# Conditional Wasserstein GAN for Energy Load Forecasting in Large Buildings

George-Silviu Năstăsescu
*Computer Science Department*
*ETH Zürich*
Zürich, Switzerland
snastasescu@student.ethz.ch

Dumitru-Clementin Cercel
*Computer Science Department*
*University Politehnica of Bucharest*
Bucharest, Romania
dumitru.cercel@upb.ro

2022 International Joint Conference on Neural Networks (IJCNN) | 978-1-7281-8671-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IJCNN55064.2022.9892876

***Abstract*—Energy forecasting is necessary for planning electricity consumption, and large buildings play a huge role when making these predictions. Because of its importance, numerous methods to predict the buildings' energy load have appeared during the last decades, remaining an open area of research. In recent years, traditional machine learning techniques such as Random Forest, K-Nearest Neighbors, and AutoRegressive Integrated Moving Average (ARIMA) have been replaced with deep learning methods, which have an increased ability to capture underlying consumption trends. However, large amounts of data are mandatory for training neural networks to forecast energy load. With scarce data, augmentation techniques are necessary to ensure high-quality predictions. This paper introduces cWGAN-GP-SN, a conditional (convolutional) Wasserstein Generative Adversarial Network with Gradient Penalty and Spectral Normalization used to generate new electrical records. Our architecture leverages the advantages of multiple GAN models to enrich training stability and data quality. The experimental results based on the Building Data Genome dataset show how classification and regression tasks benefit from the enrichment of the dataset. Additionally, adversarial attacks were performed to investigate whether models trained on large amounts of synthetic data are more robust.**

***Index Terms*—Energy Consumption, Generative Adversarial Network, Wasserstein Distance, Data Augmentation, Adversarial Attacks.**

## I. Introduction

Every developed city has its complex energy system, but the power grid must be reliably and efficiently managed to evolve into a smart city. While its primary purpose is to be highly available and not failure-prone, the level of waste that can be achieved is both economically and environmentally harmful.

In order to prevent this from happening, numerous measures have been taken. The best known is replacing the systems based on fossil fuel and nuclear energy with ones that use green energy as much as possible because it does not affect nature and is free to collect. The most common eco-energy resources are solar, wind, and hydropower. The primary issue is that even after investing vast amounts of money in building large-scale hydroelectric power plants, solar photovoltaic panels, and wind farms, the produced energy is far from enough for metropolises.

In an attempt to solve this problem, reasonable estimations of resources can save much money and still supply customers' needs, but it is not a simple task. Decisions must be taken in real-time, and the energy demand is hard to be anticipated. This task is too complicated for humans to do, so artificial intelligence-based solutions are a more reasonable choice.

Many machine learning algorithms [1], [2] have been developed to make qualitative forecasts of the energy load. Among them, the best-performing ones [3] use deep neural networks to make these predictions because of possessing great learning abilities and effectively use data parallelism. The downside is that they need rich datasets for learning useful patterns, and most of the current consumption measurements of large buildings are not public. Collecting new data from large buildings implies a lot of effort and patience because the measurements must be taken over at least one year to be truly valuable. Thus, data augmentation is a common approach to combat this shortcoming [4]. For time series forecasting, the classic augmentation data techniques such as rotations, flipping, scaling, jittering, permutation, slicing, and window warping are limited and generally do not introduce considerably new information to the dataset [5].

To facilitate the research on energy consumption forecasting, we design a conditional (convolutional) Wasserstein Generative Adversarial Network with Gradient Penalty and Spectral Normalization (cWGAN-GP-SN) framework. It can be easily applied to any time series regardless of the number of data features or labels the data is conditioned on. Our contributions are summarized as follows: (i) we introduce a novel Generative Adversarial Network (GAN) [6] architecture able to generate veridical and diverse samples that will help the predictors better understand the data distribution and make better forecasts in the context of energy consumption; (ii) we use both gradient penalty [7] and spectral normalization [8] to stabilize the training process; (iii) we generate multivariate data to increase both the quality and interpretability of the synthetic data; (iv) we tested several adversarial attacks [9] to the energy consumption forecasting domain to examine whether models trained on more data are more robust.

The paper is structured as follows. Section II reviews relevant related works on energy forecast as well as data augmentation in time series. Section III describes the propose architecture in detail. Section IV presents the dataset, the hyperparameters, and the feature engineering we perform. Section V displays a visualization of all data, the boost in

978-1-7281-8671-9/22/$31.00 ©2022 IEEE

performance of predictors after using the new data, as well as how adversarial attacks perform on these models. Finally, the conclusions and future directions are shown in Section VI.

## II. Related Work

### A. Energy Forecast

The demand and usage of electric energy are constantly growing since almost every activity done in this modern era hinges on electricity. Thus, a performant energy load predictor is crucial to deal with this scale of electricity consumption. Both academia and industry have been trying to find the best solution for decades, but it has remained an open research area. Ghelardoni et al. [10] emphasize the importance of the long-term load forecasting problem compared to the short-term one, proposing Support Vector Regression with Empirical Decomposition Mode to accomplish this task, stating its ability to make valuable predictions for several months ahead. Nichiforov et al. [11] use deep neural networks with Long Short-Term Memory (LSTM) layers [12], a common approach when it comes to sequence-learning models thanks to their capacity to maintain information in memory for longer periods compared to the vanilla recurrent neural networks [13]. The shortcoming is that they use datasets with not enough samples, which hinders the learning process of the actual data distribution; thus, data augmentation is needed.

### B. Data Augmentation Methods for Time Series

Merging multiple existing datasets to create a new large one may seem to be a good idea. However, this does not work as well as expected because every dataset has its features, making them incompatible. Thus, instead of using data from several places, it is possible to use data from one place and enrich it.

Data augmentation is widely used in the Computer Vision (CV) field [14], but not as much when it comes to sequences. Le Guennec et al. [15] present two methods of augmentation that they used for a classifier, namely window slicing, consisting of extracting slices from the time series and classifying each of them, the majority vote deciding the predicted label, and window warping, which randomly selects slices and speeds them up and down. Although these methods may improve the quality of datasets, they lack imagination, creating similar data to what already exists.

GANs represent the most popular framework for generating new realistic samples for a dataset. Even though it is mainly used in CV to create new images like human faces [6], it also fits perfectly with sequences. Gu et al. [16] use a GAN-based model for residential load generation, considering typical consumption patterns. Pang et al. [17] use generative adversarial learning for commercial building electricity load prediction rather than using it on classification tasks as it is more common. Koochali et al. [18] propose a novel probabilistic model for multivariate time-series forecasting, which employs the conditional GAN framework. Fekri et al. [4] generate new energy data using a recurrent GAN with LSTM layers and does various data pre-processing techniques to create new features but use windows with only 60-time steps to reduce the network capacity and training time.

## III. Method

### A. General Overview

GAN framework was introduced in 2014 by Goodfellow et al. [6] to be used for generating high fidelity data from pure noise. It consists of two jointly optimized networks: a generator that learns the data distribution and a discriminator that assesses the probability that a sample is real or fake. The generator tries to fool the discriminator during the training process by making higher-quality samples until they become indistinguishable from the real ones.

This work presents a conditional (convolutional) Wasserstein GAN with gradient penalty and spectral normalization, called cWGAN-GP-SN. Its generator and critic are presented in detail in Fig. 1. Our solution leverages the advantages of multiple architectures:

- The high-quality results of *GAN* [6].
- The auxiliary information to foster a deterministic behaviour of the model from the *conditional* case [19].
- The increased stability of the training process with lower chances of mode collapse [20], and vanishing gradients of the *Wasserstein-1 distance* [21].
- The utility of the critic to measure sample quality as proposed in *MH-GAN* [22].
- The simplicity and effectiveness of satisfying the 1-Lipschitz constraint [21] with *spectral normalization* [8].
- The close-to-1 gradient values as a result of the two-sided *gradient penalty* [7].

### B. Generator

The generator consisted of convolutional layers to learn useful patterns while generating data and upsampling layers to increase signal length. Dense, reshape, and flatten layers were used to match different data lengths. Batch normalization accelerated the convergence, while the Gaussian Error Linear Unit (GELU) activation function [23] was preferred to avoid the vanishing gradient problem [24]. Another activation function was the hyperbolic tangent to generate data between -1 and 1. Also, spectral normalization was used to regularize the weights and make the training process more stable.

### C. Critic

Although the second component of a GAN architecture is often referred to as ”the discriminator”, in the case of Wasserstein GAN (WGAN), it is called ”the critic”, as it outputs a real value instead of a $0/1$ label. Regarding its structure, spectral normalization was applied on all trainable layers to satisfy the 1-Lipschitz condition. Convolutional layers with a stride of 2 halve the length of the tensors to achieve a proper size. Dense, reshape, and flatten layers, as well as the GELU activation function, were used here for the same reasons as we mentioned above. At the same time, dropout [25] was used to add extra regularization. The final dense layer has no activation, the loss function requiring a real value.

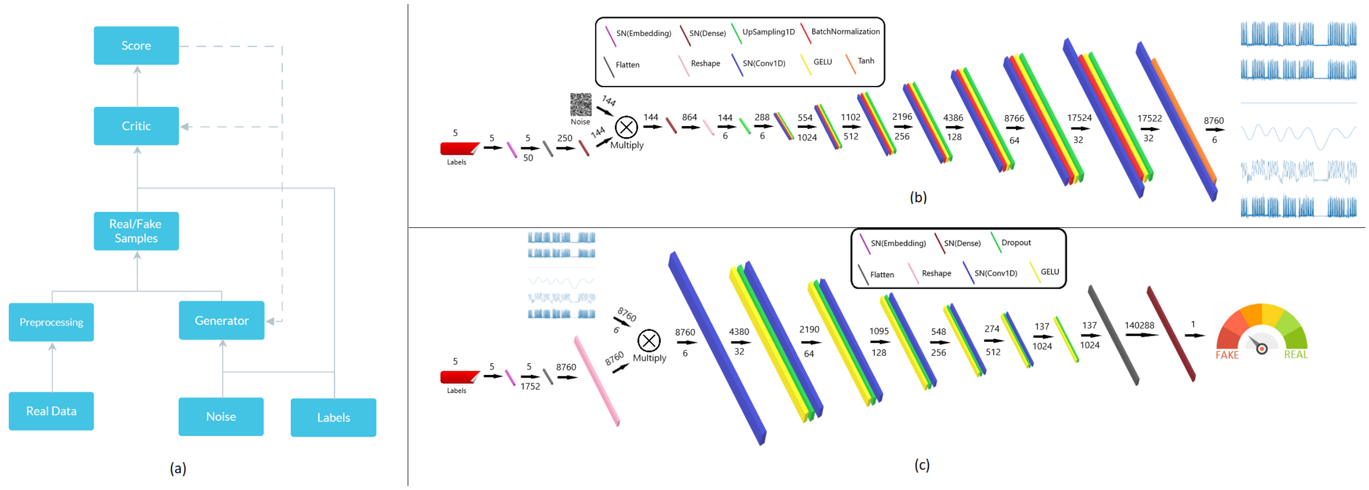

Fig. 1. (a) Schematic diagram of our cWGAN-GP-SN. (b) The Generator structure. (c) The Critic structure.

### D. Model Training

Algorithm 1 summarizes the generator and critic training process. The loss function of the generator network is complex and consists of three parts:

- Wasserstein loss. It encourages the generator to produce more realistic data to fool the critic.
- First-order derivative of the first Fast Fourier Transform (FFT) [26] feature, a transform which uses only the main frequency, so the resulted signal is a straight line. These are computed by evaluating finite differences, i.e. every left measurement is subtracted from its right neighbour. All these values should be zero in theory. Therefore, the mean of their absolute values is calculated, and any value greater than zero is penalized. It is scaled by a factor $\lambda_{G1} = 100$.
- Second-order derivative of the second FFT feature. It is represented by a time series generated from the first ten dominant frequencies. Moreover, it is a smooth curve that condenses the behaviour of the original signal. In our experiment, after computing the mean and variance of the Laplacian of this feature for all real data, it was observed that the standard deviation was extremely small. Thus, after generating a sample and computing the mean of these absolute values, from it was subtracted the original mean $\mu = 1.3e - 6$. Finally, the absolute value of this difference was penalized for encouraging it to tend to zero. It was scaled by a factor $\lambda_{G2} = 10$.

The loss function of the critic network (1) is the one specified in the WGAN with gradient penalty paper [7]. The first two terms make the critic discern between real/fake data, while the third term is the 2-sided gradient penalty that forces close-to-1 gradients to respect the 1-Lipschitz constraint.

$$\begin{aligned} \mathcal{L} = & \underset{\tilde{x}\sim\mathcal{P}_g}{\mathbb{E}}[C(\tilde{x})] - \underset{\hat{x}\sim\mathcal{P}_r}{\mathbb{E}}[C(\hat{x})] \\ & + \lambda_C \underset{\bar{x}\sim\mathcal{P}_{\bar{x}}}{\mathbb{E}}[(||\nabla_{\bar{x}} C(\bar{x})||_2 - 1)^2] \end{aligned} \tag{1}$$

**Algorithm 1**: Mini-batch SGD training process of cWGAN-GP-SN.

```
for #epochs do:
    Sample mini-batch $(x, y) \sim P_r$
    for $k$ steps do:
        $\epsilon \sim N(0, \sigma^2)$
        $\alpha \sim U(0, 1)$
        Sample mini-batch $z \sim P_g$
        $\tilde{x} \leftarrow G_\theta(z \mid y) + \epsilon$
        $\hat{x} \leftarrow x + \epsilon$
        $\bar{x} \leftarrow \hat{x} + \alpha(\tilde{x} - \hat{x})$
        Optimize critic:
        $L_C \leftarrow C_\phi(\tilde{x} \mid y) - C_\phi(\hat{x} \mid y) + \lambda_C \left(\left\|\nabla_{\bar{x}} C_\phi(\bar{x} \mid y)\right\|_2 - 1\right)^2$
    end
    Sample mini-batch $z \sim P_g$
    $\tilde{x} \leftarrow G_\theta(z \mid y)$
    Optimize generator:
    $L_G \leftarrow - C_\phi(\tilde{x} \mid y) + \lambda_{G1} \mathbb{E}_{\tilde{x}}[\,|\,\tilde{x}'_{FFT1}\,|\,] + \lambda_{G2} \,|\, \mathbb{E}_{\tilde{x}}[|\,\tilde{x}''_{FFT10}|] - \mu \,|$
end
```

## IV. EXPERIMENTS

### A. Data Analysis

The evaluation was carried out on the Building Data Genome dataset [27], which is a collection of 507 whole building electrical meters, most of them from American university campuses. Each has at least 8760 measurements because the data was collected every hour over one year. Most of the data were obtained by conducting several site visits but also from online sources.

Although a raw version of the data exists, the pre-processed one without gross outliers and small gaps was preferred. Also, these data are accompanied by metadata to bring additional helpful information, such as weather and gross floor area. These metadata can be useful in classification and regression tasks, but they were set aside for the presented data augmentation process.

Because data were collected between the 1st of January 2010 and the 1st of January 2016, 2012 was a leap year, and not all of the records began on the 1st of January, it was pre-processed again to have 8760 measurements and to be between the 1st of January and 31st of December, without taking into account the year. After reorganizing the data to conform to a format easy to understand and to work with, the data was normalized to facilitate the training process. Finally, the chosen labels to be fed to the generator and critic were as follows: industry, sub-industry, primary space use, continent, and city.

### *B. Hyperparameters and Feature Engineering*

The dataset has only 507 electrical records, so a batch size of 32 was chosen. By analyzing the training process using a callback monitor, 500 epochs seemed enough. This monitor aimed to plot ten randomly generated samples conditioned by some known labels, which were compared to the real ones with the same labels to find similarities. To obtain more features, Auto-ARIMA [28] and Real FFT (RFFT) [29] were used before data normalization. Auto-ARIMA was preferred because it automates the configuration process of the ARIMA model, finding the most suitable hyperparameters based on a selection criterion, the corrected Akaike information criterion [30] in our case. RFFT was chosen instead of FFT as the electrical values are not complex, reducing time and memory consumption. In this paper, it will still be referred to as FFT for simplicity.

The $p, q$, and $d$ hyperparameters of the ARIMA models had a value of 1 most of the time, meaning that only one time lag, one error term, and a degree of differencing of one were sufficient to fit the data. To the ARIMA fitted values were appended 4 FFT features created by keeping the first $n = 10^x, x \in \{0, 1, 2, 3\}$ values in the frequency domain and reproducing the time sequence. All these features define a dataset of shape (507, 8760, 6), which is normalized afterwards.

At every step, the critic is optimized for $k = 5$ times while the generator is only once. The gradient penalty weight $\lambda_C$ is equal to 10, a value which was also used by the authors of WGAN-GP [7]. For regularization purposes, Gaussian noise with mean $\mu = 0$ and standard deviation $\sigma = 0.02$ is used on both real and generated data. It is added before penalty computation only to the first, second, and last features. The other features are very smooth, so adding noise to them would be destructive. For both the generator and the critic, the Adam optimizer [31] was used with a learning rate of $\gamma = 2e - 6$, and $\beta_1 = 0.5, \beta_2 = 0.9$ decay rates. When the training ends, a total number of 5000 samples are generated with 1000 trials for each one.

## V. Results

It was noticed that a spike in the first FFT feature corresponds to oscillating curves for the rest of the features. This error was always localized between the 5000 and 5200 measurements (i.e., almost a summer week). Because only the first feature is of interest, FFT was used, and the first 1000 frequencies are preserved to restore the original signal with the inverse transform (i.e., IRFFT), which is the same process used to create the fourth feature. In this way, the oscillations disappeared without losing critical information. Fig. 2 shows one generated signal, its first FFT feature, and the post-processed signal.

Extensive analysis has been carried out on this architecture, namely multiple auxiliary classification and regression models, different GAN architectures, data visualizations using t-SNE [32] with Principal Component Analysis (PCA) [33] and adversarial attacks for classification and regression tasks. To perform more studies, a conditional Deep Convolutional GAN (cDCGAN) model [34] was trained to be compared to our proposed cWGAN-GP-SN. This choice was made because their architectures are conceptually similar. The difference between them is the loss function, which does not use the Wasserstein-1 distance, but the modified version of the GAN's original loss function, which usually suffers from mode collapse. Inspired by [4], we conduct experiments in various ways:

- Train on Real, Test on Real (TRTR). It shows how the model behaves when using only the original data. Although it does not reflect the quality of the synthetic data directly, it allows comparison with the models trained or tested on the synthetic data.
- Train on Synthetic, Test on Real (TSTR). It reflects how well the generated data can replace the real ones in the training process. Usually, if the GAN collapses, the results are wrong because the diversity of the real data was not captured properly. Otherwise, if TSTR and TRTR accuracies are close and not small, then it can be considered that the generated data is closely related to the real data.
- Train on Real, Test on Synthetic (TRTS). It is theoretically like TSTR in terms of the information that emerges, although it has less practical relevance as the models are designed to be used on real data.
- Train on Synthetic, Test on Synthetic (TSTS). TRTR and TSTS are very much alike, offering additional information for comparisons. Here, a much higher value than the other signals mode collapse.

### *A. Qualitative Analysis*

The full dataset contains 507 real data, 5000 cDCGAN generated samples and 5000 instances produced by cWGAN-GP-SN. To perform t-SNE to visualize the performance of cDCGAN and cWGAN-GP-SN, a feature extractor was needed to transform electrical sequences into meaningful features. Luckily, the already trained critic without the final one-neuron layer turned out to be very useful. PCA was initially performed to reduce the number of the resulted features, retaining only the first 100 components, which preserved 99.8% of their variance. t-SNE with 40 as perplexity and 100 as the number of iterations without progress was applied afterwards.

Fig. 3 shows that more than half of the real data (red dots) are in only one (right side) cluster, emphasizing its unbalanced character. Synthetic data produced by cDCGAN can be seen only in concentrated clusters, and only those in the centre

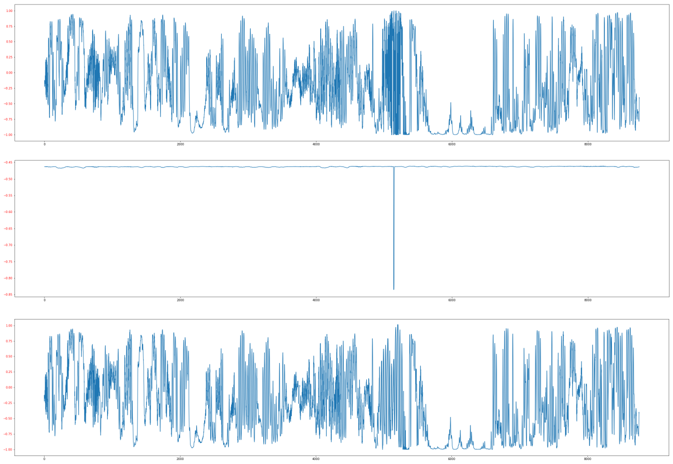

Fig. 2. Example of generated signal (top), its first FFT feature (middle), and the result after post-processing (bottom).

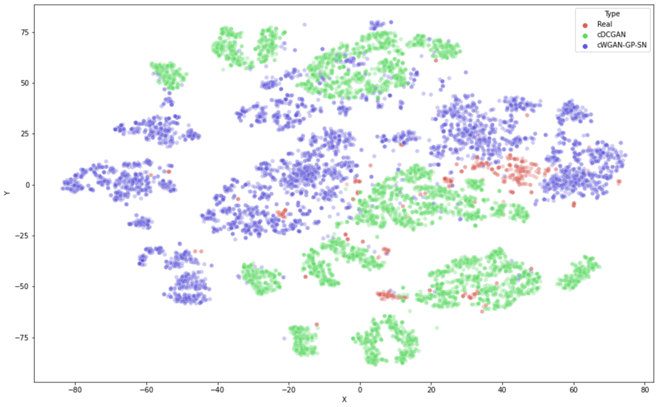

Fig. 3. Real and synthetic data visualization.

and right side have real values nearby, which can signal the presence of the mode collapse problem. On the other hand, the cWGAN-GP-SN samples are better spread, and each real point has at least one generated point nearby, encouraging the idea that this model learned the entire data distribution and not just some of its modes.

## B. Results for Classification

An enriched dataset should lead to better classifiers if the synthetic data succeeds in creating new realistic samples. To validate this idea, TRTR along with TSTR, TRTS and TSTS for our cWGAN-GP-SN and only TSTR for cDCGAN were conducted. The classifier was trained identically for every experiment to be relevant to the analysis. The dataset was split into train-validation-test by 80%-10%-10% ratio, and the labels were one-hot encoded. The classifier was implemented using a stack of six convolutional blocks with convolutional, batch normalization, max pooling, and dropout layers, with ReLU activation and with a final softmax layer. Adam optimizer was chosen, and accuracy, precision, recall, and F1-score metrics were computed during training and testing.

TABLE I
CSLASSIFICATION RESULTS USING FIVE SCENARIOS ON THE BUILDING DATA GENOME DATASET.

| Scenario | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| TRTR | 22.65% | 25.54% | 18.31% | 20.72% |
| TSTR | *56.14%* | 60.72% | *42.17%* | *48.92%* |
| cDCGAN TSTR | 40.24% | *66.62%* | 32.41% | 42.95% |
| TRTS | 35.42% | 35.90% | 31.81% | 33.01% |
| TSTS | **96.14%** | **95.90%** | **95.66%** | **93.73%** |

In Table I, we report the results of the classifier in all five scenarios. Bold values are the best overall values, while the italic ones represent the best values obtained when models were tested on real data, i.e. TRTR, TSTR and cDCGAN TSTR, reflecting their performance when applied in real life. From this table, we can draw the following conclusions:

- TRTR is too small to make qualitative comparisons with TSTS, acting like randomly choosing a class, whereas the TSTS does a great job but can reflect mode collapse.
- TRTS and TSTR obtain better performance than TRTR and worse performance than TSTS. This observation shows the significant information captured by the generator while learning the data distribution, the similarity of the real and synthetic data, and the performance boost by using a larger dataset.
- The classifier trained on real data is the weakest, the main reason being the small number of training samples and unbalanced dataset.
- The model trained on cDCGAN data outperforms the previous one. This can happen if the generator learned useful patterns and created good-looking time series, or suffered from mode collapse, the memorized data matching the test data. The latter is encouraged by the large difference between precision and recall.
- The model trained on cWGAN-GP-SN samples has better results in most of the cases, which reflects its superiority.

## C. Results for Regression

Similar to classification, an enriched dataset should lead to better regression models if the synthetic data succeeds in creating new realistic samples. The same five scenarios were studied here.

A month of previous data seemed to be enough to predict the energy load for the next hour, so every window consisted of 720 measurements. Because a sample has 8760 measurements, the window size is 720, and only the next value is predicted, every sample can be broken into 8040 different windows. The dataset obtained would be very large, but this does not mean that all this information will be useful. Most of it is redundant because neighbouring windows would be extremely similar. For this reason, a data generator was used. This special structure randomly selects samples, and for each one, a specific starting hour is picked, from which the window will begin. These hours are important because they encode

TABLE II
REGRESSION RESULTS USING FIVE SCENARIOS ON THE BUILDING DATA GENOME DATASET.

| Scenario | MSE | MAE | MAPE |
|---|---|---|---|
| TRTR | ***9.57*** | ***2.34*** | ***3.99%*** |
| TSTR | 23.33 | 3.70 | 5.98% |
| cDCGAN TSTR | 31.90 | 4.09 | 6.61% |
| TRTS | 20.26 | 3.58 | 7.68% |
| TSTS | 13.35 | 2.95 | 6.98% |

the period in which that signal occurred and provide useful extra information. Using this approach, 200 steps per epoch and 50 validation steps, a different batch is generated at every step, and the network uses it for training and validation. This method generalizes well and has a low computational cost.

The chosen architecture was a recurrent convolutional neural network [35], where convolutional layers were used, the resulting features being fed to the recurrent layer. Specifically, three convolutional blocks with kernel sizes of 3, 5, and 7 were used as feature extractors, with the same components as those used for classifications. The concatenated features were fed to two consecutive LSTM layers. Then, the results are concatenated with the embedded labels, and a dense layer with hyperbolic tangent activation outputs the result. Adam optimizer was chosen, and Mean Square Error (MSE), Mean Average Error (MAE), and Mean Absolute Percentage Error (MAPE) [36] were computed during training and testing.

To our knowledge, only the study in [4] has attempted to enrich this dataset by using a GAN variant called R-GAN. Their architecture is similar to ours in terms of pre-processing and choosing the best sample, but it is not conditional, uses GAN's original loss function, and has LSTM layers that work with windows of only 60 measurements. The authors performed a similar analysis with MAE and MAPE, but the exact structure of their predictors was not specified; thus, it was hard to compare their results to ours.

From Table II, we can draw the following conclusions:

- TRTR has the best values for every metric. This reflects that splitting one-year-long sequences into multiple one-month windows (not only 12 windows but many more because the starting point is randomly chosen) increases the amount of data sufficiently to train a regression model with them properly.
- TRTS and TSTR are comparable, and none of them is always better than the other. This leads to the idea that real and synthetic data look alike.
- cDCGAN TSTR has the worst results of all three cases where the model was tested on real data. This shows that our architecture is superior.
- TSTS has the second-best results for MSE and MAE (after TRTR). This emphasizes that our GAN doesn't suffer from mode collapse; otherwise, it would have had almost perfect results.
- cWGAN-GP-SN surpasses cDCGAN in this case too.

### D. Adversarial Attacks

Although neural networks deliver excellent results in various applications, they are hardly interpretable. They act like black boxes, receiving inputs and giving outputs without showing how the learned patterns used behind are correct and meaningful. Adversarial attacks [9] appeared to exploit this weakness, crafting inputs designed to trick the models. The creation process consists of taking real inputs and adding very small values to them until the predicted result is wrong. They are commonly used in the image classification task [37], where a small perturbation is added to every pixel, making the original and resulting images look indistinguishable from a human perspective but completely different from a predictor.

The cWGAN-GP-SN architecture proved to generate realistic samples, which help classification and regression models to learn useful patterns. TSTS values were always the best ones, so we decided that several attacks should be applied to the models trained on synthetic data, which will be adversely modified to test whether good predictions are maintained. We evaluate the impact of the following adversarial attacks:

- Fast Gradient Signed Method (FGSM) [38], which is among the first to appear and is still often used. It works by computing the loss function and backpropagating the gradients to inputs to maximize the loss value.
- Basic Iterative Method (BIM) [39], being the iterative version of FGSM.
- Projected Gradient Descent (PGD) [40], which is a BIM starting from a random point in the $\epsilon$-neighborhood of the original input.
- Carlini&Wagner (C&W) [41], considered one of the most effective adversarial attacks. It transforms the simple iterative approach into an optimization problem where the goal is to reach the adversarial example with the minimum perturbation.
- Boundary Attack (BA) [42], a decision-based attack. It is considered more robust because techniques such as obfuscation of gradients do not affect it. It starts with a large perturbation and seeks to reduce it while remaining adversarial.
- Elastic-Net (EN) [43], inspired by C&W and the elastic net regularized regression.
- DeepFool (DF) [44], which considers at every step all possible results to find the optimal direction and step size.
- NewtonFool (NF) [45], inspired by DF, not search for the best direction to make the model misclassify, but the best step to decrease the probability of the correct label.

For the classification task, these adversarial attacks were carried out under three usage scenarios:

- Untargeted with predictions instead of labels to prevent label leaking [46].
- Untargeted with labels. The attacks were supposed to make the model predict a different class by adding a small perturbation to the input.
- Targeted. Target labels were explicitly selected to be very different from the true label.

TABLE III
UNTARGETED WITH PREDICTIONS (UP), UNTARGETED WITH LABELS (UL), AND TARGETED (T) ATTACKS RESULTS.

| | Accuracy | | | Perturbation | | |
|---|---|---|---|---|---|---|
| **Attack** | **UP** | **UL** | **T** | **UP** | **UL** | **T** |
| Original | 0.962 | 0.962 | 0.962 | 0 | 0 | 0 |
| FGSM | 0.088 | 0.066 | 0.054 | 5.335 | 5.335 | 5.335 |
| BIM | **0.018** | **0** | **0** | 4.290 | 4.290 | 3.850 |
| PGD | 0.020 | **0** | **0** | 4.345 | 4.345 | 3.850 |
| C&W | 0.470 | 0.350 | 0.646 | **0.165** | 0.275 | **0.220** |
| BA | 0.026 | 0.018 | 0.132 | 5.555 | 5.555 | 5.995 |
| EN | 0.342 | 0.332 | 0.384 | 5.555 | 5.445 | 7.150 |
| DF | 0.076 | 0.076 | - | 9.075 | 9.075 | - |
| NF | 0.198 | 0.198 | - | **0.165** | **0.165** | - |

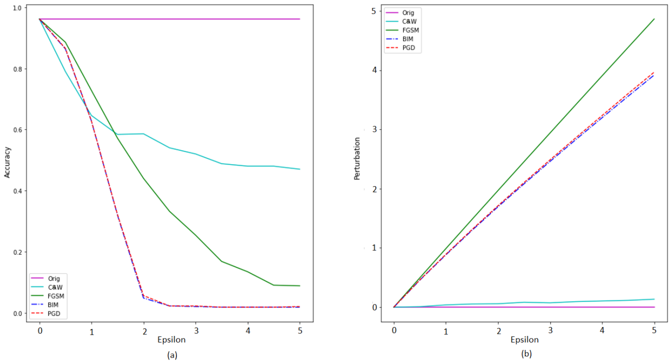

Fig. 4. (a) Accuracy variation and (b) Perturbation variation, both with respect to the epsilon value for FGSM, BIM, PGD, C&W) in the UL scenario.

The first 4 of them have a maximum perturbation parameter $\epsilon$, which helps visualize its effect as it increases. Only the visualization of the untargeted with labels use case will be presented due to space limitations. For all the adversarial attacks, the resulted accuracy of the classifier and the average perturbation (we will call it ”Perturbation”) of the input values were reported. Analyzing the results of the attacks presented in Fig. 4 and Table III, it can be stated that:

- FGSM, BA, EN, and DF do not stand out.
- BIM and PGD are almost identical, and only these succeed in completely fooling the model.
- C&W is unsurpassed in terms of added perturbation, making adversarial examples almost indistinguishable from real ones. However, NewtonFool achieves better performance when the maximum perturbation is higher.
- Unlabeled data tends to create slightly less accurate examples with slightly less perturbation, so there is not a big difference between these two approaches.
- Targeted results show poorer performance. This is caused by the complete opposite chosen target, which requires more effort to make the model misclassify.

Only FGSM, BIM, and PGD were implemented for the regression task because their algorithms are intuitive and easy to extend to the regression case. The three usage scenarios used here were:

- Untargeted, where any big difference between the predicted and actual value is accepted.
- Max targeted, where the attacker tries to fool the model to predict a value as large as possible, which in practice would result in a waste of resources to produce more electrical energy.
- Min targeted, the opposite of the previous one, where fewer resources than needed would be used.

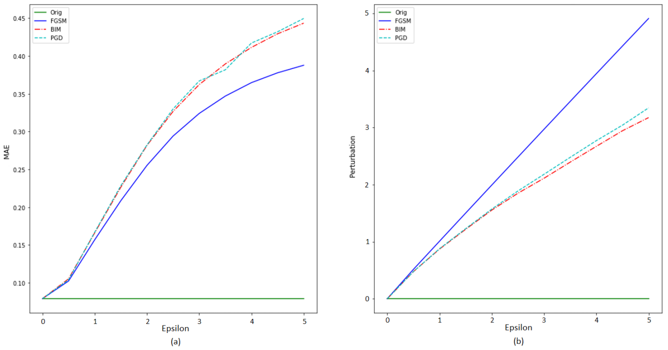

Fig. 5. (a) MAE variation and (b) Perturbation variation, both with respect to the epsilon value for FGSM, BIM, PGD in the untargeted scenario.

Only the visualization of the untargeted scenario will be presented due to space limitations. As shown in Fig. 5, it can be concluded that:

- Although FGSM is fast, it is still inferior.
- BIM and PGD results are almost identical, like before.

## VI. CONCLUSIONS

This paper proposed cWGAN-GP-SN, a novel GAN-based architecture that encapsulated mechanisms used in other variations to generate realistic electrical sequences and combat its common problems: vanishing gradient, mode collapse, and convergence failure. To the best of our knowledge, this is the first work that presents a conditional (convolutional) Wasserstein GAN with both gradient penalty and spectral normalization to perform energy consumption forecasting in large buildings.

Five features were added to help neural networks better understand the data. Because of its reduced size, we augmented the Building Data Genome dataset by training the cWGAN-GP-SN on it. We have shown that this approach is superior to the similar cDCGAN variant by analyzing the generated data alone and when these data are used for classification or regression. Subsequently, various adversarial examples were crafted to emphasize that more training samples are insufficient to preserve the model’s accuracy. Also, a quantitative analysis was performed to visualize how different adversarial attacks influence the predictions.

In the future, we plan to enrich the proposed system by adding a classification branch to the critic, like AC-GAN [47]. This will encourage the critic to learn more useful patterns and stabilize the training process when the number of classes is too high, allowing class-wise data separation.

# References

[1] A. Camara, W. Feixing, L. Xiuqin *et al.*, "Energy consumption forecasting using seasonal arima with artificial neural networks models," *International Journal of Business and Management*, vol. 11, no. 5, p. 231, 2016.

[2] C. Nichiforov, G. Stamatescu, I. Stamatescu, V. Calofir, I. Fagarasan, and S. S. Iliescu, "Deep learning techniques for load forecasting in large commercial buildings," in *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2018, pp. 492–497.

[3] M. M. Forootan, I. Larki, R. Zahedi, and A. Ahmadi, "Machine learning and deep learning in energy systems: A review," *Sustainability*, vol. 14, no. 8, p. 4832, 2022.

[4] M. N. Fekri, A. M. Ghosh, and K. Grolinger, "Generating energy data for machine learning with recurrent generative adversarial networks," *Energies*, vol. 13, no. 1, p. 130, 2019.

[5] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *Plos one*, vol. 16, no. 7, p. e0254841, 2021.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[8] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations*, 2018.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[10] L. Ghelardoni, A. Ghio, and D. Anguita, "Energy load forecasting using empirical mode decomposition and support vector regression," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 549–556, 2013.

[11] C. Nichiforov, G. Stamatescu, I. Stamatescu, and I. Făgărăşan, "Evaluation of sequence-learning models for large-commercial-building load forecasting," *Information*, vol. 10, no. 6, p. 189, 2019.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[14] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[15] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data augmentation for time series classification using convolutional neural networks," in *ECML/PKDD workshop on advanced analytics and learning on temporal data*, 2016.

[16] Y. Gu, Q. Chen, K. Liu, L. Xie, and C. Kang, "Gan-based model for residential load generation considering typical consumption patterns," in *2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2019, pp. 1–5.

[17] Y. Pang, X. Zhou, D. Xu, Z. Tan, M. Zhang, N. Guo, and Y. Tian, "Generative adversarial learning based commercial building electricity time series prediction," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2019, pp. 1800–1804.

[18] A. Koochali, A. Dengel, and S. Ahmed, "If you like it, gan it—probabilistic multivariate times series forecast with gan," in *Engineering Proceedings*, vol. 5, no. 1. Multidisciplinary Digital Publishing Institute, 2021, p. 40.

[19] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[20] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, 2016.

[21] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[22] R. Turner, J. Hung, E. Frank, Y. Saatchi, and J. Yosinski, "Metropolis-hastings generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6345–6353.

[23] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[24] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] H. J. Nussbaumer, "The fast fourier transform," in *Fast Fourier Transform and Convolution Algorithms*. Springer, 1981, pp. 80–111.

[27] C. Miller and F. Meggers, "The building data genome project: An open, public data set from non-residential building electrical meters," *Energy Procedia*, vol. 122, pp. 439–444, 2017.

[28] S. L. Ho and M. Xie, "The use of arima models for reliability forecasting and analysis," *Computers & industrial engineering*, vol. 35, no. 1-2, pp. 213–216, 1998.

[29] H. V. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast fourier transform algorithms," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 35, no. 6, pp. 849–863, 1987.

[30] C. M. Hurvich and C.-L. Tsai, "A corrected akaike information criterion for vector autoregressive model selection," *Journal of time series analysis*, vol. 14, no. 3, pp. 271–279, 1993.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[32] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[33] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[34] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[35] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[36] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[37] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 819–828.

[38] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[39] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[40] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[41] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[42] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *International Conference on Learning Representations*, 2018.

[43] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: Elastic-net attacks to deep neural networks via adversarial examples," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[44] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[45] U. Jang, X. Wu, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017.

[46] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[47] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *International conference on machine learning*. PMLR, 2017, pp. 2642–2651.