Energy & Buildings 247 (2021) 111152

Contents lists available at ScienceDirect

Energy & Buildings

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

Comparing Generative Adversarial Networks architectures for electricity demand forecasting

Nadjib Mohamed Mehdi Bendaoud [a,*], Nadir Farah [b], Samir Ben Ahmed [a]

[a] Department of Computer Sciences, LIPSIC, University of Tunis El Manar Faculty of Sciences of Tunis, Tunisia
[b] Department of Computer Sciences, LABGED, University of Badji Mokhtar Annaba, Algeria

ARTICLE INFO

Article history:
Received 19 February 2021
Revised 26 May 2021
Accepted 29 May 2021
Available online 01 June 2021

Keywords:
Load forecasting
Short-term load forecasting
Generative Adversarial Networks
Generative model

ABSTRACT

This paper introduces short-term load forecasting (STLF) using Generative Adversarial Networks (GAN). STLF was explored using several Artificial Intelligence based methods that offered excellent results. However, the usage of GAN models in this field is very limited, and usually works on creating synthetic load profiles to increase load datasets. This paper investigates the application of GAN for load forecasting by generating daily load profiles. Predicting the daily load is a challenging task that requires accurate and stable models that can capture seasonality and variation in load data. This paper presents a conditional GAN (cGAN) architecture, that uses only four exogenous variables (maximum and minimum temperature, day of the week and month), to predict a daily profile (24 h of the day). Several types of GAN have been compared such as Deep Convolutional GAN, Least Squares GAN and Wasserstein GAN. The generated load profiles were tested on one year of data and compared to the real load profiles. The proposed GAN models provided excellent predictions, averaging a Mean Absolute Percentage Error (MAPE) of 4.99%.

1. Introduction

Load forecasting generated a large amount of paper especially with the growing interest to smart grids, The ability to anticipate and manage electrical energy implies the development of accurate forecasting systems. Short-term load forecasting (STLF) allows to provide power load predictions for a small future time frame (few minutes to few days ahead), this type of forecasting helps grid system operators to supervise the power needs in relatively small periods and hence, regulate power production. Load forecasting systems were developed using a plethora of techniques. Conventional methods such as Autoregressive moving average (ARIMA) and linear regression [1–3] were explored and showed encouraging results for medium and long term forecasting, and also for wind speed forecasting [4], however, future works [5,6] demonstrated that Artificial Intelligence based (AI-Based) methods showed to be more suitable for the non-linear time series data.

Deep learning (DL) methods are the most used STLF ones and have shown higher accuracy than other techniques. Artificial neural networks (ANN) were largely used in the literature [7,8]. [9,10] proposed a radial basis ANN, [11] introduced an ensemble of Wavelet Neural Networks for STLF, while [12] applied ANN forecasting to micro-grids. [13] used k-nearest neighbors (KNN) to spot similarities between training and testing sets, then a Deep Belief Network (DBN) was built to perform forecasting.

Other types of neural networks have also been explored such as recurrent neural networks. Recurrent neural networks (RNN) had great success with times series data as they capture well the temporal aspect of the task. Applications of RNN such as long short term memory (LSTM) are explored by [14–16] for STLF, [14] compared the results of a standard and a sequence to sequence (S2S) LSTM for a very small term forecasting (one minute).On the other hand, [16] applied LSTM model for residential load prediction, [15] proved that RNN models perform well for commercial and residential building for medium to long term prediction. [17] proposed a multi-layer bidirectional LSTM for feature extraction and a standard LSTM for forecasting future gas consumption, [18] added attention mechanism to LSTM and BI-LSTM model and tested them on several building types. [19] combined the capabilities of LSTM with support vector machines (SVM) and random forests (RF) to improve forecasting capabilities. An adaptive online learning system was proposed by [20], this algorithm is capable of continuous learning and can adapt to changes in the grid infrastructure. Furthermore, other architectures of RNN such as Elman Neural Networks (ENN) and Echo State Network (ESN) were used

* Corresponding author.
E-mail address: bendaoud@labged.net (N.M.M. Bendaoud).

https://doi.org/10.1016/j.enbuild.2021.111152
0378-7788/

for STLF. [21] used multiple forecasting blocks consisting of ENN, while [22] combined ENN with a ridgelet neural network. [23] used Bayesian optimization to optimize the hyper-parameters of a ESN model.

Convolutional neural networks (CNN) [24] is one of the most known DL methods, their convolution layers are very efficient in capturing hidden features. [25] used K-means to cluster data into subsets then fed it to a CNN model to predict hourly load. [26] used the hourly load of the previous seven days as 2d-input to the CNN and aimed to predict the future three days hourly load. [27] presented a three dimensional representation of the data using a cube (past, present and future load) then used a CNN to extract relationships and pass them through a SVR for load forecasting. [28] presented a multi-temporal-spatial-scale temporal convolution network, which transformed the original load data into temporal and special features before using the CNN. [29] used a convolution layer and RNN units on historical load data, and a simple multilayer perceptron (MLP) for the exogenous variables. Other works have combined CNN and Reinforcement Learning (RL) to improve STLF, [30–33] proposed hybrid models with components from both CNN and RL architecture, which helped to obtain better prediction and more robust models for STLF. [34] introduced a probabilistic approach to anticipate future load uncertainties. The system uses multiple quantile forecasts of different techniques such as Gated Recurrent Units (GRU), Gradient Boosting and Random Forests, which are transformed individually into probability density curves. [35] proposed an ensemble learning approach using multiple machine learning algorithms for wind energy forecasting.

From the above literature, the classical approach used for developing forecasting systems is supervised learning. The forecasted time load is subsequently the output of the model. Meanwhile, there are two types of inputs that are usually fed to these models, endogenous inputs, which are previous load values, they are of the same nature of the outputs and are directly related to them. Then there are exogenous inputs, which are variables that are not of the same nature of the outputs (load value), but are correlated to them, the covered literature witnessed a large usage of weather, calendar, and social variables.

This paper explores Generative Adversarial Networks (GAN) for STLF. GAN was initially developed by Goodfellow et al. [36], this technique consists of two neural networks which play a zero-sum game with each other. Given a training set, the first network, which is the generator, generates fake samples; while the second network, the discriminator tries to identify the fake from the real samples. GANs are generative models capable of generating new samples, which have similar characteristics as the training data.

Following the large success of GAN's several works have used it for other domains or improved the model with some significant changes. [37] introduced deep convolutional GAN (DCGAN) that used convolution and transpose convolution layers. [38] presented the conditional GAN (cGAN) which allowed to control the output of the model, by adding another input that represents the condition, to both discriminator and generator. For image data, a large number of applications were explored, [39] offers an outstanding evolution for GAN; the generator was improved by allowing complete control of the latent space by mapping it to an intermediate latent space, this offers the possibility to change output images style and quality. Pix2Pix [40] and Cycle GAN [41] proposes an image to image translation, where unlike a cGAN, the condition here is an image.

Despite being heavily used for image data, GAN's are not widely applied for STLF. Lan et al. [42] used a Wasserstein conditional GAN (wCGAN) for demand side consumption for small and medium enterprises, and the generated consumption profiles matched the real consumption. [43–45] used GAN to generate synthetic (artificial) load data, and used it to expand the real dataset in order to have more sample. A machine learning model is then built to forecast the load, it will be trained on real or synthetic data or with mixed data [45], and tested on real or synthetic data. Combinations of training and testing set are made to further evaluate the synthetic data such as: train on real test on real (TRTR), train on real test on synthetic (TRTS), train on synthetic test on real (TSTR) and train on synthetic test on synthetic (TSTS). Zhang et al. [43] used a Wasserstein GAN and maximum mean discrepancy (MMD) to measure the divergence between real and synthetic data. Fekri et al. [44] proposed a GAN with recurrent neural network units in both generator and discriminator (R-GAN), the paper also proposed the Wasserstein GAN and Metropolis-Hastings GAN (MH-GAN). Tian et al. [45] added a filter after the generator to constraint its output and avoid unrealistic synthetic load profiles from being generated. Then proposed GAN was compared with Information Diffusion Technology (IDT), Heuristic-Mega-Trend-Diffusion (HMDT) and Bootstrap method. Zhou et al. [46] presented a bidirectional GAN (BiGAN) for data augmentation which uses an encoder. The real data are encoded before being passed through the discriminator.

Chen et al. [47] used GAN to generate power profile scenarios of wind and solar power plants, the paper used DCGAN architecture and the output of the generator is a daily profile of wind or solar power plant. A conditional model has also been proposed using weather and calendar variables. Gu et al. [48] proposed a GAN for residential daily load forecasting, using a two-dimensional input, which is made of the seven previous days load data with 30mn granularity, hence the input data was (48*7) matrix. Furthermore, Zhang and Guo [49] proposed an ensemble method using auto-encoders and three GANs, the encoders processed the input features, then each one of the three GAN's generated different scenarios, the output of the model is coupled with a self-organizing map (SOM). Meanwhile, Wang et al. [50] proposed GANs for probabilistic load forecasting, where the GAN is used to assist a forecaster by generating residual scenarios. Yuan et al. [51] used the progressive growing of GANs to generate realistic wind power scenarios.

From the above literature review on the application of GAN for load forecasting, it is obvious that there are two main approaches of using GAN with load data. First, using a simple GAN architecture to generate artificial load profiles, the evaluation is done by comparing the divergence (convergence) of the synthetic and real data. Second, more complex GAN architecture were used to generate synthetic load profiles, those are added to the real dataset to expand it, and then the learning algorithm is applied to the new dataset composed of real and artificial data.

The first approach only provides a scenario generation solution and generates load profiles that lacks in accuracy. The second and the most used approach is the data augmentation, despite being very effective, this approach doesn't show the full capabilities of GANs.

This paper explores the forecasting capabilities of GAN, and aims to show that this generative model could offer accurate prediction, without using other supervised models. This work introduces a direct forecasting approach where the goal is to provide a daily load profile (24 h) forecasting. A conditional GAN architecture is proposed, the historical load data is given as an input vector, while the conditional vector consists of four features; day of the week, month, maximum and minimum temperature. The condition vector adds context to the model and offers the possibility to control the artificial load profile produced by the generator. Four types of GAN were used; Wasserstein GAN, Leas Squares GAN and Deep Convolutional GAN. The results of each model are compared by month and day of week and showed very accurate forecasting.

The main contributions of this paper are:

- Propose a conditional GAN (cGAN) with small condition vector (few exogenous variables) for daily load forecasting.

- Compare different GANs such (Wasserstein GAN, Least-squares GAN and DCGAN)
- Compare the direct GAN prediction approach with simple supervised learning approaches.

Section 2 will give an overview of GANs and their architecture. Section 3 introduces the followed experiment and then Section 4 will present and discuss the obtained results. Finally, Section 5 concludes the paper with an overview of the proposed method and some future perspectives.

2. Generative Adversarial Networks

Generative Adversarial Networks (GAN) is a generative model capable of drawing artificial (fake) samples from a given data distribution. These synthetic samples may be used for creating new instances from a given distribution. They are also used for data argumentation when needing more samples for other tasks. Moreover, GAN's may also be used for supervised learning tasks.

The original GAN proposed by Ian Goodfellow [36] featured a generator and a discriminator; the generator G is trained to generate fake samples that can fool the discriminator D, while the latter is trained to differentiate between real and fake samples. As shown in Fig. 1, the generator uses a noise from the latent space to create fake samples. The discriminator compares real samples with fake samples. Finally the discriminator loss is calculated and used to update both discriminator and generator.

Given a noise from a noise z from the latent space $p(z)$, and a sample x from a real data distribution $p_{data}(x)$. The discriminator output is $D(x)$ for real samples and $D(G(x))$ for fake samples. The generator output is $G(z)$. The discriminator is simultaneously trying to improve his ability to recognize real samples by maximizing $\log D(x)$ to 1, and fake samples by maximizing $\log D(1 - D(G(z)))$ to 0. The generator is avoiding generating samples that are easily recognizable by the discriminator by minimizing$\log(1 - D(G(z)))$.

The min–max game played between generator and discriminator is represented by Eq.1:

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

where $V(D,G)$ is the objective function, $\mathbb{E}_{x\ p_{data}(x)}$ and $\mathbb{E}_{z\ p_z(z)}$ symbolize the expected loss for $logD(x)$ and $log(1 - D(G(z))$ respectively.

The training process of the GAN will first compute the loss and update the discriminator (Eq. (2)). Then compute and update the generator (Eq. (3)).

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))] \quad (2)$$

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} log(1 - D(G(z^{(i)}))) \quad (3)$$

The discriminator is updated by ascending its stochastic gradient ∇_{θ_d}, while the generator is updated by descending its stochastic gradient ∇_{θ_g}. $m$is the batch size used for training, i is the sample index.

2.1. Deep Convolutional GAN

Deep Convolutional GAN (DCGAN) [37] is a special type of GAN that is particularly efficient for images, it uses convolution layers [24] in the discriminator to reduce the dimensionality of the input data. It also uses a transpose convolution to up-sample the input of the generator (Fig. 2).

Fig. 2 shows that the discriminator downsizes the input using convolution layers, and gives a single output (real or fake). The generator up-sample the noise vector from the latent space using transpose-convolution, the output is a real image.

The convolution and transpose convolution operations are explained in [52].

2.2. Conditional GAN

Conditional GAN [38] introduced conditional inputs to train both generator and discriminator. Regular GAN usually generates a random synthetic sample from the input domain, but despite the ability to generate realistic and diverse fake samples, some type of application needs to generate specific samples from the input domain. CGAN uses the label of each image from the MNIST dataset as a condition to train the model. The generator is now able to produce an image of the desired number by simply feeding the one-hot encoded number together with noise from the latent space.

As shown in Fig. 3, a condition vector is used in both generator and discriminator. In the discriminator the condition vector is concatenated with the input image. In the generator the condition vector is concatenated with the noise from the latent space.

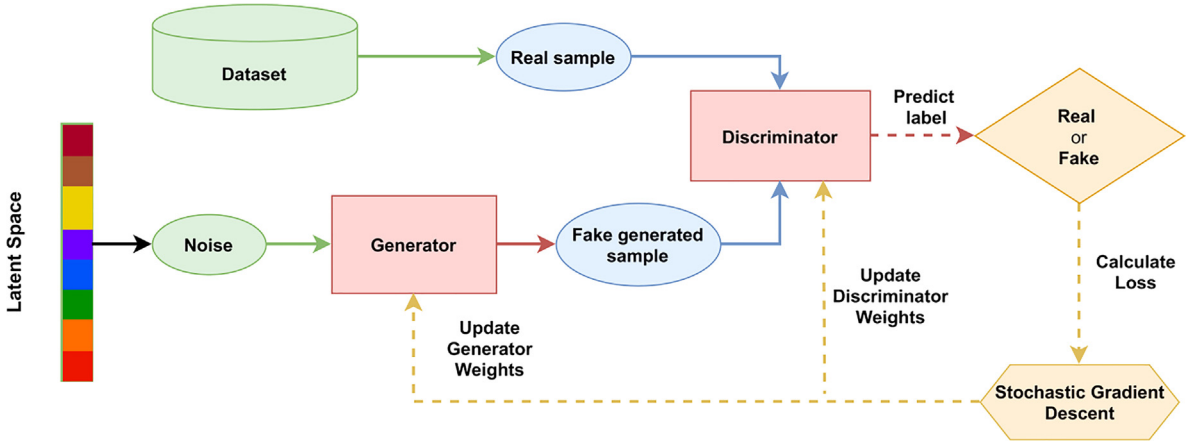

Fig. 1. Generative Adversarial Network architecture.

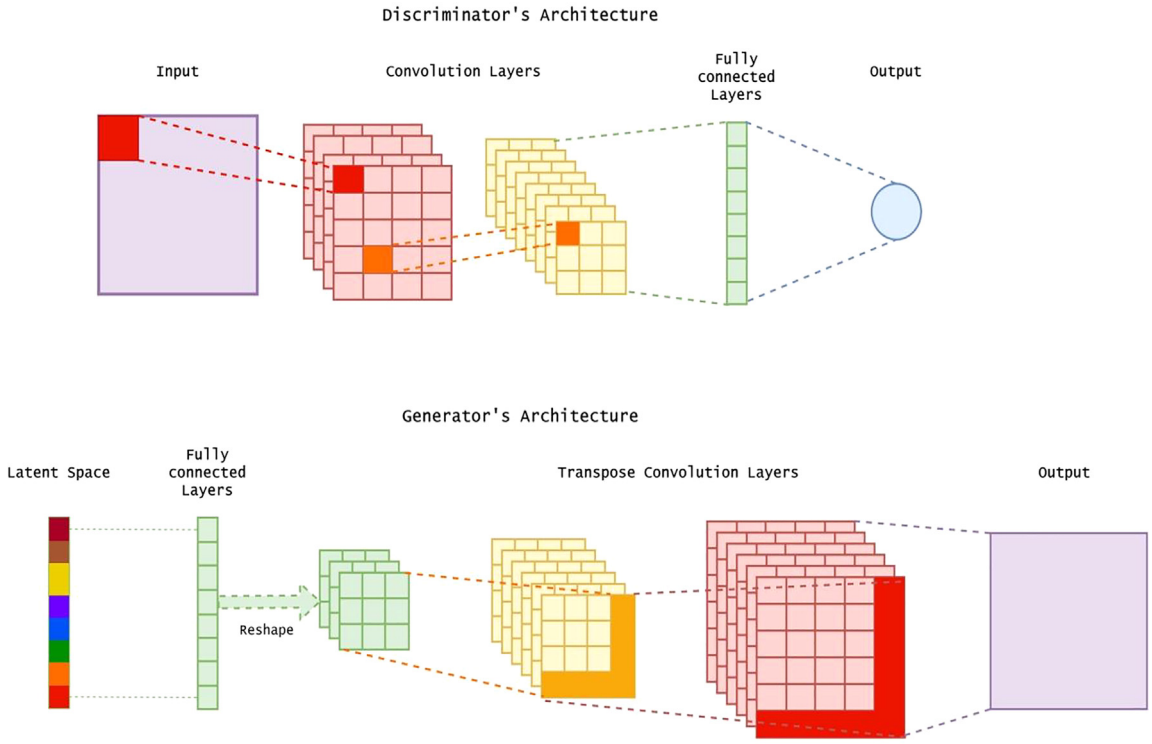

Fig. 2. Deep Convolutional GAN architecture.

2.3. Wasserstein GAN

The Wasserstein GAN (WGAN) was first introduced in [53], it provides an alternative for training GANs. The difference here is that the discriminator (called critic) will measure the quality of a given sample (the degree of fakeness or realness), instead of predicting the probability of the sample of being real or fake.

WGAN uses the Wasserstein distance which is also known as the Earth-Mover (EM) distance. EM distance can be explained intuitively as how much mass (matter) is moved from a distribution P_x to another distributionG_y.

EM distance is given as by the following equation (Eq. (4)):

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\ \gamma}[\|x - y\|] \tag{4}$$

where $W(P_r, P_g)$ is the earth mover or (Wasserstein distance), $\Pi(P_r, P_g)$ is the set of all disjoint distributions $\gamma(x, y)$ whose marginals are respectively P_r andP_g. The EM distance is the optimal cost needed for transporting$\gamma(x, y)$. $\gamma(x, y)$ is the amount of mass (matter) transported from x to y needed to transform the distribution P_r into P_g [53].

In order to use the EM distance with GAN, the Kantorovich-Rubinstein duality is applied to (Eq. (4)) giving the Eq. (5).

$$W(P_r, P_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x\ p_r}[f(x)] - \mathbb{E}_{x\ p_g}[f(x)] \tag{5}$$

Since the Kantorovich-Rubinstein is a duality of the original equation, the supremum *sup* is used instead of *inf*. The function f here should be K-Lipschitz continuous, and it should satisfy $||f||_L \leq K$, where K is the Lipschitz constant. Hence, given a function f from a parameterized family of K-Lipschitz continuous functions $\{f_w\}_{w \in W}$, the loss function for the Wasserstein GAN is given by Eq. (6):

$$\max_{w \in W} \mathbb{E}_{x\ p_r}[f_w(x)] - \mathbb{E}_{z\ p_r(z)}[f_w(g_\theta(z))] \tag{6}$$

where $g_\theta(z)$ are the generated data.

According to the original paper [35], WGAN training is less sensitive to hyper-parameters tuning and architectural changes, besides having a more stable training process. The main differences with the original GAN are:

- The critic (discriminator) is updated more times than the generator.
- Weight clipping is used to enforce K-Lipschitz continuity, the weights are constrained into a small interval [-c, c], where c is the clipping constraint.
- In the critic, the labels are equal to −1 for real samples and 1 for fake samples.

WGAN may show some unwanted behavior such as training instability or slow convergence, which is mainly due to weight clipping. This issue has been addressed by [54] which introduced gradient penalty as an alternative for weight clipping also called WGAN-GP.

A differentiable function is considered 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere [54], hence, WGAN-GP will use an interpolated data point between real and generated samples, and penalize the model if its gradient norm is not equal to 1, the objective function is given in Eq. (7).

$$L = \mathbb{E}_{x\ p_g}[D(x)] - \mathbb{E}_{x\ p_r}[D(x)] + \lambda \mathbb{E}_{\widehat{x}\ p_{\widehat{x}}}\left[\left(\|\nabla_{\widehat{x}} D(\widehat{x})\|_2 - 1\right)^2\right] \tag{7}$$

where $D(x)$ and $D(x)$ represents the discriminator output for respectively the artificial and real data. $\left(\|\nabla_{\widehat{x}} D(\widehat{x})\|_2 - 1\right)^2$ is the gradient penalty.

$\widehat{x}$ Is the interpolated data sampled from artificial data x and real data x (Eq. (8)):

$$\widehat{x} = t\,x + (1 - t)x | 0 \leq t \leq 1 \tag{8}$$

λ is the penalty constant.

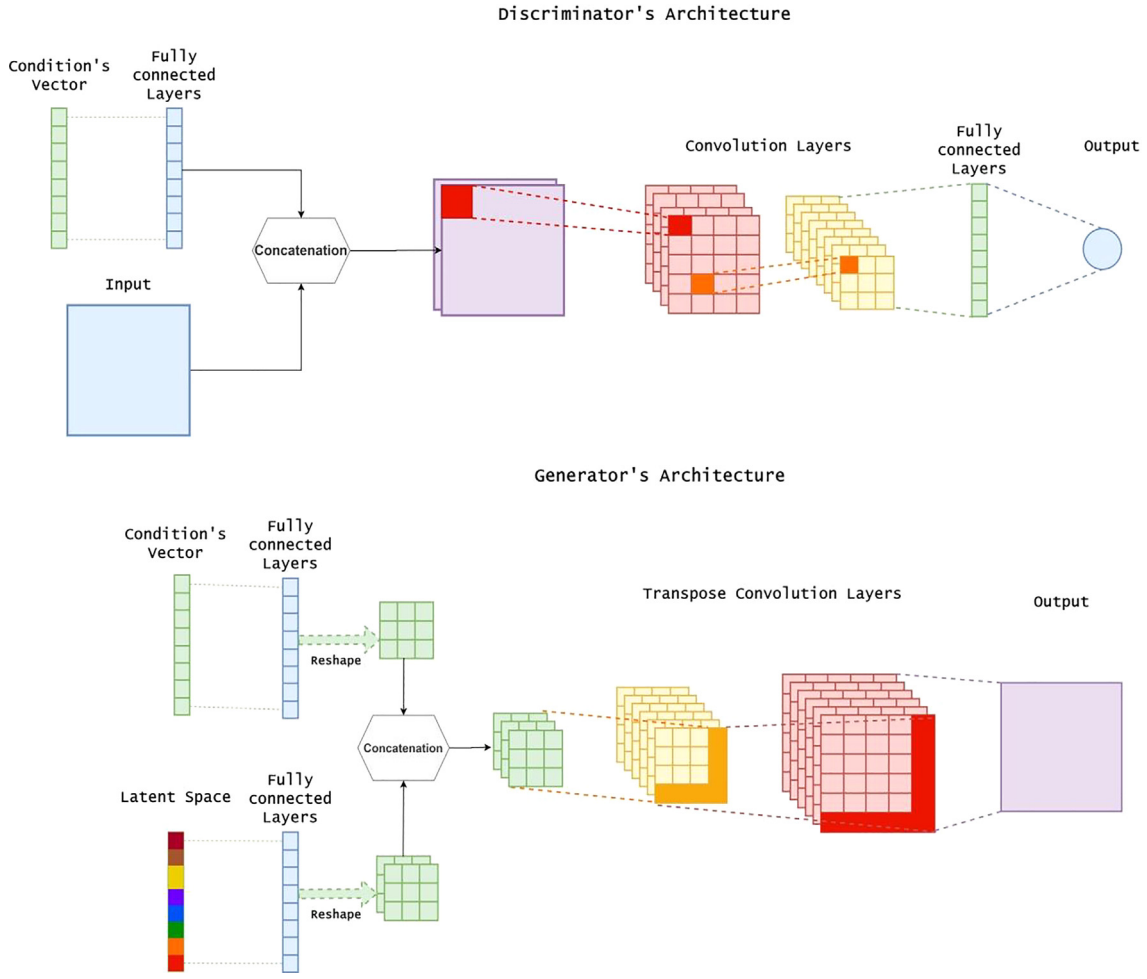

Fig. 3. Conditional GAN architecture.

2.4. Least-Squares GAN

Least-Squares GAN (LSGAN) [55] is another form of GAN that uses the L2 loss (Least Squares) as loss function for the model (Eq. (9) and Eq. (10)).

$$\min_D V_{LSGAN}(D) = \frac{1}{2}\mathbb{E}_{x\ p_{data}(x)}\left[(D(x)-b)^2\right] + \frac{1}{2}\mathbb{E}_{z\ p_z(z)}\left[(D(G(z))-b)^2\right] \quad (9)$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2}\mathbb{E}_{z\ p_z(z)}\left[(D(G(z))-c)^2\right] \quad (10)$$

where $D(x)$ and $D(G(z))$ are respectively the discriminator output for real samples and generated samples, $G(z)$ is the generator output. The constant values a, b and c symbolize the way real and fake data are indentified. Typically, real data will be given a label of 1 ($b = 1$), generated (fake) data will have a label of 0 ($a = 0$). The generator aims to fool the discriminator, it presents fake data as being real, hence $c = 1$.

3. Experiment

3.1. Data preprocessing

The Algerian National Agency for Electricity and Gas (SONELGAZ) provided the data which was used for this experiment. It consists of three years of hourly load data (2017–2019). In addition to the power load, temperature data for the same time frame are also used for this experiment.

Both load and temperature data are normalized using Eq. (11).

$$V_{normalized} = \frac{V_{real} - V_{\min}}{V_{\max} - V_{min}} \quad (11)$$

where $V_{normalized}$ is the normalized value, V_{real} is the original value, and $V_{\min}$ and $V_{\max}$ are respectively the maximum and minimum values of the dataset. The values here are load or temperature data.

3.2. Daily load forecasting

The purpose here is to build a model able to forecast a daily profile, i.e. the 24 h of a day. A conditional GAN (cGAN) architecture is proposed to perform this task. A cGAN as explained in Section 2.2 is a special architecture of GAN which allows to constraint the input of the model by adding conditions, hence the proposed cGAN will then have two inputs; first, a daily profile and a condition.

Typically, forecasting models provides previous load information in addition to calendar or weather data as input features as it produces better models. However, the proposed cGAN uses only month, day of the week and daily maximum and minimum temperatures as a condition vector. The first two conditions are calendar variables, which will help to temporally locate the forecasted daily profiles. Whereas, the daily maximum and minimum temperatures will help to measure the amplitude of the daily profile in

order to generate more accurate forecasting. Table 1 shows the components of the condition vector used for the cGAN.

The cGAN model architecture will be the following; A daily load profile in addition to the corresponding condition vector are fed to the discriminator. The generator uses a condition vector with random values together with a noise from latent space to generate an artificial daily load profile. This will be passed to the discriminator as well as the random condition vector used to generate the fake daily profile. Fig. 4 shows an example where, the day used is a Monday in the month of April, the maximum and minimum temperature of the day are respectively 27 and 12.

For each sample in the dataset, the discriminator is fed with a daily load profile i.e. a vector containing 24 h load, a condition vector containing the month, day of the week and the minimum and maximum temperatures corresponding to the selected daily profile.

On the other hand, the generator is fed with a noise from the latent space i.e. a random Gaussian distribution and, a vector with random values representing the conditions. The temperature varies according to the season, hence the random values of both max and min temperatures depend on the season, Table 2 shows the intervals used for the random values of min and max temperatures.

Table 1
List of variables used with the condition vector.

Conditions			
D_t	M_t	$T_t^{\max}$	$T_t^{\min}$
Day of the week	Month	Maximum temperature	Minimum Temperature

3.3. Building the cGAN model

The proposed architecture is tested using three types of GAN, a DCGAN, LSGAN and WGAN. Several hyper-parameters were put to test for all the types of GAN and only the architectures with the best training results were selected.

For all the models, the discriminator takes a load profile in the form of a one-dimensional tensor of shape$(24, 1)$, and a condition vector of size 4. The generator has two inputs, a noise from the latent space, and a condition vector of size 4.

3.3.1. Conditional DCGAN (cDCGAN)

In the discriminator, the condition vector is passed through two dense layers then reshaped to be concatenated with the load profile input. Two 1-d convolution layers are used with a stride of 2. Then, the discriminator output is a dense layer of size 1.

In the generator, the noise and vector inputs are concatenated after being passed through a dense layer then reshaped. After the concatenation, two 1-d convolution layers with a stride of 2 and a dense layer are used, and finally, the output is a one-dimensional load profile tensor of shape$(24, 1)$ (Table 3).

The number of epochs used in cDCGAN is 20.000 with a batch size of 128. The generator used ReLu activation for all the layers, except the output which uses *tanh* activation. The discriminator used Leaky ReLu activation with $(\alpha = 0.2)$ for all the layers, while the output takes a sigmoid activation.

The loss function for is Binary Cross-Entropy, the training is optimized using Adam [56].

3.3.2. Conditional LSGAN (cLSGAN)

The architecture of the cLSGAN is similar to cDCGAN except for the size of the convolution filters (kernels) for the discriminator

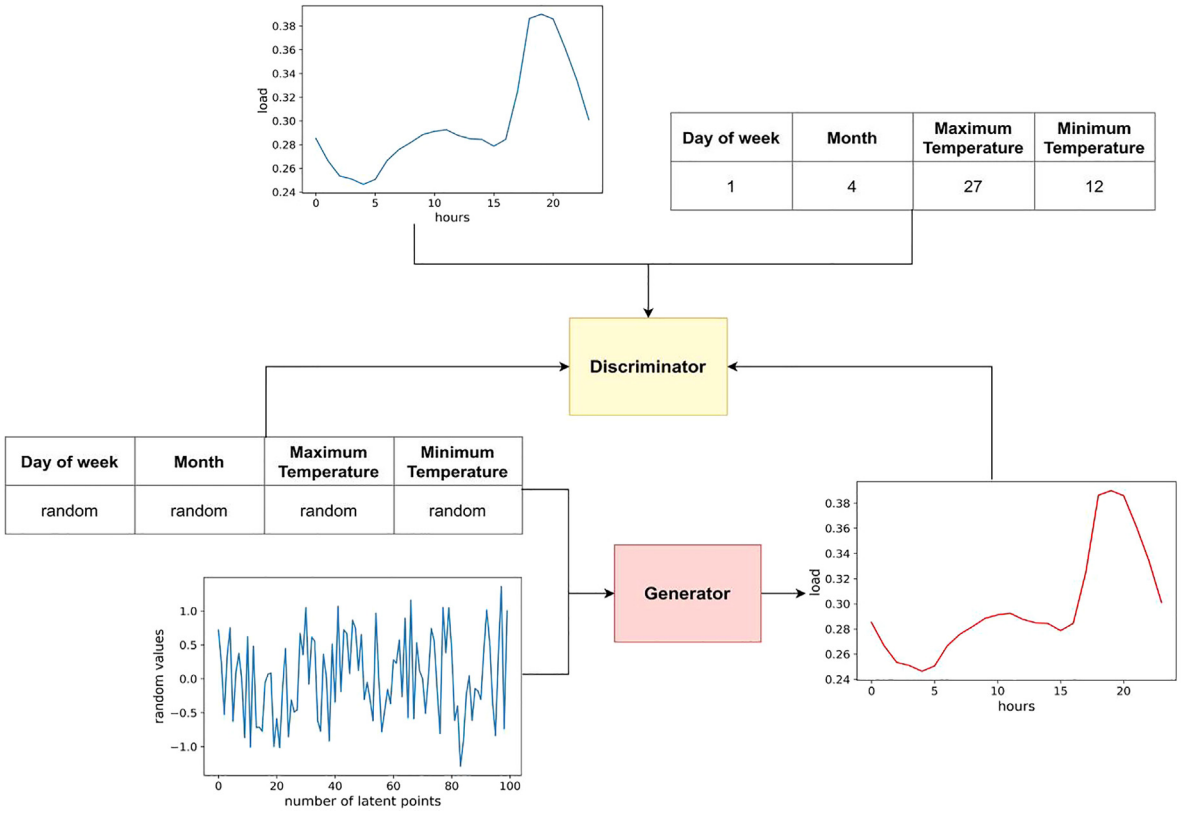

Fig. 4. The proposed GAN architecture.

(size = 2) and their number for both discriminator and generator (conv1d-layer 1 = 32, conv1d-layer 2 = 64) (Table 4).

The number of epochs used in cLSGAN is 9.000 with a batch size of 128. The generator used ReLu activation for all the layers, except the output which uses *tanh* activation. The discriminator also used ReLu activation, whereas the output here has a linear activation.

The training is optimized using adam optimizer, however the loss function here is the mean squares error (MSE).

3.3.3. Conditional WGAN (cWGAN)

The training process for cWGAN showed to be very difficult, this behavior has been highlighted by the literature as the weight clipping procedure is not very reliable. The evaluation accuracy of the model is very low and the training is fairly unstable, even after considering different architectures and tuning the hyper-parameters such as learning rate, and weight clipping (table 5).

The clipping constraint has been set to 0.02, and the kernel initialization for both discriminator and generator is 0.05. The model uses a batch size of 64, and has been trained for 800 epochs and optimized with RMSProp (Root Mean Square Propagation), the loss function is the Wasserstein loss.

3.3.4. Conditional WGAN-GP (cWGAN-GP)

The latent space input size here is set to 128 while the sizes and number of filters of the convolution layers in both discriminator and generator have been increased (Table 6).

The batch size used for the cWGAN-GP is 128, with 800 epochs, the activation functions are the same for the previous models. The loss function here is the Wasserstein loss, while the training is optimized with Adam. The gradient penalty parameter is set to 0.1.

Table 2
Maximum and Minimum temperature interval for each season, used for the random condition vector.

Season	Autumn	Winter	Spring	Summer
Minimum Temperature Interval	$[-2, 11]$	$[-5, 8]$	$[2, 13]$	$[10, 19]$
Maximum Temperature Interval	$[19, 30]$	$[10, 21]$	$[19, 32]$	$[23, 46]$

Table 3
Architecture of the proposed cDCGAN model.

Discriminator		Generator	
Load Profile **Input** (24,1)	Condition **Input** (4)	Latent Space **Input** (100)	Condition **Input** (4)
	Dense (50)	Dense (300)	Dense (10)
	Dense (24)	Reshape (10,30)	Reshape (10,1)
	Reshape (24,1)	Concatenate (10,31)	
Concatenate (24,2)		Conv1d (kernel = 64, size = 2)	
Conv1d (kernel = 64, size = 3)		Conv1d (kernel = 128, size = 2)	
Conv1d (kernel = 128, size = 3)		Dense(24)	
Output – Dense (1)		**Output** – Reshape(24,1)	

Table 4
Architeture of the proposed cLSGAN model.

Discriminator		Generator	
Load Profile **Input** (24,1)	Condition **Input** (4)	Latent Space **Input** (100)	Condition **Input** (4)
	Dense (50)	Dense (300)	Dense (10)
	Dense (24)	Reshape (10,30)	Reshape (10,1)
	Reshape (24,1)	Concatenate (10,31)	
Concatenate (24,2)		Conv1d (kernel = 32, size = 2)	
Conv1d (kernel = 32, size = 2)		Conv1d (kernel = 64, size = 2)	
Conv1d (kernel = 64, size = 2)		Dense (24)	
Output – Dense (1)		**Output** – Reshape (24,1)	

4. Results and discussion

The conducted experiments in the previous section are evaluated using the mean absolute forecasting error (MAPE) (Eq. (12)) and root squares error (RMSE) (Eq. (13)).

$$MAPE = \frac{100\%}{N}\sum_{i=0}^{N}\left|\frac{R_i - F_i}{R_i}\right| \tag{12}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(R_i - F_i)^2} \tag{13}$$

where N is the number of samples, R is the real value, and F is the forecasted value.

Hold-out cross validation was used, for the training and the evaluation of the proposed models. The data was split chronologically between training and testing. Two years of data are used for training and one year is held out for testing, as shown in Table 7.

4.1. Baseline model

A vanilla Artificial Neural Network (ANN) was built to serve as a baseline model for comparison with the proposed cGAN models. The input vector for the ANN is the same as the condition vector

Table 5
Architecture of the proposed cWGAN model.

Discriminator		Generator	
Load Profile Input (24,1)	Condition Input (4)	Latent Space Input (100)	Condition Input (4)
	Dense(50)	Dense (300)	Dense (10)
	Dense(24)	Reshape (10,30)	Reshape (10,1)
	Reshape (24,1)	Concatenate (10,31)	
Concatenate (24,2)		Conv1d (kernel = 32, size = 2)	
Conv1d (kernel = 64, size = 2)		Conv1d (kernel = 64, size = 2)	
Conv1d (kernel = 128, size = 2)		Dense(24)	
Output – Dense (1)		Output – Reshape(24,1)	

Table 6
Architecture of the proposed cWGAN-GP model.

Discriminator		Generator	
Load Profile **Input** (24,1)	Condition **Input** (4)	Latent Space **Input** (128)	Condition **Input** (4)
	Dense(50)	Dense (800)	Dense (10)
	Dense(24)	Reshape (10,80)	Reshape (10,1)
	Reshape (24,1)	Concatenate (10,81)	
Concatenate (24,2)		Conv1d (kernel = 64, size = 3)	
Conv1d (kernel = 128, size = 3)		Conv1d (kernel = 128, size = 3)	
Conv1d (kernel = 256, size = 2)		Dense(24)	
Output – Dense (1)		**Output** – Reshape(24,1)	

Table 7
Training and test sets description.

Load Data	Start	End	Number of Samples
Training Set	01-01-2017	12-31-2018	17,520
Test Set	01-01-2019	12-31-2019	8760

Table 8
Overall MAPE and RMSE accuracy for the GAN models.

Models	cDCGAN	cLSGAN	cWGAN	cWGAN-GP	ANN
MAPE (%)	4.99	5.44	13.04	5.07	5.54
RMSE (MW)	557.31	610.89	1844.03	598.77	609.15

Table 9
MAPE and RMSE results for the generated profiles by the proposed cGAN models for each month.

	MAPE (%)				RMSE (MW)			
Month	cDCGAN	cLSGAN	cWGAN-GP	ANN	cDCGAN	cLSGAN	cWGAN-GP	ANN
January	3.33	4.29	4.21	3.73	356.7	427.42	433.61	366.66
February	3.34	4.09	3.61	4.4	344.47	399.44	382.28	421.42
March	3.88	5.25	3.43	4.8	365.89	491.97	322.12	446.85
April	4.87	5.51	3.49	6.05	409.96	454.35	307.22	506.04
May	6.64	6.67	5.13	6.66	590.32	601.8	504.35	589.35
June	8.43	9.0	7.74	8.37	953.58	968.33	873.15	940.4
July	4.15	5.56	4.51	5.57	605.33	780.05	621.05	733.97
August	4.69	5.43	6.65	5.72	675.07	779.2	952.91	791.11
September	6.08	7.0	8.02	6.95	698.98	825.94	900.91	825.6
October	4.63	4.29	4.4	4.06	429.39	412.42	440.79	396.24
November	5.23	4.91	4.21	6.15	482.37	456.66	411.27	555.31
December	4.56	3.24	5.37	4.07	439.49	328.79	526.25	379.39

Table 10
MAPE and RMSE results for the generated daily profiles by the proposed cGAN models for weekdays.

	MAPE (%)				RMSE (MW)			
Month	cDCGAN	cLSGAN	cWGAN-GP	ANN	cDCGAN	cLSGAN	cWGAN-GP	ANN
Sunday	5.24	6.4	5.06	6.34	487.19	741.03	603.61	706.26
Monday	4.38	4.75	4.33	4.91	529.65	548.64	550.66	539.29
Tuesday	4.77	5.09	4.86	5.38	585.4	579.47	613.38	576.17
Wednesday	4.91	5.33	5.41	6.01	540.33	664.05	736.24	700.89
Thursday	4.89	5.88	5.14	5.96	517.11	611.54	543.47	620.46
Friday	5.12	5.11	5.47	4.73	604.53	528.84	596.34	515.03
Saturday	5.61	5.5	5.2	5.45	624.19	577.58	632.86	580.05

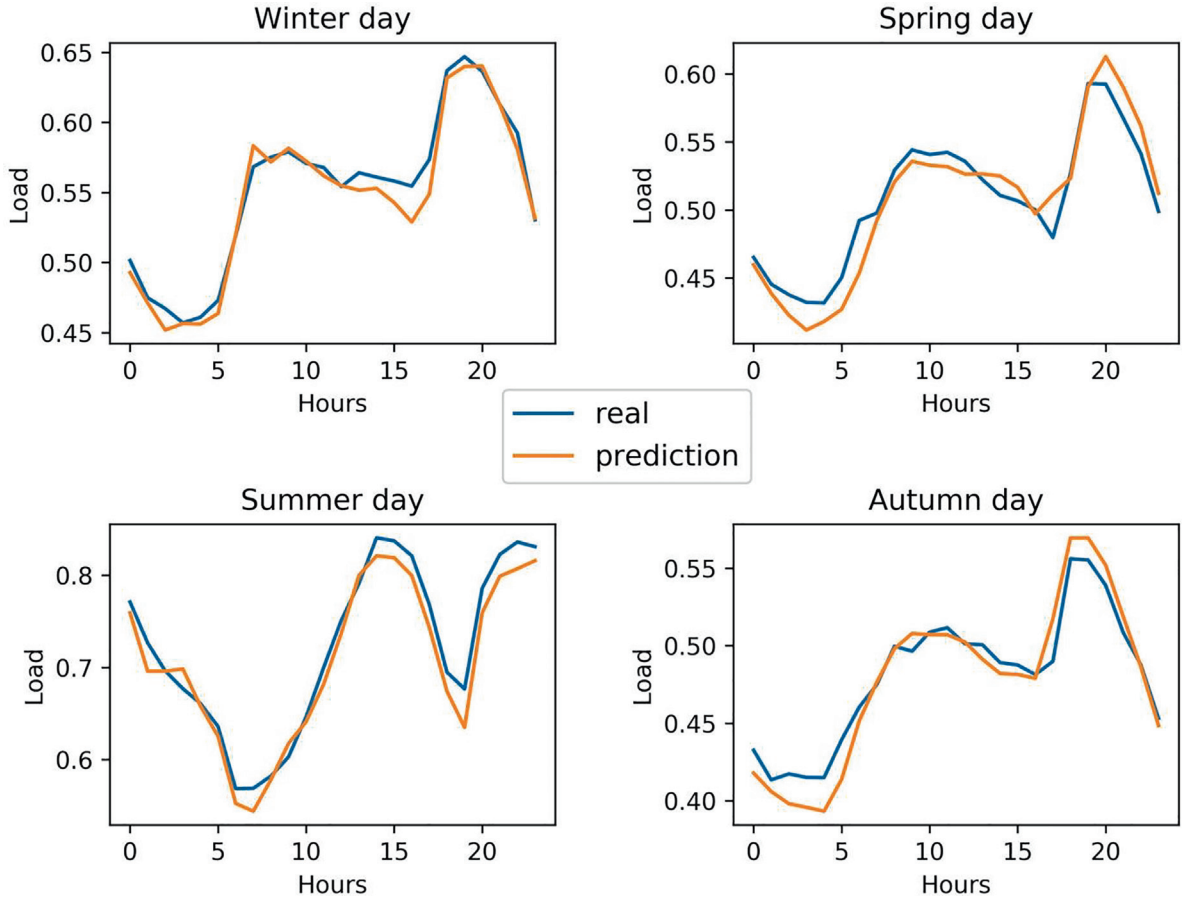

Fig. 5. cDCGAN load prediction of a random day of each season.

of the cGAN, which contains only four features: day of the week, month, minimum and maximum temperatures.

4.2. cGAN models evaluation

Table 8 shows the MAPE and RMSE accuracy for the proposed models

The cWGAN model test results are not good (MAPE = 13.04, RMSE = 1844.03), however the gradient penalty version of cWGAN-GP obtained excellent results (MAPE = 5.07%, RMSE = 59 8.77 MW) which is better than cLSGAN (MAPE = 5.44%, RMSE = 6 10.89 MW), the best model is the cDCGAN (MAPE = 4.99%, RMSE = 557.31 MW). All the proposed models showed to be better than the baseline vanilla ANN.

Tables 9 and 10 shows respectively, the monthly and daily prediction evaluation using MAPE and RMSE.. The cWGAN model is not included in the tables.

Table 9 displays the monthly MAPE and RMSE of the models. The first four months of the year show very good results especially for cDCGAN and cWGAN-GP which show an MAPE in the range [3.33% – 4.87%] and RMSE in the range [356.7 MW – 409.96 MW]. However, we can clearly see that all the models are less accurate in the beginning of summer and autumn (June and September), this translates by high RMSE values reaching 953.58 MW and 900.91 MW for cDCGAN and cWGAN-GP respectively. The struggle of the models in the beginning of summer and autumn may be explained by the variation of the weather in Algeria for these periods of the year. The metrics show excellent for the remaining months for every model especially the cDCGAN which offered the best MAPE [4.15% – 5.23%].

The weekdays' results are excellent, working days offer the best accuracies with MAPE in range [4.33% – 5.88%], cWGAN-GP scored the better results for Sundays and Mondays, whereas cDCGAN performed better for Tuesdays, Wednesdays and Thursdays. The RMSE results for weekdays are all in favor of cDCGAN which means that there is less larger error in the model's prediction, except for Tuesday where ANN scored better. Week-ends (Friday and Sunday for Algeria) show slightly different results, despite cWGAN-GP having the best MAPE for Saturdays, Fridays show that ANN scored the best result (4.73%). Same for RMSE, ANN shows the best score for Fridays, while cLSGAN obtained the best RMSE for Saturdays. The overall values of the metrics are excellent especially for GAN models.

The obtained results in Tables 9 and 10 show the impact of months and weekdays in the prevision process. The GAN models have been able to adapt to the variations of the load curve and learned to anticipate the behavior of the system. The fact of adding month and weekdays in the condition vector of the GAN, have shown to be essential to reinforce the models' behavior on particular month and days of the week, and to adapt to the variations in power demand.

Figs. 5 and 6, show the prediction of the cDCGAN and cWGAN-GP respectively, for a random day for each season.

The cDCGAN is the best performing and it is clearly illustrated Fig. 5, the prediction curve approximates well the real curve.

Fig. 6 shows that cWGAN-GP is less accurate than the cDCGAN, the prediction overestimates certain peaks in summer and autumns, however it offers an overall good approximation of the load curve.

A comparison of the cDCGAN and cWGAN-GP is given in Fig. 7. In order to display the prediction of all the test set, the daily load average has been used.

Fig. 8 shows the autocorrelation between real load profiles and predicted load profiles by cDCGAN model. The generated profiles

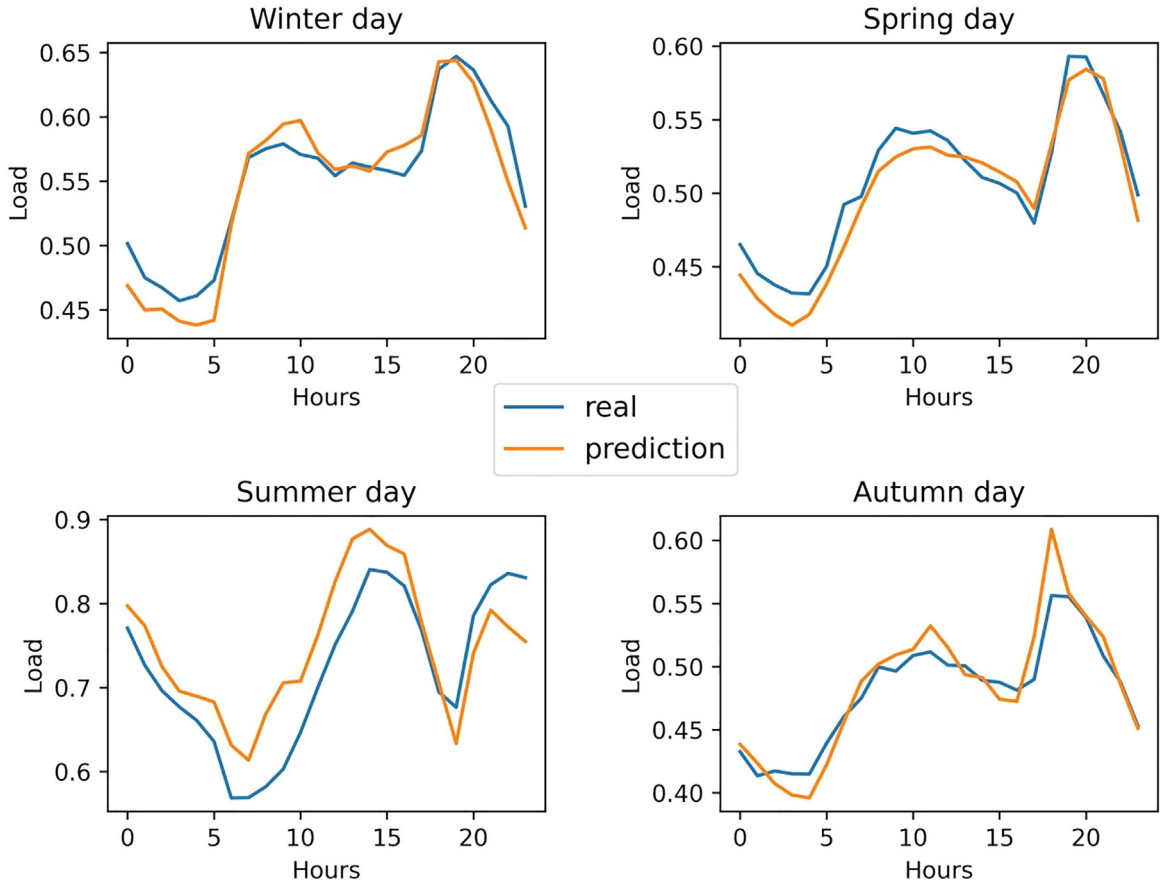

Fig. 6. cWGAN-GP load prediction of a random day of each season.

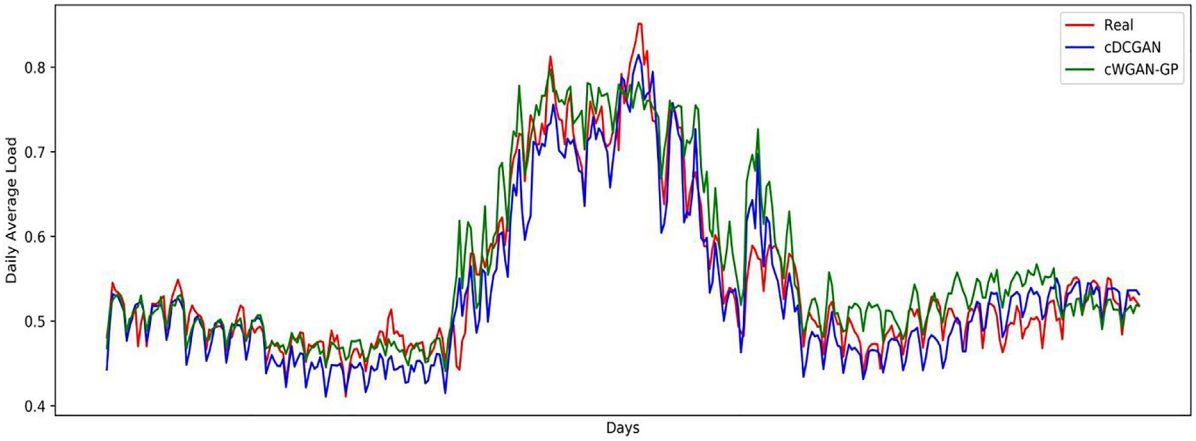

Fig. 7. Comparison of cDCGAN and cWGAN-GP with real data for average daily prediction.

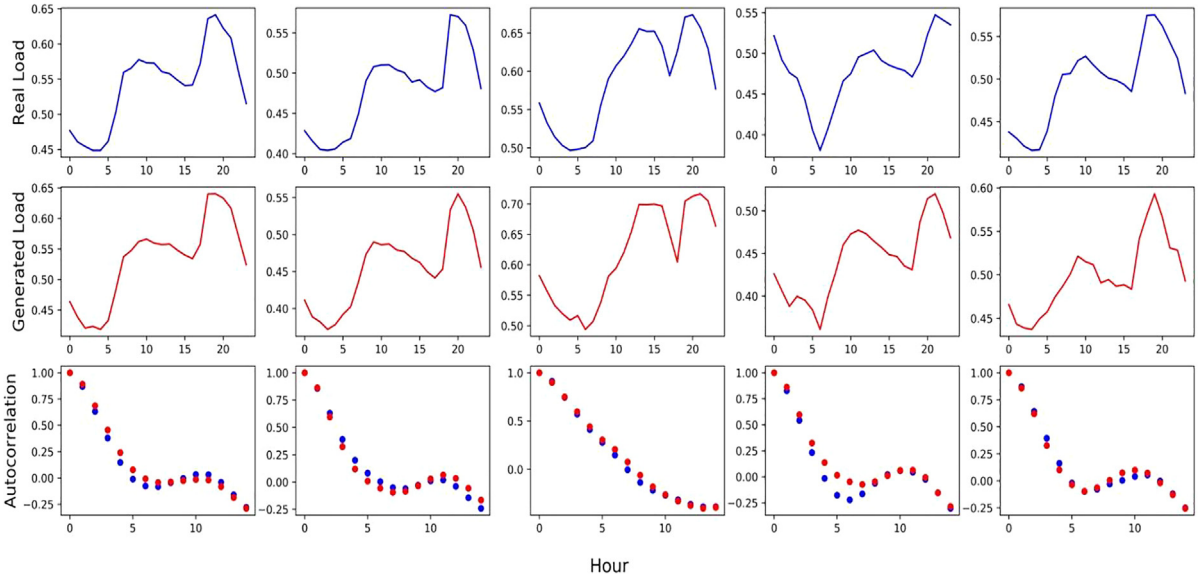

Fig. 8. The results of daily profiles generated by cDCGAN. The first row represents real daily profiles. The second row represents the generated load profiles for the same day. The third row displays the autocorrelation of both real data and generated data.

curves looks very similar to real ones, it manage to successfully replicate the daily load curve and also, easily reaching daily peaks. The autocorrelation of both real and generated profiles are very similar, this means that they evolve similarly through time.

Low MAPE and RMSE results highlight the accuracy and effectiveness of the generated load profiles across the predicted year. The GAN models seem to have successfully captured seasonality and variation in the load data and can provide stable predictions throughout the year. Facing the challenging growth in power demand at summertime for the Algerian market, the model managed to produce excellent predictions despite showing lower accuracy than the other months (Fig. 7). This phenomenon is much more related to the quick increase on power demand from one year to another, which makes it harder to the model to learn this pattern.

5. Conclusion

Forecasting power demand is essential to the control and management of the electricity grid; it allows anticipating future power demands, and it represents a step further towards the development of more intelligent and environment friendly grids. This paper introduced GAN for short-term load forecasting; STLF witnessed several breakthrough papers, which featured numerous AI-based techniques and provided accurate and stable STLF models. Despite being popular in the DL world especially for image data, GANs were not well explored in the load-forecasting field and were used only as a data augmentation tool. This paper uses GAN to predict daily power load (24 h). After a general explanation of the GAN's, a conditional GAN (cGAN) approach was investigated for this task. A conditional vector containing four exogenous variables (maximum and minimum temperature, day of the week, and month) is used to train the model. Several types of GAN has been used which are DCGAN, LSGAN, WGAN and WGAN-GP. The conditional architecture has been trained with each type of GAN aforementioned and then, tested on one year of unseen data. The results showed that cDCGAN, cLSGAN and cWGAN-GP obtained excellent results, offering very accurate prediction throughout the test set with cDCGAN averaging 4.99% MAPE. GANs showed to be very effective for load forecasting tasks and hence, several research perspectives could be explored in the future. The conditional architecture could be enhanced by manipulating the latent

space as done in style-based GAN [36]. It could be very interesting to find a correlation between generated load profiles and the latent space which will help to improve the quality of the generated profiles. Furthermore, one way of enhancing the proposed work could come by using multiple condition vectors. Each vector may represent a one-hot encoded feature, or every vector could contain different variable types (for climatic variables or calendar variables ...etc). Finally, another future contribution could be the use of a model that controls the quality of the generated load profiles, this model could be trained beforehand.

Declaration of Competing Interest

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

Acknowledgment

We are grateful to SONELGAZ (Algerian National Company of Electricity and Gas) for providing electricity data for this project.

References

[1] V.Ş. Ediger, S. Akar, ARIMA forecasting of primary energy demand by fuel in Turkey, Energy Policy 35 (3) (2007) 1701–1708, https://doi.org/10.1016/j.enpol.2006.05.009.
[2] S.S. Pappas, L. Ekonomou, P. Karampelas, D.C. Karamousantas, S.K. Katsikas, G. E. Chatzarakis, P.D. Skafidas, Electricity demand load forecasting of the Hellenic power system using an ARMA model, Electr. Power Syst. Res. 80 (3) (2010) 256–264, https://doi.org/10.1016/j.epsr.2009.09.006.
[3] M.S. AL-Musaylh, R.C. Deo, J.F. Adamowski, Y. Li, Short-term electricity demand forecasting using machine learning methods enriched with ground-based climate and ECMWF Reanalysis atmospheric predictors in southeast Queensland, Australia, Renew. Sustain. Energy Rev. 113 (2019) 109293, https://doi.org/10.1016/j.rser.2019.109293.
[4] S.R. Moreno, V.C. Mariani, L.dos.S. Coelho, Hybrid multi-stage decomposition with parametric model applied to wind speed forecasting in Brazilian Northeast, Renew. Energy. 164 (2021) 1508–1526, https://doi.org/10.1016/j.renene.2020.10.126.
[5] N. Wei, C. Li, X. Peng, F. Zeng, X. Lu, Conventional models and artificial intelligence-based models for energy consumption forecasting: a review, J. Pet. Sci. Eng. 181 (2019) 106187, https://doi.org/10.1016/j.petrol.2019.106187.
[6] M. Cai, M. Pipattanasomporn, S. Rahman, Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques, Appl. Energy. 236 (2019) 1078–1088, https://doi.org/10.1016/j.apenergy.2018.12.042.
[7] A. Baliyan, K. Gaurav, S. Kumar Mishra, A review of short term load forecasting using artificial neural network models, Procedia Comput. Sci. 48 (2015) 121–125, https://doi.org/10.1016/j.procs.2015.04.160.
[8] A.S. Khwaja, X. Zhang, A. Anpalagan, B. Venkatesh, Boosted neural networks for improved short-term electric load forecasting, Electr. Power Syst. Res. 143 (2017) 431–437, https://doi.org/10.1016/j.epsr.2016.10.067.
[9] C. Cecati, J. Kolbusz, P. Rozycki, P. Siano, B.M. Wilamowski, A novel RBF training algorithm for short-term electric load forecasting and comparative studies, IEEE Trans. Ind. Electron. 62 (10) (2015) 6519–6529, https://doi.org/10.1109/TIE.2015.2424399.
[10] C. Xia, J. Wang, K. McMenemy, Short, medium and long term load forecasting model and virtual load forecaster based on radial basis function neural networks, Int. J. Electr. Power Energy Syst. 32 (7) (2010) 743–750, https://doi.org/10.1016/j.ijepes.2010.01.009.
[11] G.T. Ribeiro, V.C. Mariani, L. dos S. Coelho, Enhanced ensemble structures using wavelet neural networks applied to short-term load forecasting, Eng. Appl. Artif. Intell. 82 (2019) 272–281, https://doi.org/10.1016/j.engappai.2019.03.012.
[12] A. Ahmad, N. Javaid, A. Mateen, M. Awais, Z.A. Khan, Short-Term load forecasting in smart grids: An intelligent modular approach, Energies 12 (2019) 1–21, https://doi.org/10.3390/en12010164.
[13] Y. Dong, X. Ma, T. Fu, Electrical load forecasting: a deep learning approach based on K-nearest neighbors, Appl. Soft Comput. 99 (2021) 106900, https://doi.org/10.1016/j.asoc.2020.106900.
[14] D.L. Marino, K. Amarasinghe, M. Manic, Building energy load forecasting using Deep Neural Networks, IECON Proc. (Industrial Electron. Conf. (2016) 7046–7051. https://doi.org/10.1109/IECON.2016.7793413.
[15] A. Rahman, V. Srikumar, A.D. Smith, Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks, Appl. Energy 212 (2018) 372–385, https://doi.org/10.1016/j.apenergy.2017.12.051.
[16] W. Kong, Z.Y. Dong, Y. Jia, D.J. Hill, Y. Xu, Y. Zhang, Short-term residential load forecasting based on LSTM recurrent neural network, IEEE Trans. Smart Grid. 10 (1) (2019) 841–851, https://doi.org/10.1109/TSG.2017.2753802.
[17] H. Su, E. Zio, J. Zhang, M. Xu, X. Li, Z. Zhang, A hybrid hourly natural gas demand forecasting method based on the integration of wavelet transform and enhanced Deep-RNN model, Energy. 178 (2019) 585–597, https://doi.org/10.1016/j.energy.2019.04.167.
[18] G. Chitalia, M. Pipattanasomporn, V. Garg, S. Rahman, Robust short-term electrical load forecasting framework for commercial buildings using deep recurrent neural networks, Appl. Energy. 278 (2020) 115410, https://doi.org/10.1016/j.apenergy.2020.115410.
[19] W. Guo, L. Che, M. Shahidehpour, X. Wan, Machine-Learning based methods in short-term load forecasting, Electr. J. 34 (1) (2021) 106884, https://doi.org/10.1016/j.tej.2020.106884.
[20] M.N. Fekri, H. Patel, K. Grolinger, V. Sharma, Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network, Appl. Energy. 282 (2021) 116177, https://doi.org/10.1016/j.apenergy.2020.116177.
[21] W. Gao, A. Darvishan, M. Toghani, M. Mohammadi, O. Abedinia, N. Ghadimi, Different states of multi-block based forecast engine for price and load prediction, Int. J. Electr. Power Energy Syst. 104 (2019) 423–435, https://doi.org/10.1016/j.ijepes.2018.07.014.
[22] N. Ghadimi, A. Akbarimajd, H. Shayeghi, O. Abedinia, Two stage forecast engine with feature selection technique and improved meta-heuristic algorithm for electricity load forecasting, Energy. 161 (2018) 130–142, https://doi.org/10.1016/j.energy.2018.07.088.
[23] G. Trierweiler Ribeiro, J. Guilherme Sauer, N. Fraccanabbia, V. Cocco Mariani, L. dos Santos Coelho, Bayesian optimized echo state network applied to short-term load forecasting, Energies 13 (9) (2020) 2390, https://doi.org/10.3390/en13092390.
[24] Y. LeCun, P. Haffner, L. Bottou, Y. Bengio, Object recognition with gradient-based learning, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 1681 (1999) 319–345, https://doi.org/10.1007/3-540-46805-6_19.
[25] D. Xishuang, Q. Lijun, H. Lei, Short-term load forecasting in smart grid: A combined CNN and K-means clustering approach, 2017 IEEE Int. Conf. Big Data Smart Comput. BigComp 2017. (2017) 119–125. https://doi.org/10.1109/BIGCOMP.2017.7881726.
[26] P.H. Kuo, C.J. Huang, A high precision artificial neural networks model for short-Term energy load forecasting, Energies. 11 (2018) 1–13, https://doi.org/10.3390/en11010213.
[27] M. Imani, Electrical load-temperature CNN for residential load forecasting, Energy 227 (2021) 120480, https://doi.org/10.1016/j.energy.2021.120480.
[28] L. Yin, J. Xie, Multi-temporal-spatial-scale temporal convolution network for short-term load forecasting of power systems, Appl. Energy 283 (2021) 116328, https://doi.org/10.1016/j.apenergy.2020.116328.
[29] W. He, Load forecasting via deep neural networks, Procedia Comput. Sci. 122 (2017) 308–314, https://doi.org/10.1016/j.procs.2017.11.374.
[30] H. Shi, M. Xu, R. Li, Deep learning for household load forecasting-a novel pooling deep RNN, IEEE Trans. Smart Grid. 9 (5) (2018) 5271–5280, https://doi.org/10.1109/TSG.2017.2686012.
[31] J. Kim, J. Moon, E. Hwang, P. Kang, Recurrent inception convolution neural network for multi short-term load forecasting, Energy Build. 194 (2019) 328–341, https://doi.org/10.1016/j.enbuild.2019.04.034.
[32] G. Sideratos, A. Ikonomopoulos, N.D. Hatziargyriou, A novel fuzzy-based ensemble model for load forecasting using hybrid deep neural networks, Electr. Power Syst. Res. 178 (2020) 106025, https://doi.org/10.1016/j.epsr.2019.106025.
[33] Z. Shi, W. Yao, L. Zeng, J. Wen, J. Fang, X. Ai, J. Wen, Convolutional neural network-based power system transient stability assessment and instability mode prediction, Appl. Energy. 263 (2020) 114586, https://doi.org/10.1016/j.apenergy.2020.114586.
[34] S. Zhang, Y. Wang, Y. Zhang, D. Wang, N. Zhang, Load probability density forecasting by transforming and combining quantile forecasts, Appl. Energy. 277 (2020) 115600, https://doi.org/10.1016/j.apenergy.2020.115600.
[35] R.G. da Silva, M.H.D.M. Ribeiro, S.R. Moreno, V.C. Mariani, L.dos.S. Coelho, A novel decomposition-ensemble learning framework for multi-step ahead wind energy forecasting, Energy. 216 (2021) 119174, https://doi.org/10.1016/j.energy.2020.119174.
[36] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 3 (2014) 2672–2680.
[37] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc. (2016) 1–16.
[38] M. Mirza, S. Osindero, Conditional Generative Adversarial Nets, (2014) 1–7. http://arxiv.org/abs/1411.1784.
[39] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, Proc. IEEE Comput. Soc. Conf. Comput. Vis Pattern Recognit. (2019) 4396–4405, https://doi.org/10.1109/CVPR.2019.00453.
[40] P. Isola, J.Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017. 2017-Janua (2017) 5967–5976. https://doi.org/10.1109/CVPR.2017.632.

[41] J.Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks, Proc. IEEE Int. Conf. Comput. Vis. 2017-Octob (2017) 2242–2251. https://doi.org/10.1109/ICCV.2017.244.
[42] J. Lan, Q. Guo, H. Sun, Demand side data generating based on conditional generative adversarial networks, Energy Procedia 152 (2018) 1188–1193, https://doi.org/10.1016/j.egypro.2018.09.157.
[43] C. Zhang, S.R. Kuppannagari, R. Kannan, V.K. Prasanna, Generative Adversarial Network for Synthetic Time Series Data Generation in Smart Grids, 2018 IEEE Int. Conf. Commun. Control. Comput. Technol. Smart Grids, SmartGridComm 2018. (2018) 1–6. https://doi.org/10.1109/SmartGridComm.2018.8587464.
[44] M.N. Fekri, A.M. Ghosh, K. Grolinger, Generating energy data for machine learning with recurrent generative adversarial networks, Energies 13 (2019) 1–23, https://doi.org/10.3390/en13010130.
[45] C. Tian, C. Li, G. Zhang, Y. Lv, Data driven parallel prediction of building energy consumption using generative adversarial nets, Energy Build. 186 (2019) 230–243, https://doi.org/10.1016/j.enbuild.2019.01.034.
[46] D. Zhou, S. Ma, J. Hao, D. Han, D. Huang, S. Yan, T. Li, An electricity load forecasting model for Integrated Energy System based on BiGAN and transfer learning, Energy Rep. 6 (2020) 3446–3461, https://doi.org/10.1016/j.egyr.2020.12.010.
[47] Y. Chen, Y. Wang, D. Kirschen, B. Zhang, Model-free renewable scenario generation using generative adversarial networks, IEEE Trans. Power Syst. 33 (3) (2018) 3265–3275, https://doi.org/10.1109/TPWRS.2018.2794541.
[48] Y. Gu, Q. Chen, K. Liu, L. Xie, C. Kang, GAN-based Model for Residential Load Generation Considering Typical Consumption Patterns, 2019 IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf. ISGT 2019. (2019). https://doi.org/10.1109/ISGT.2019.8791575.
[49] G. Zhang, J. Guo, A novel ensemble method for residential electricity demand forecasting based on a novel sample simulation strategy, Energy. 207 (2020) 118265, https://doi.org/10.1016/j.energy.2020.118265.
[50] Y. Wang, G. Hug, Z. Liu, N. Zhang, Modeling load forecast uncertainty using generative adversarial networks, Electr. Power Syst. Res. 189 (2020) 106732, https://doi.org/10.1016/j.epsr.2020.106732.
[51] R. Yuan, B. Wang, Z. Mao, J. Watada, Multi-objective wind power scenario forecasting based on PG-GAN, Energy 226 (2021) 120379, https://doi.org/10.1016/j.energy.2021.120379.
[52] V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning, (2016) 1–31. http://arxiv.org/abs/1603.07285.
[53] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein GaN, ArXiv. (2017).
[54] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, Improved training of wasserstein GANs, Adv. Neural Inf. Process. Syst. 2017-Decem (2017) 5768–5778.
[55] X. Mao, Q. Li, H. Xie, R.Y.K. Lau, Z. Wang, S.P. Smolley, Least Squares Generative Adversarial Networks, Proc. IEEE Int. Conf. Comput. Vis. 2017-Octob (2017) 2813–2821. https://doi.org/10.1109/ICCV.2017.304.
[56] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. (2015) 1–15.