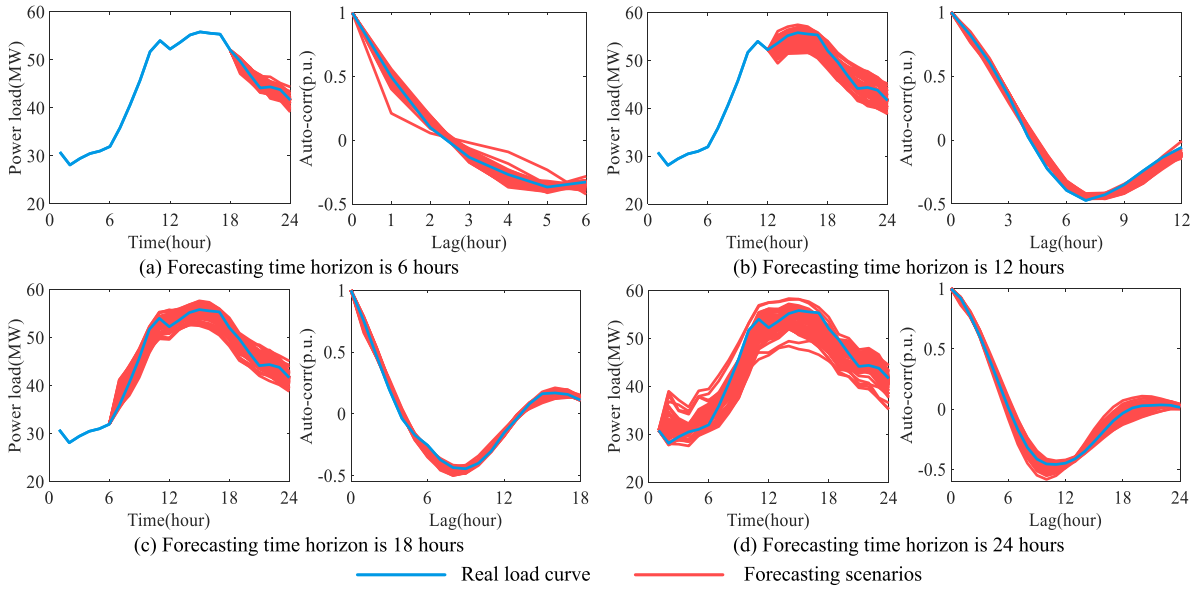


**Fig. 11.** Real scenarios and generated scenarios with different time horizons.

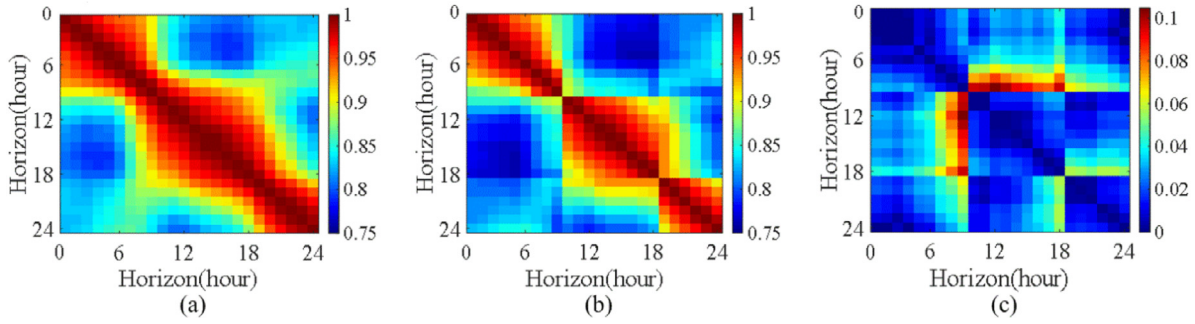


**Fig. 12.** The Pearson correlation matrices. (a) Pearson correlation matrix of real power loads; (b) Pearson correlation matrix of generated power loads; (c) Error of Pearson correlation matrices between real power loads and generated ones.

scenarios with different time horizons, which cover real power load curves. In addition, the auto-correlation (Auto-corr) functions (Zhao et al., 2020a) of the generated power load scenarios are extremely close to the real ones, indicating that generated power load scenarios can represent the temporal dependence of different time horizons.

In addition to comparing the auto-correlation function in single time series, previous publications often use the Pearson correlation coefficient to analyze whether all the generated scenarios have a similar temporal correlation with real ones (Ge et al., 2020; Chen et al., 2018). Therefore, Fig. 12 visualizes the heat map of Pearson correlation matrices of 24 h real power loads, along with 3000 generated power load scenarios for each realization.

In Fig. 12, the $x$-axis and $y$-axis denote the time horizon. Obviously, the errors of Pearson correlation matrices between real power loads and generated ones are smaller than 0.13. Similar values in Pearson correlation matrices of real and generated power loads indicate that the PixelCNN can accurately capture the temporal dependency of power load curves with different time horizons.

Normally, the probability density function is widely used to evaluate the overall similarity between the generated samples and the actual samples. In order to analyze the ability of PixelCNN to capture the probability distribution characteristics of power load curves, the popular generative networks (e.g., VAE, GAN, and NICE) are used as baselines to generate a large number of scenarios, and then the proposed optimization model is solved to obtain a group of suitable scenarios. Similarly, the control variable method is employed to select the structure and parameters as follows:

(1) VAE consists of an encoder and a decoder. Specifically, the encoder includes 4 Conv2D layers, 1 flatten layer, and 1 dense layer. The size of strides in the first 2 Conv2D layers is 2, and that in the last 2 Conv2D layers is 1. The size of all convolutional kernels is 2. The number of neurons in the dense layer is 64. The decoder includes 2 dense layers, 3 transposed Conv2D layers, and 1 Conv2D layer. The number of neurons in the dense layer is 64 and 144, respectively. The size of strides in the first 2 transposed Conv2D layers is 1, and that in the last transposed Conv2D layer is 2. The size of all transposed convolutional kernels is 2. The size of the stride is 1, and the convolutional kernel is 2 in the Conv2D layer. The Conv2D layer of the decoder uses the sigmoid function as the activation function, and other layers of the VAE use the ReLU function as the activation function. Lastly, the optimizer is Adam optimization, and the training epoch is 300. (2) GAN consists of a generator and a discriminator. Specifically, the generator includes a dense layer and 4 transposed Conv2D layers. The number of neurons in the dense layer is 256. The size of strides is 1 and size of transposed convolutional kernels is 2. The last transposed Conv2D layer uses the Tanh function as the activation function, and other layers use ReLU functions. The sizes of filters in transposed Conv2D layers are 32,16,8, and 1, respectively. The discriminator include 3 Conv2D layers, 1 flatten layer, and 1 dense layer. The size of strides is 2, and size of transposed

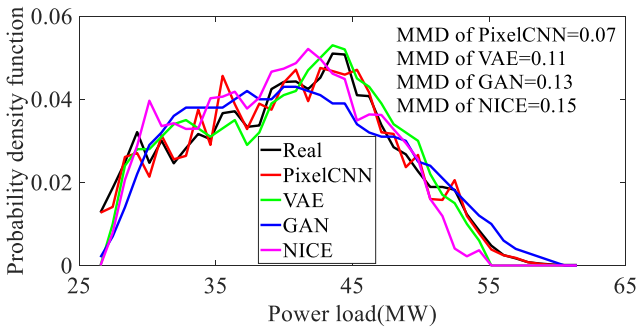


**Fig. 13.** The probability density functions of the real samples and generated samples.

**Table 2**
The average computation time for different models.

| Model | Time to train the model (min) | Time to generate scenarios (s) |
|---|---|---|
| PixelCNN | 204.38 | 219.32 |
| VAE | 1.76 | 0.45 |
| GAN | 14.47 | 0.29 |
| NICE | 5.83 | 0.64 |

convolutional kernels is 2. The last Conv2D layer uses the sigmoid function as the activation function, and other layers use Leaky ReLU functions. The sizes of filters in Conv2D layers are 8, 16, and 32, respectively. The number of neurons in the dense layer is 1. In the end, the optimizer is Adam optimization, and the training epoch is 10000. (3) NICE consists of four additive coupling layers whose parameters can be found in Hu et al. (2021).

Fig. 13 shows the probability density functions of the real samples and 3000 generated samples. In addition, the statistical distance between the generated samples and the real samples is evaluated by the MMD.

In general, the probability density functions of power load scenarios generated by these four generative models are basically consistent with the real ones. This phenomenon shows that they can account for the probability distribution characteristics of power load scenarios through unsupervised learning. Further, the PixelCNN significantly outperforms popular baselines (e.g., VAE, GAN, and NICE), since the MMD between real samples and fake samples generated by the PixelCNN is smaller than those of the VAE, GAN, and NICE.

Further, the scenario generation of 3000 new samples is considered as an example. Each model is repeated 30 times, respectively. The average computation time of each model is shown in Table 2. Specifically, PixelCNN's training time and running time for generating new scenarios are much longer than other generative models (e.g., VAE, GAN, and NICE), because its stochastic scenario generation proceeds row by row and pixel by pixel, as described in Section 2.2. The large time consumption is the main limitation of PixelCNN. Note that a few hours of training time and a few minutes of running time to generate scenarios are acceptable in real engineering. For example, there is no real-time requirement when the system operators use the generated load scenarios as inputs to the day-ahead robust optimization scheduling of power systems.

### 5.4. Results of scenario prediction

In order to visualize the effectiveness of the proposed method for the scenario prediction of power loads, the LSTM is utilized to obtain deterministic point prediction values of 3 power load curves randomly selected from the test set, and the GA is employed to search for a group of appropriate scenarios by solving unconstrained optimization model in Section 3. For each real power load curve and point prediction value, 100 predicted scenarios are randomly selected. Then, Fig. 14 shows the shapes of real power load curves, point prediction values, predicted scenarios, and their distribution characteristics.

The following conclusions can be drawn from Fig. 14: (1) From the box-plot, it can be found that there are large differences in the data distributions between the point prediction values and real values. For example, when the forecasting time horizon is 48 h, the maximum point prediction value is much smaller than the real peak. If the point prediction values are used to obtain the dispatching scheme of power systems, the point prediction-based solutions may not guarantee the security of power systems. The predicted scenarios can cover the full range of real values if the parameter $\alpha$ is large enough. In this case, a group of realistic scenarios generated by the proposed method can be utilized for risk-based decision-making problems (e.g., robust optimization) with strong robustness for large error cases. (2) The balance between sharpness and reliability can be controlled by fine-tuning the parameter $\alpha$. The increase of the parameter $\alpha$ will make PINAW and PICP increase at the same time. For example, when parameter $\alpha$ is 1.1 and the prediction time horizon is 48 h or 72 h, the predicted power load scenarios are close to deterministic point prediction values, but they may not be able to cover the peak or valley of real power load curves. With the increase of parameter $\alpha$, although predicted power load scenarios are less concentrated, they are more likely to cover all the real power loads. Generally, parameter $\alpha$ can be fine-tuned freely based on the system operator's requirements.

Besides qualitative observations, the performance of the proposed method is also verified by quantitatively calculating the PICP and PINAW of different models. The proposed method, the VAE, GAN, and NICE are trained 30 times, respectively. The average PICP and PINAW of the test set are shown in Table 3.

The following conclusions can be drawn from Table 3:

(1) Obviously, NICE is worse than the VAE, because it has a larger PINAW than VAE and a smaller PICP than VAE. Although the PINAW of the VAE is the smallest, it also has the worst relatively worse. In contrast, the PICP of GAN is the largest, but it pays the price, i.e., the PINAW of the GAN is much larger than those of other generative networks. The PICP of PixelCNN is not only much larger than that of VAE, and its PINAW is smaller than that of GAN. In other words, the PixelCNN keeps the trade-off between PINAW and PICP well. For example, when the prediction time horizon is 24 h and the parameter $\alpha$ is 1.1, the PICPs of the PixelCNN and the GAN are increased by approximately 26.82% and 28.84% compared with the VAE, and the PINAWs are approximately 28.93% and 70.87% higher than that of the VAE. Practically, the predicted scenarios produced by the VAE fail to cover as many real power loads as possible. It is hard to ensure the security of power systems (i.e., small PIs may not cope with possible extreme cases). For GAN, the PINAW of the predicted scenarios is too large, which will increase the reserve capacity for risk-based decision-making problems of power systems (i.e., economic waste is caused by excess reserve capacity). The PixelCNN is superior to VAE, GAN, and NICE in taking into account both PINAW and PICP. In other words, PixelCNN balances economy and security.

(2) When parameter $\alpha$ is fixed, the larger the prediction time horizon, the smaller the PICP of each model. For example, when the parameter $\alpha$ is 1.2 in the PixelCNN, the PICP of the 48-hour time horizon and 72-hour time horizon are reduced by approximately 6.24% and 10.47% compared with the 24-hour time horizon. This is because the accuracy of deterministic point prediction models deteriorates with the increase of time horizons, i.e., the increase of time horizons intensifies the uncertainty of power loads.

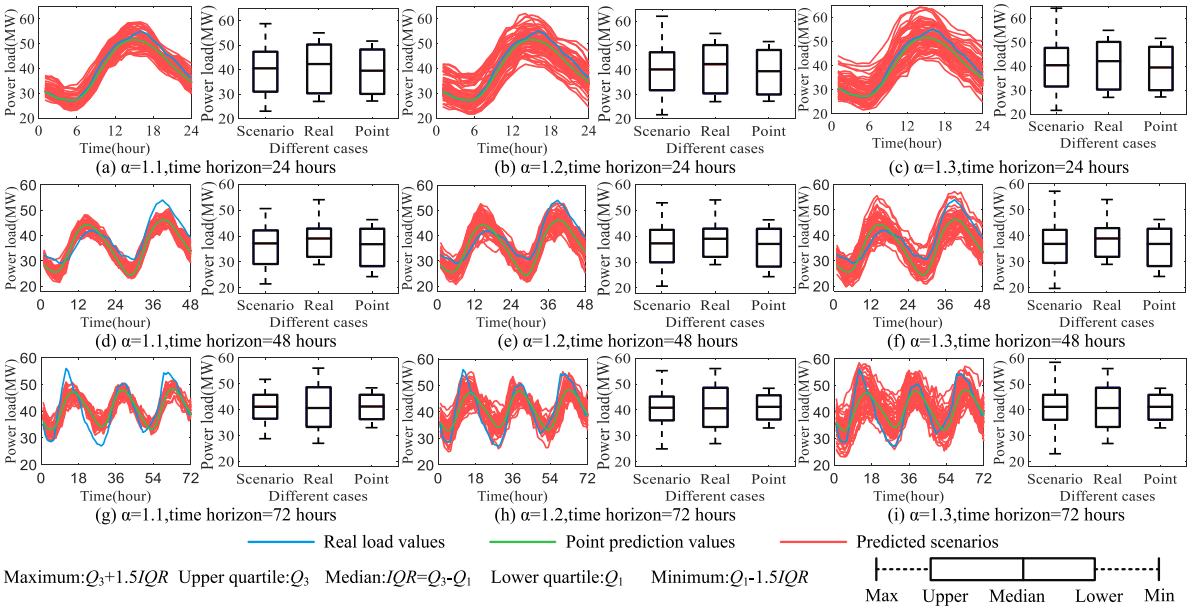


**Fig. 14.** Real values, point prediction values, and predicted scenarios of different parameters.

**Table 3**
Average PICP and PINAW of the test set.

| Different parameters | | PINAW (MW) | | | | PICP (%) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | PixelCNN | GAN | VAE | NICE | PixelCNN | GAN | VAE | NICE |
| Forecasting time horizon is 24 h | $\alpha = 1.1$ | 6.64 | 8.80 | 5.15 | 5.75 | 85.15% | 87.17% | 58.33% | 56.75% |
| | $\alpha = 1.15$ | 10.52 | 12.84 | 9.06 | 9.85 | 94.92% | 96.94% | 88.61% | 85.69% |
| | $\alpha = 1.2$ | 13.86 | 16.33 | 12.50 | 13.11 | 99.37% | 99.58% | 96.25% | 93.50% |
| Forecasting time horizon is 48 h | $\alpha = 1.1$ | 5.84 | 8.69 | 4.88 | 5.46 | 77.60% | 79.65% | 49.43% | 50.13% |
| | $\alpha = 1.15$ | 9.65 | 12.76 | 8.74 | 8.93 | 88.68% | 91.25% | 77.99% | 75.46% |
| | $\alpha = 1.2$ | 13.41 | 16.37 | 12.44 | 12.75 | 96.20% | 97.08% | 90.56% | 88.95% |
| Forecasting time horizon is 72 h | $\alpha = 1.1$ | 5.98 | 8.11 | 4.93 | 5.53 | 66.05% | 68.29% | 44.58% | 44.54% |
| | $\alpha = 1.15$ | 8.75 | 12.24 | 7.20 | 8.16 | 84.45% | 86.11% | 54.54% | 53.87% |
| | $\alpha = 1.2$ | 10.10 | 15.90 | 9.80 | 10.42 | 90.06% | 93.52% | 70.19% | 69.25% |

(3) In practical applications, the choice of parameter $\alpha$ is crucial. In fact, when selecting the $\alpha$, the system operators should fully consider PICP and PINAW, which depend on the dataset, time horizon, point prediction model, and generative model. For example, when the parameter $\alpha$ is 1.15 and the time horizon is 24 h, the PICP of PixelCNN is 94.92%, which can be considered as the probability that the scenarios generated by PixelCNN cover the true value. In other words, the robust optimization solutions obtained based on these generation scenarios have a 94.92% probability to ensure that the limits are not crossed. If the system operators want higher security, they can choose to increase the $\alpha$, but they will also pay an economical price, because the solutions of the robust optimization will be more conservative after parameter $\alpha$ is increased. In general, system operators need to test the PICP and PINAW corresponding to different $\alpha$ values before making a decision. Then, a suitable $\alpha$ value can be selected based on the balance of economy and safety.

## 6. Conclusion

To improve the accuracy of scenario prediction for power loads, a PixelCNN-based method is proposed in this paper. After performing the simulation and analysis on a real-world power load dataset from the University of Texas at Austin, the following conclusions are obtained:

(1) The number of residual blocks in the middle layer, learning rate, and the parameter $k$ in the loss function should not be too large or too small. Normally, 16, 0.001, and 32 are good starting points for the number of residual blocks, learning rate, and the parameter $k$, respectively. Higher values or lower values may be fine for some datasets. Besides, the RMSprop algorithm outperforms other popular optimizers, such as Nadam, Adadelta, Adamax, Adam, Adagrad, and SGD. Unlike other generative networks (e.g., GAN) where the loss functions involve unstable training problems (e.g., non-convergence and mode collapse), the training process of the PixelCNN is very stable, which is one of its strengths.

(2) The PixelCNN can not only accurately capture the temporal dependency of power load curves with different prediction time horizons (e.g., the time horizon range from 0 h to 24 h), but also fit the probability distribution of power load scenarios through unsupervised learning. Further, the PixelCNN can easily generate power load scenarios with different time horizons (e.g., the time horizon range from 24 h to 72 h) by fine-tuning the parameter $m$ in the unconstrained optimization model.

(3) The balance between sharpness and reliability can be controlled by fine-tuning the parameter $\alpha$. when parameter $\alpha$ is small, the predicted power load scenarios are close to deterministic point prediction values, but they may not be able to cover the peak or valley of real power load curves. On the contrary, large parameter $\alpha$ disperses predicted power load scenarios, which are more likely to cover all the real power loads. Compared with other generative networks (e.g., VAE, GAN, and NICE), PixelCNN shows better performance on taking into account both PINAW and PICP. In other words, PixelCNN balances economy and security in practical applications.

(4) The main limitation of PixelCNN is that training time and running time for generating new scenarios are much longer than other generative models (e.g., VAE, GAN, and NICE), but a few hours of training time and a few minutes of running time to generate scenarios are acceptable in real engineering.

In this paper, PIs from previous works are used to validate the proposed method, and future work may extend the proposed method for other PIs constructed by Gaussian, Delta, mean–variance estimation, and bootstrap techniques. Besides, the absence of special days (e.g., holidays) in the feature conditions of the real-world dataset may affect the performance of point prediction, but these missing feature conditions can be easily included in future work.

## CRediT authorship contribution statement

**Wenlong Liao:** Methodology, Roles/Writing – original draft, Conceptualization, Data curation, Formal analysis. **Leijiao Ge:** Writing – review & editing. **Birgitte Bak-Jensen:** Writing – review & editing. **Jayakrishnan Radhakrishna Pillai:** Writing – review & editing. **Zhe Yang:** 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.

## References


Abdelhady, S., Osama, A., Shaban, A., Elbayoumi, M., 2020. A real-time optimization of reactive power for an intelligent system using genetic algorithm. IEEE Access 8, 11991–12000. http://dx.doi.org/10.1109/ACCESS.2020.2965321.

Azeem, A., Ismail, I., Jameel, S.M., Harindran, V.R., 2021. Electrical load forecasting models for different generation modalities: a review. IEEE Access 9, 142239-142263. http://dx.doi.org/10.1109/ACCESS.2021.3120731.

Chen, K., Chen, K., Wang, Q., He, Z., Hu, J., He, J., 2019. Short-term load forecasting with deep residual networks. IEEE Trans. Smart Grid 10 (4), 3943–3952. http://dx.doi.org/10.1109/TSG.2018.2844307.

Chen, Y., Wang, X., Zhang, B., 2018. An unsupervised deep learning approach for scenario forecasts. In: 2018 Power Systems Computation Conference (PSCC). pp. 1–7. http://dx.doi.org/10.23919/PSCC.2018.8442500.

Dordonnat, V., Pichavant, A., Pierrot, A., 2016. GEFCom2014 probabilistic electric load forecasting using time series and semi-parametric regression models. Int. J. Forecast. 32 (3), 1005–1011. http://dx.doi.org/10.1016/j.ijforecast.2015.11.010.

Ge, L., Liao, W., Wang, S., Bak-Jensen, B., Pillai, J.R., 2020. Modeling daily load profiles of distribution network for scenario generation using flow-based generative network. IEEE Access 8, 77587–77597. http://dx.doi.org/10.1109/ACCESS.2020.2989350.

Gm, H., Gourisaria, M.K., Pandey, M., Rautaray, S.S., 2020. A comprehensive survey and analysis of generative models in machine learning. Comp. Sci. Rev. 38, 1–29. http://dx.doi.org/10.1016/j.cosrev.2020.100285.

Guo, P., Ni, X., Chen, X., Ji, X., 2017. Fast PixelCNN: Based on network acceleration cache and partial generation network. In: 2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS). pp. 71–76. http://dx.doi.org/10.1109/ISPACS.2017.8266448.

Hu, S., Zhu, R., Li, G., Song, L., 2021. Scenario forecasting for wind power using flow-based generative networks. Energy Rep. 7, 369–377. http://dx.doi.org/10.1016/j.egyr.2021.08.036.

Jiang, C., Mao, Y., Chai, Y., Yu, M., 2021. Day-ahead renewable scenario forecasts based on generative adversarial networks. Int. J. Energy Res. 45 (5), 7572–7587. http://dx.doi.org/10.1002/er.6340.

Khosravi, A., Nahavandi, S., Creighton, D., 2010. Construction of optimal prediction intervals for load forecasting problems. IEEE Trans. Power Syst. 25 (3), 1496–1503. http://dx.doi.org/10.1109/TPWRS.2010.2042309.

Khosravi, A., Nahavandi, S., Creighton, D., Jaafar, J., 2012. Wind farm power uncertainty quantification using a mean–variance estimation method. In: Proceedings of the 2012 IEEE International Conference on Power System Technology. pp. 1–6. http://dx.doi.org/10.1109/PowerCon.2012.6401280.

Khosravi, A., Nahavandi, S., Srinivasan, D., Khosravi, R., 2015. Constructing optimal prediction intervals by using neural networks and bootstrap method. IEEE Trans. Neural Netw. Learn. Syst. 26 (8), 1810–1815. http://dx.doi.org/10.1109/TNNLS.2014.2354418.

Kong, W., Dong, Z.Y., Jia, Y., Hill, D.J., Xu, Y., Zhang, Y., 2019. Short-term residential load forecasting based on LSTM recurrent neural network. IEEE Trans. Smart Grid 10 (1), 841–851. http://dx.doi.org/10.1109/TSG.2017.2753802.

Li, J., Deng, D., Zhao, J., Cai, D., Hu, W., Zhang, M., et al., 2021. A novel hybrid short-term load forecasting method of smart grid using MLR and LSTM neural network. IEEE Trans. Ind. Inf. 17 (4), 2443–2452. http://dx.doi.org/10.1109/TII.2020.3000184.

Liao, W., Yang, Z., Chen, X., Li, Y., 2021a. WindGMMN: Scenario forecast for wind power using generative moment matching networks. IEEE Trans. Artif. Intell. 1–8. http://dx.doi.org/10.1109/TAI.2021.3128368.

Liao, W., Yang, D., Wang, Y., Ren, X., 2021b. Fault diagnosis of power transformers using graph convolutional network. CSEE J. Power Energy Syst. 7 (2), 241–249. http://dx.doi.org/10.17775/CSEEJPES.2020.04120.

Lin, J., Ma, J., Zhu, J., Cui, Y., 2022. Short-term load forecasting based on LSTM networks considering attention mechanism. Int. J. Electr. Power Energy Syst. 137, 1–10. http://dx.doi.org/10.1016/j.ijepes.2021.107818.

Lindberg, K., Seljom, P., Madsen, H., Fischer, D., Korpas, M., 2019. Long-term electricity load forecasting: current and future trends. Util. Policy 58, 102–119. http://dx.doi.org/10.1016/j.jup.2019.04.001.

McSharry, P.E., Bouwman, S., Bloemhof, G., 2005. Probabilistic forecasts of the magnitude and timing of peak electricity demand. IEEE Trans. Power Syst. 20 (2), 1166–1172. http://dx.doi.org/10.1109/TPWRS.2005.846071.

Nguyen, T.T., Nguyen, N.D., Nahavandi, S., 2020. Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications. IEEE Trans. Cybern. 50 (9), 3826–3839. http://dx.doi.org/10.1109/TCYB.2020.2977374.

Oord, A.V.D., Kalchbrenner, N., Kavukcuoglu, K., 2016a. Pixel recurrent neural networks. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning. pp. 1747–1756, https://dl.acm.org/doi/10.5555/3045390.3045575.

Oord, Avd, Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., Kavukcuoglu, K., 2016b. Conditional image generation with pixelcnn decoders. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. pp. 4797–4805, https://dl.acm.org/doi/10.5555/3157382.3157633.

Pan, Z., Wang, J., Liao, W., Chen, H., Yuan, D., Zhu, W., Fang, X., Zhu, Z., 2019. Data-driven EV load profiles generation using a variational auto-encoder. Energies 12 (5), 1–15. http://dx.doi.org/10.3390/en12050849.

Powell, K.M., Sriprasad, A., Cole, W.J., Edgar, T.F., 2014. Heating, cooling, and electrical load forecasting for a large-scale district energy system. Energy 74, 877–885. http://dx.doi.org/10.1016/j.energy.2014.07.064.

Shepero, M., van der Meer, D., Munkhammar, J., Widén, J., 2018. Residential probabilistic load forecasting: a method using Gaussian process designed for electric load data. Appl. Energy 218, 159–172. http://dx.doi.org/10.1016/j.apenergy.2018.02.165.

Tan, M., Yuan, S., Li, S., Su, Y., Li, H., He, F., 2020. Ultra-short-term industrial power demand forecasting using LSTM based hybrid ensemble learning. IEEE Trans. Power Syst. 35 (4), 2937–2948. http://dx.doi.org/10.1109/TPWRS.2019.2963109.

Turhan, C.G., Bilge, H.S., 2018. Recent trends in deep generative models: a review. In: 3rd International Conference on Computer Science and Engineering (UBMK). pp. 574–579. http://dx.doi.org/10.1109/UBMK.2018.8566353.

Wan, C., Lin, J., Song, Y., Xu, Z., Yang, G., 2017. Probabilistic forecasting of photovoltaic generation: an efficient statistical approach. IEEE Trans. Power Syst. 32 (3), 2471–2472. http://dx.doi.org/10.1109/TPWRS.2016.2608740.

Wang, Y., Chen, Q., Zhang, N., Wang, Y., 2018. Conditional residual modeling for probabilistic load forecasting. IEEE Trans. Power Syst. 33 (6), 7327–7330. http://dx.doi.org/10.1109/TPWRS.2018.2868167.

Wang, Y., Hug, G., Liu, Z., Zhang, N., 2020a. Modeling load forecast uncertainty using generative adversarial networks. Electr. Power Syst. Res. 189, 1–8. http://dx.doi.org/10.1016/j.epsr.2020.106732.

Wang, Z., Wang, W., Liu, C., Wang, B., 2020b. Forecasted scenarios of regional wind farms based on regular vine copulas. J. Mod. Power Syst. Clean Energy 8 (1), 77–85. http://dx.doi.org/10.35833/MPCE.2017.000570.

Wen, H., Gu, J., Ma, J., Yuan, L., Jin, Z., 2021. Probabilistic load forecasting via neural basis expansion model based prediction intervals. IEEE Trans. Smart Grid 12, 3648–3660. http://dx.doi.org/10.1109/TSG.2021.3066567.

Wu, C., Chen, L., Wang, G., Chai, S., Jiang, H., Peng, J., et al., 2020. Spatiotemporal scenario generation of traffic flow based on LSTM-GAN. IEEE Access 8, 186191-186198. http://dx.doi.org/10.1109/ACCESS.2020.3029230.

Xie, J., Hong, T., 2018. Temperature scenario generation for probabilistic load forecasting. IEEE Trans. Smart Grid 9 (3), 1680–1687. http://dx.doi.org/10.1109/TSG.2016.2597178.

Zhang, Y., Liu, K., Liang, Q., An, X., 2016. Deterministic and probabilistic interval prediction for short-term wind power generation based on variational mode decomposition and machine learning methods. Energy Convers. Manage. 112, 208–219. http://dx.doi.org/10.1016/j.enconman.2016.01.023.

Zhang, K., Sun, M., Han, T.X., Yuan, X., Guo, L., Liu, T., 2018. Residual networks of residual networks: multilevel residual networks. IEEE Trans. Circuits Syst. Video Technol. 28 (6), 1303–1314. http://dx.doi.org/10.1109/TCSVT.2017.2654543.

Zhang, L., Zhang, B., 2020. Scenario forecasting of residential load profiles. IEEE J. Sel. Areas Commun. 38 (1), 84–95. http://dx.doi.org/10.1109/JSAC.2019.2951973.

Zhao, Q., Liao, W., Wang, S., Pillai, J.R., 2020a. Robust voltage control considering uncertainties of renewable energies and loads via improved generative adversarial network. J. Mod. Power Syst. Clean Energy 8 (6), 1104–1114. http://dx.doi.org/10.35833/MPCE.2020.000210.

Zhao, C., Wan, C., Song, Y., Cao, Z., 2020b. Optimal nonparametric prediction intervals of electricity load. IEEE Trans. Power Syst. 35, 2467–2470. http://dx.doi.org/10.1109/TPWRS.2020.2965799.

Zhu, R., Liao, W., Wang, Y., 2020. Short-term prediction for wind power based on temporal convolutional network. Energy Rep. 6, 424–429. http://dx.doi.org/10.1016/j.egyr.2020.11.219.