Generative Adversarial Network with Soft-Dynamic Time Warping and Parallel Reconstruction for Energy Time Series Anomaly Detection

Hardik Prabhu [1], **Jayaraman Valadi** [2], **Pandarasamy Arjunan** [3]

[1, 3]Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science, Bengaluru, India
[2]School of Computing and Data Sciences, FLAME University, Pune, India
hardik.prabhu@gmail.com, jayaraman.vk@flame.edu.in, samy@iisc.ac.in

Abstract

In this paper, we employ a 1D deep convolutional generative adversarial network (DCGAN) for sequential anomaly detection in energy time series data. Anomaly detection involves gradient descent to reconstruct energy sub-sequences, identifying the noise vector that closely generates them through the generator network. Soft-DTW is used as a differentiable alternative for the reconstruction loss and is found to be superior to Euclidean distance. Combining reconstruction loss and the latent space's prior probability distribution serves as the anomaly score. Our novel method accelerates detection by parallel computation of reconstruction of multiple points and shows promise in identifying anomalous energy consumption in buildings, as evidenced by performing experiments on hourly energy time series from 15 buildings.

Introduction

Buildings consume significant energy, accounting for approximately 40% of total energy usage worldwide. The recent proliferation of smart metering systems has led to an unprecedented volume of energy time-series data. Through data-driven analysis, valuable insights about the buildings' energy use patterns have been obtained, offering informative perspectives on energy usage. However, it is crucial to acknowledge the influence of anomalies on energy management. In commercial buildings, the presence of inadequately maintained, faulty, or deteriorated hardware, along with improper operational practices, is estimated to contribute to the wastage of 15 to 30 % of energy consumption (Katipamula and Brambley 2005; Schein et al. 2006). Furthermore, if anomalous energy use instances are not accurately identified and properly corrected, it can distort the reference points used for making predictions, leading to inaccurate and unreliable future forecasts as well. Hence, energy anomaly detection is essential for efficient energy management. The existing methods in this field primarily focus on identifying power samples that show significant deviations from the normal consumption patterns, either being excessively high or low. However, this approach may not effectively capture the sequential anomalies that could indicate more complex issues in energy usage (Himeur et al. 2021).

In recent years, generative models have gained a lot of attention due to their remarkable ability to learn and mimic complex data distributions, leading to advancements in diverse fields such as image synthesis, natural language processing, and time series analysis. Generative models such as Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) can effectively learn the patterns, trends, and dependencies present in normal data and then generate synthetic data points that closely resemble the original data. GANs are good at synthesis, particularly time series, and are widely used for anomaly detection (Di Mattia et al. 2019) despite the challenges faced in the training of GANs, such as vanishing gradients, non-convergence, and diminishing gradients (Brophy et al. 2021). Their superiority in effectively identifying intricate features in time series data, aids in the detection of complex anomalies. This makes them stand out in comparison with the previously used anomaly detection models (Zhu et al. 2019).

Unlike the statistical likelihood-based anomaly detection models such as those used in (Coluccia, D'Alconzo, and Ricciato 2013), GANs do not rely on identifying the likelihood of a data point coming from the dense region of the data distribution. Instead, GANs effectively learns a mapping between a simple prior, typically the standard Gaussian distribution in a latent space to the data distribution. The mapping of the data distribution to the normal distribution is then followed by the reconstruction of data points from the normal distribution. If the GAN is unable to generate (reconstruct) a particular sample accurately, it can be logically concluded that the sample is an anomaly (Li et al. 2019; Bashar and Nayak 2020; Geiger et al. 2020). In our work, we invert the generator network (inverse mapping) of the GAN for reconstruction by applying gradient descent in the latent space (Schlegl et al. 2017), and subsequently perform anomaly detection.

The main contributions of this paper are:

- Efficient utilization of Generative Adversarial Networks (GANs) for anomaly detection in univariate energy time series, derived from meter readings from a dataset of real buildings.
- Proposing Soft-DTW (Cuturi and Blondel 2017) as an alternative to Euclidean distance as a differentiable reconstruction loss.

- Parallel computation of reconstruction of multiple data points simultaneously, effectively mitigating the performance bottleneck.
- Adapting a method for effectively evaluating sequence-level anomaly detection with GANs for dataset annotated with pointwise anomaly labels.

Background and Motivation

It has been reported that more than 20% of the total energy consumed within buildings is wasted due to various factors (Roth et al. 2005). Therefore, it is important to identify these energy wastage events to reduce buildings' operational costs. Various existing research has centred on data-driven anomaly detection methods that utilize smart metering data (Himeur et al. 2021). Despite these advancements, robust anomaly detection in real buildings remains a challenging task due to the complex nature of anomalies caused by various deviations in the operational patterns of the buildings. While there is increasing interest among them, their primary focus is on detecting instantaneous energy wastage (point anomaly). Whereas, the studies are limited in scope in detecting continuous energy wastage events (sequence anomaly) which is a challenging task (Himeur et al. 2021).

In recent years, GANs have gained increasing prominence in detecting complex anomalies across various domains. Their capability to capture complex patterns and dependencies in normal data enables them to generate synthetic data points that closely resemble the original data, making them effective tools for anomaly detection. While various variants of GANs are available, in this paper, we employ a variant known as the 1-dimensional Deep Convolutional Generative Adversarial Network (1D-DCGAN) (Radford, Metz, and Chintala 2015) for detecting anomalies in building energy time series data.

Generative Adversarial Networks (GANs)

A GAN consists of two networks, a generator (G) and a discriminator (D) which train in an adversarial setting. The main idea behind GANs is to have the generator network learn to generate data that resembles samples from a target data distribution, without explicitly modelling the distribution itself. The generator takes in random noise as input and tries to produce synthetic data samples that mimic the real data. The discriminator is trained to distinguish between real data samples from the actual dataset and fake data samples produced by the generator. The value function $V(G, D)$ for the min-max game played between the generator and the discriminator is given below.

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

With a prior distribution $p(z)$ in the noise space, the generator implicitly defines a distribution p_g for the generated output $G(z)$. The discriminator assigns a probability score $D(x)$ to input x for belonging to the target distribution p_{data}. In the original GAN (Goodfellow et al. 2014), several theoretical guarantees are derived which are important to our ongoing discussion. For a fixed G, the optimal discriminator (D^*) is given as $D^* = \max_D V(G, D)$. The optimal discriminator could be then written in terms of the generator distribution (p_g) and the data distribution (p_{data}).

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{2}$$

By substituting the optimal discriminator D^* in V, which is implicitly a function of G, the equilibrium of the min-max game is obtained by minimizing $C(G) = V(G, D^*)$. The simplified expression is given below.

$$C(G) = -\log(4) + 2 \cdot \text{JSD}(p_{\text{data}} \| p_g) \tag{3}$$

The Jensen–Shannon divergence (JSD) is non-negative and only reaches its minimum when $p_g = p_{data}$. Minimizing the JSD between the data distribution and the generator's output is desirable for data generation. However, this approach might be overly cautious, leading to mode collapse, where the generator produces limited and repetitive samples without exploring the full diversity of the data distribution. To prevent false positives during anomaly detection, it becomes essential to learn how to generate from the entire distribution of normal data. Mode collapse and other stability issues have led to the evolution of alternative loss functions.

GAN Training with Wasserstein Loss

W-GAN (Arjovsky, Chintala, and Bottou 2017) suggests an alternative training process to provide a more stable and reliable training process for GANs. Wasserstein distance measures the distance between two probability distributions. It effectively tackles issues like the vanishing gradient and mode collapse, making it a favourable choice for enhancing the training and performance of GAN models. The theoretical formulation of the distance between p_g and p_{data} is given below.

$$W(p_{data}, p_g) = \inf_{\gamma \in \Gamma(p_g, p_{data})} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{4}$$

Where γ is the joint distribution over (x, y) such that the marginal distributions of x and y are p_g and p_{data}. A more tractable version of the loss is given by the Kantorovich-Rubinstein duality.

$$W(p_{\text{data}}, p_g) = \sup_{D \in \text{Lip}(K)} \left(\mathbb{E}_{x\sim p_{\text{data}}}[D(x)] - \mathbb{E}_{y\sim p_g}[D(y)]\right) \tag{5}$$

A function (discriminator) D is K-Lipschitz if for all x and y in the domain of D, the following condition holds:

$$|D(x) - D(y)| \leq K \cdot \|x - y\| \tag{6}$$

Therefore, the generator and discriminator are required to optimize the following min-max function.

$$V(G, D) = \min_G \max_{D \in \text{Lip}(K)} \left(\mathbb{E}_{x\sim p_{\text{data}}}[D(x)] - \mathbb{E}_{y\sim p_g}[D(y)]\right) \tag{7}$$

It is important to note that the discriminator is no longer limited to being a classifier; it can be any function as long as it adheres to the Lipschitz constraint. To maintain this constraint during training, gradient clipping is applied (See (Arjovsky, Chintala, and Bottou 2017)). Additionally, the loss

function of the discriminator could be used to evaluate convergence. The absolute value of the discriminator loss in WGAN serves as an approximation of the Wasserstein distance between p_{data} and p_g.

Anomaly Detection Using GAN Inversion

The GAN in our study is trained solely on normal data, which means its generator is designed to produce only realistic samples. If the generator is unable to replicate a certain sample, it can be inferred that the sample is an anomaly. GANs, being likelihood-free models that generate realistic samples, require a mechanism to invert this generation process. This inversion maps the data back to the corresponding noise in the latent space, which is then passed through the generator. Comparing this output with the original data point helps in anomaly detection.

While TadGAN (Geiger et al. 2020) uses training an encoder along with the generator for inverse mapping, we have elected to implement a different strategy. Our approach focuses on gradient descent in the latent space, which involves separate processes for generation and reconstruction. This distinction enables us to use any differentiable loss, providing us with more flexibility in how we reconstruct data. Moreover, this way the success of our reconstruction process is largely dependent on the quality of the generator and the choice of the differentiable reconstruction error.

Drawing inspiration from AnoGAN (Schlegl et al. 2017), which uses a Deep Convolutional GAN (DCGAN) trained on normal image datasets, we also use DCGAN, in a one-dimensional version suited for time series data. To address the stability challenges in GANs highlighted by (Arjovsky, Chintala, and Bottou 2017), we implement W-GAN for training our model.

Bashar et al. (Bashar and Nayak 2020) performed inverse mapping for time-series anomaly detection using gradient descent in latent space. However, they have encountered the same limitations as AnoGAN, it requires performing extensive optimization steps for reconstruction of each data point, which results in poor test-time performance (Di Mattia et al. 2019). We address this challenge by utilizing parallel computation to simultaneously reconstruct multiple points. The details are present in the methodology section.

Methodology

Given an univariate time series $\{x_t\} = \{x_1, x_2, \dots x_T\}$, where $x_i \in R$ is a meter reading at time step i. The objective is to devise a methodology to find a subset $A \subset \{x_t\}$ such that it contains the points that deviate from the normal energy usage pattern. We define the following terms:

1. Time segment: A subset of consecutive points from a time series. For example $S = \{x_4, x_5, x_6, \dots x_{103}\}$.
2. Time sub-sequence: It is a smaller segment of the time series consisting of a fixed length, given by the variable window size.

The LEAD Dataset and Pre-processing

The LEAD1.0 dataset (Gulati and Arjunan 2022) is used in this study. This public dataset includes hourly-based electricity meter readings for commercial buildings over up to one year. Each building contains about 8,784 data points. Anomaly annotations are provided, marking individual anomalous points within each building's time series. In this study, we selected 15 buildings with adequate normal data to assess the performance of the proposed framework.

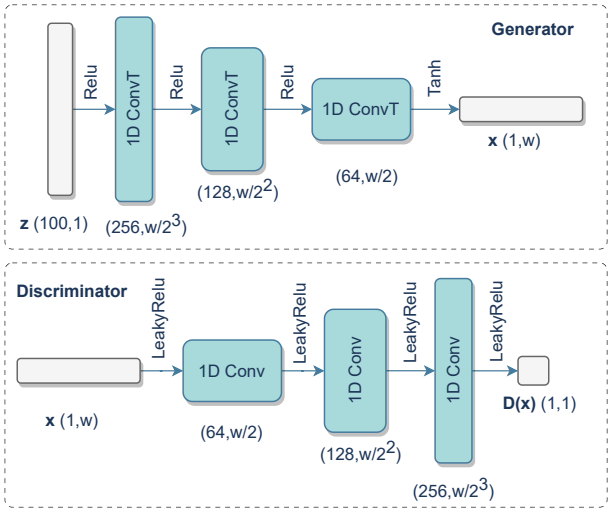

Figure 1: 1D DCGAN architecture for anomaly detection.

Train-test Segments First, the time series data for individual buildings' electricity consumption is partitioned into segments. Subsequently, a fixed-size rolling window is applied to each segment, creating time sub-sequences that serve as input to the model. For each of the annual meter reading time series in our dataset, we first remove missing readings. Then we divide each series into 25 contiguous, non-overlapping time segments. The segments contain point-wise anomaly annotations. The segments which have no anomalous points are used in training. The rest of the segments are used for testing. Each segment is normalized to be in the range [-1,1]. This step is crucial because our generator model utilizes a *tanh* activation function at its end, which outputs within this specific range.

Model Input A segment of time series data is further processed using a rolling window of fixed length to generate the model inputs. Let $S = \{x_1, x_2, \dots x_{n+w-1}\}$ be a segment of time series. Then, using window size w, a collection of n time sub-sequences $\mathbf{X}$ is generated as shown below.

$$\mathbf{X} = \{(x_1, \dots x_w), (x_2, \dots x_{w+1}), \dots (x_n, \dots x_{n+w-1})\} \tag{8}$$

The sub-sequences are overlapping in nature. The sub-sequences undergo further processing to form the model input, resulting in a tensor with the shape *(Batch size, features (1), window size (w))*.

1-D DCGAN Architecture

Both the generator and discriminator have 3 symmetrical hidden layers as shown in Figure 1. The generator uses a

Algorithm 1: Inverse Mapping with Gradient Descent

Requires: Query sub-sequence X, Generator G, Latent space prior p_z, Learning rate α, Reconstruction loss L

Result: Reconstructed sub-sequence X'

Initialize random latent vectors $Z \sim p_z$; **while** *stopping criteria not met* **do**

  Generate synthetic data samples using the generator: $X_r = G(Z)$;

  Calculate the reconstruction error: $Loss = L(X, X_r)$;

  Calculate the gradient of the loss w.r.t. Z: $\nabla_Z L = \frac{\partial Loss}{\partial Z}$;

  Update the latent vector using gradient descent: $Z = Z - \alpha \cdot \nabla_Z L$;

Reconstruct data sub-sequence using the updated latent vector: $X' = G(Z)$;

series of convolutional transpose layers to upsample the input noise vector into a 1D time series output. starting with a 1D transposed convolutional layer with 100 input channels (noise dimensions) and 256 output channels, followed by *ReLU* activation, and three more transposed convolutional layers with decreasing output channels (128, 64, and 1), followed by *Tanh* activation at the end. The discriminator consists of 4 1D convolutional layers, each followed by *Leaky ReLU* activation. In a DCGAN, batch normalization is commonly used in intermediate layers to stabilize training and improve convergence. However, it is typically not used in the generator's output layer and in the discriminator's input layer and output layer. The latent space is made up of 100 dimensions and we take the prior distribution p_z as an independent standard Multivariate Gaussian over the latent space.

Inverse Mapping

The goal of inverse mapping is to estimate the input that generated a particular output. In the context of deep learning, it is the reversing of the forward mapping that a neural network might have learned. Inverse mapping is an active area of research in representation learning, generative modelling, explainability and adversarial robustness. Key technical approaches include encoder-decoder architectures, conditional generation, and inverting using gradient descent in the input space.

In this work, we focus on the gradient descent approach in the latent (input) space. The idea is straightforward. To invert the query data X. It requires starting from noise in latent space $Z \sim p_z$ and updating it using gradient descent by evaluating a differentiable (reconstruction) loss $L(X, G(Z))$. After few iterations, Z is considered as $G^{-1}(X)$. The details are described in Algorithm 1.

Reconstruction Loss When evaluating the error (dissimilarity) between two time series sub-sequences, Dynamic Time Warping (DTW) (Berndt and Clifford 1994) is often used as a loss function rather than mean squared error (MSE). DTW allows for flexible alignment between the original and reconstructed time sub-sequence, overcoming small shifts and distortions. MSE assumes a fixed one-to-one alignment and penalizes any mismatch equally, even if just slightly misaligned. DTW compares the overall shapes of the time sub-sequences, rather than exact value matching. This is more appropriate for generative modelling where we need to capture the essence of temporal patterns and dynamics, not necessarily reproduce the original numerical values. We use Soft-DTW (Cuturi and Blondel 2017) as a differentiable alternative to DTW for reconstruction loss, enabling gradient descent-based optimization.

Given two sequences $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_m)$, the Soft-DTW distance can be defined using the following recurrence relation:

$$\begin{aligned} D[i,j] =& \text{softmin}_\gamma \left(D[i-1,j], D[i,j-1], D[i-1,j-1] \right) \\ &+ d(x_i, y_j) \end{aligned} \tag{9}$$

where $d(x_i, y_j)$ is a distance metric between the elements x_i and y_j, and softmin_γ is the soft minimum operator (Cuturi and Blondel 2017). The Soft-DTW distance between the two sequences X and Y is found at $D[n, m]$.

Parallel Computation The way the neural networks are designed, the generator G can perform parallel computations to construct k number of data points in data space from k noise vectors. These could further compared with k query points $\mathbf{X} = [X_1, X_2, ...X_k]$ to get a loss vector.

$$L(G(\mathbf{Z}), \mathbf{X}) = \begin{bmatrix} L(G(Z_1), X_1) \\ L(G(Z_2), X_2) \\ L(G(Z_3), X_3) \\ \vdots \\ L(G(Z_k), X_k) \end{bmatrix} \tag{10}$$

If the final loss in the backward computation is the mean (or the sum) of all the losses, then since the weights of the network are not being updated, all the points could be updated simultaneously.

$$\mathbf{Z} = \mathbf{Z} - \alpha \nabla_{\mathbf{Z}} \frac{1}{k} \sum_{i=1}^{k} (L(G(Z_i), X_i)) \tag{11}$$

Hence, the reconstruction operations can be executed in parallel for multiple data points. When combined with GPU acceleration, our experiments demonstrate a significant reduction in computation time. Refer to Figure 3 for an illustration.

Anomaly Score

After training the GANs, any input sub-sequence X with the shape $(1, 1, w)$ can be inverted as Z and reconstructed as $G(Z)$ of the same shape using Algorithm 1. Multiple sub-sequences could be reconstructed in parallel. The reconstruction loss is measured using Soft-DTW, which also provides a direct way of obtaining an anomaly score. Our objective goes beyond identifying if there is an appropriate

input to generate a sample resembling X. We also want to ensure that this input can be generated by sampling from the latent space distribution p_z which is Gaussian-centred at the origin. The combined anomaly score is presented below.

$$\text{Anomaly score}(X) = \alpha * \text{Soft-DTW}(X, G(Z)) + \beta * \|Z\|_2 \tag{12}$$

Where α and β regulate the influence of each subpart. The input sub-sequence is marked as anomalous if the anomaly score exceeds a certain threshold.

Anomaly Detection Using Kernel Density Estimates

Due to the sequential nature of anomalies, a direct comparison between the ground truth and the prediction becomes difficult. In this regard, we use a similar method proposed in the prior work done by Gu et al. (Gu and Jazizadeh 2022). The set of time windows (sub-sequences) identified as an anomaly by the model are to be first converted into anomalous timestamps to make a comparison with the ground truth. This is achieved by performing the following steps.

1. Select a test segment for evaluation.
2. Create overlapping sub-sequences of window size w.
3. Produce anomaly scores for each of the sub-sequences.
4. For sub-sequences with anomaly scores greater than a set threshold, mark the timestamp corresponding to the middle of each sub-sequence as the critical points.
5. Use Kernel Density Estimation (KDE) (Parzen 1962) to create a distribution over the test segment of the critical points. Scale the density to lie between 0 and 1.
6. Find points of the scaled KDE above a certain height and mark those timestamps as the predicted anomalies.

Refer to Figure 2 for an illustration of anomaly detection over a test segment.

Model Evaluation

Although the labels are pointwise, time series anomalies can span a range of time, making it challenging to precisely define the onset of an anomaly. Thus, as long as the model predicts in the vicinity of the anomaly label, it should not be penalized. Introducing some tolerance (r_t) in performance calculations accounts for this consideration. To evaluate the model, we adopt precision, recall and F1 score. True Positives (TP), False Negatives (FN), and False Positives (FP) calculations are adjusted with the introduction of tolerance.

$$\begin{aligned} TP &: |d - p_{\text{closest}}| \leq r_t, \\ FN &: |d - p_{\text{closest}}| > r_t, \\ FP &: |p - d_{\text{closest}}| > r_t \end{aligned} \tag{13}$$

Where d represents the location of a labelled ground truth anomaly, and p_{closest} denotes the nearest anomaly prediction. Ground truth anomalies with a predicted anomaly in their proximity are considered TP, while those without any predicted anomaly nearby are classified as FN. FP refers to an anomaly prediction that does not correspond to any real defect in its vicinity, with p representing the location of the predicted anomaly and d_{closest} representing the nearest actual annotated anomaly location.

Table 1: Performance metrics on different buildings

Building No.	$r_t = 12$			$r_t = 24$		
	Rec.	Prec.	F1	Rec.	Prec.	F1
1	0.983	0.988	0.986	0.988	0.994	0.991
2	0.629	0.991	0.769	0.697	0.992	0.819
3	0.768	0.663	0.712	0.898	0.792	0.842
4	0.668	0.725	0.695	0.747	0.827	0.785
5	0.936	0.876	0.905	0.936	0.978	0.956
6	0.618	0.870	0.723	0.751	0.903	0.820
7	0.882	0.918	0.900	1.000	1.000	1.000
8	0.674	0.796	0.730	0.838	0.856	0.847
9	0.791	0.772	0.781	0.917	0.890	0.903
10	0.692	0.759	0.724	0.835	0.792	0.813
11	0.570	0.833	0.677	0.640	0.936	0.760
12	0.627	0.941	0.753	0.637	1.000	0.778
13	0.841	0.891	0.865	0.869	0.959	0.912
14	0.945	0.937	0.941	0.945	1.000	0.972
15	0.817	0.810	0.813	0.846	0.815	0.830
Ave	0.763	0.851	0.798	0.836	0.915	0.869

Experiments and Results

The implementation of 1D-DCGAN is done using PyTorch, a popular deep-learning library in Python. In our experimental setup, we use the Adam optimizer for training both networks, with the beta set to 0.5 and a learning rate of 0.0002. The WGAN's "ncritic" parameter is configured to 5, and the clipping value is set at 0.01. Additionally, a batch size of 128 is used. Our analysis focuses on sub-sequences with a window size of 48 and a latent space of 100 dimensions. The Gan model is trained for 200 epochs. The evaluation hyperparameters for anomaly scores are adjusted while observing the model performance on the test set.

We evaluate our methodology on each energy time series separately. The time series for each building is divided into train-test segments. The sub-sequences are created using a window size of 48 hours. The training segments are distinct from the test segments. For the evaluation, we set the hyperparameters separately for each building: 1. α, 2. β, 3. anomaly score 'threshold', and 4. KDE 'min_height'. Table 1 compares the performance metrics of the anomaly detection model for different tolerance (r_t) values.

Reconstruction done in parallel significantly decreases the testing time. Figure 3 demonstrates the time (measured in seconds) required to reconstruct data across different batch sizes on the same machine. When reconstructing one data point at a time (sequential reconstruction), the time taken displays a linear relationship with the batch size. In contrast, when reconstruction is performed in parallel, the time taken remains relatively constant across different batch sizes.

We conduct a comparative analysis of the average performance across all buildings using different reconstruction losses. Commonly, Euclidean distance is employed. However, our study confirms the effectiveness of Soft-DTW as a suitable alternative differentiable reconstruction loss. Since we are reconstructing multiple data points in parallel in a batch, our model operates in two modes: Active Statistics Mode (ASM) and Static Statistics Mode (SSM). In the ASM mode, the generator model's BatchNorm layers utilize the mean and variance of the current batch while performing

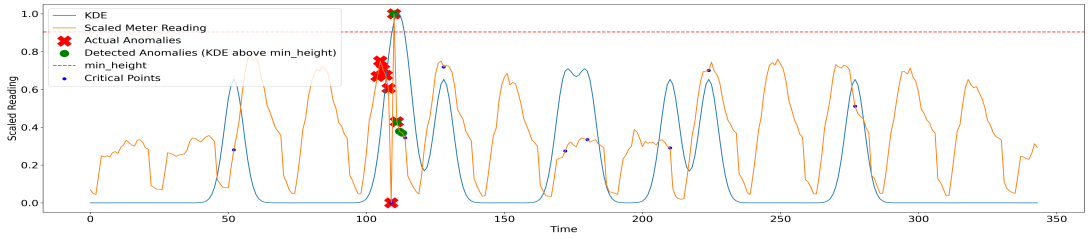

Figure 2: A test time series segment from a building with timestamps indexed starting from 0 along the x-axis. The orange line represents the scaled actual meter reading, and the blue line represents the scaled KDE. Actual annotated anomalies are indicated by red crosses, while our method marks anomalies with green dots, considering only KDE values above a threshold.

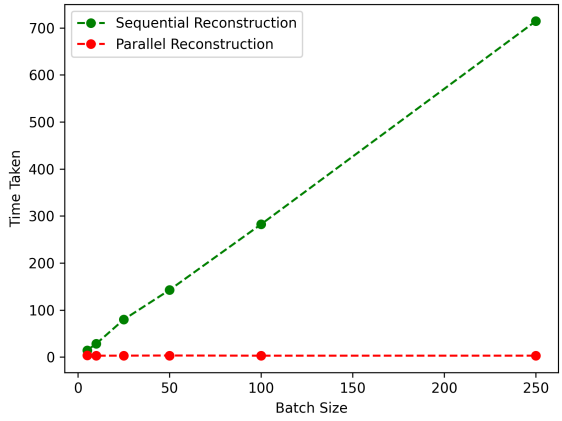

Figure 3: Comparison between time taken for reconstruction done using Mean Square Error loss sequentially and in parallel for different batch sizes.

Table 2: Average Performance Metrics for different choices of reconstruction loss

Reconstruction	$r_t = 12$			$r_t = 24$		
	Rec.	Prec.	F1	Rec.	Prec.	F1
Soft-DTW (ASM mode)	0.763	0.851	0.798	0.836	0.915	0.869
Soft-DTW (SSM mode)	0.785	0.832	0.777	0.843	0.879	0.834
Euclidean (SSM mode)	0.773	0.789	0.751	0.851	0.853	0.824
Euclidean (ASM mode)	0.516	0.520	0.476	0.603	0.582	0.548

reconstruction. In contrast, in the SSM mode, they rely on the estimated population statistics gathered during training. This distinction can significantly impact the model's performance. We test the model using both reconstruction losses, and the results are detailed in Table 2. Notably, Soft-DTW proves to be the more effective method in both scenarios, achieving F1 scores of 0.869 and 0.834. This performance is notably better than that of the Euclidean loss, which achieves F1 scores of 0.824 and 0.548. Interestingly, turning off the evaluation mode enhances the results specifically when using Soft-DTW.

In this study, we also benchmark our model against an autoencoder. Specifically, we employ a 1-D Convolutional

Table 3: Average performance metrics across 15 buildings

Model	$r_t = 24$ (Tolerance)		
	Precision	Recall	F1
GAN (Soft-DTW)*	0.915	0.836	0.869
GAN (Euclidean)	0.853	0.851	0.824
1-D CNN-Autoencoder	0.887	0.840	0.829
LOF	0.305	0.815	0.414

**Best performing model with Soft-DTW reconstruction*

Neural Network (CNN) Autoencoder, which is designed with an encoder that mirrors the generator component of our Generative Adversarial Network (GAN), and a decoder that serves as its symmetrical counterpart. Additionally, we evaluate the performance of the Local Outlier Factor (LOF) model. All the models are evaluated on each of the buildings separately and then the average performance metrics are calculated and presented in Table 3. Our GAN-based model outperforms all others, particularly when employing Soft-DTW as the loss function, achieving an F1 score of 0.869 with a 24-hour tolerance. The 1D-CNN autoencoder ranks as the second-best model, securing an F1 score of 0.829, closely followed by the GAN model with Euclidean reconstruction, which scores 0.824 in F1. In contrast, the non-deep learning-based method, Local Outlier Factor (LOF), shows a significantly lower performance, with an F1 score of 0.414, indicating its ineffectiveness in detecting sequential anomalies.

The source code and additional materials related to this study are available in the GitHub repository. [1]

Conclusion

The paper systematically demonstrates how to train the GAN on normal data, perform gradient-based inverse mapping to reconstruct query samples, and use the reconstruction error as an anomaly score to generate critical points on a given time series segment and create an anomaly distribution on the segment using KDE. Tolerance is introduced

[1] https://github.com/HardikPrabhu/Energy-Time-series-anomaly-detection

for comparison with models that detect anomalies at the subsequence or pointwise level. This makes the evaluation more flexible.

By incorporating parallel computation, we effectively mitigate the performance bottleneck, a commonly cited drawback of this approach. This enhancement opens avenues for more studies on various components in an isolated context. For example, refining the generative process through improved model architecture, optimizing the choice of reconstruction loss, and enhancing the anomaly scoring mechanism. Additionally, there is potential for exploring different approaches to benchmark sequential anomalies, which in most of the datasets, are annotated solely by timestamps (pointwise).

One limitation of our method, which involves using overlapping sub-sequences to identify critical points for a time segment, is the possibility of lower Kernel Density Estimation (KDE) values for the beginning and end parts. This reduction is attributed to the overlapping method used in deriving sub-sequences, leading to a lesser relative frequency of critical points in these specific areas. To address this issue, we could add a window length's worth of data at both the start and end of a test segment, borrowing from the preceding and subsequent segments, respectively.

While the results showcased in this study are preliminary, they underscore the potential and promise of the approach. However, it is crucial to recognize the need for more rigorous benchmarking. This entails comprehensive comparisons across a broader range of buildings and an extensive variety of models. Such thorough evaluations, which extend beyond the scope of our current work, will provide deeper insights and reinforce the findings presented in our paper. The limitations identified will be addressed in an expanded version of this paper.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223. PMLR.

Bashar, M. A.; and Nayak, R. 2020. TAnoGAN: Time series anomaly detection with generative adversarial networks. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1778–1785. IEEE.

Berndt, D. J.; and Clifford, J. 1994. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining*, 359–370.

Brophy, E.; Wang, Z.; She, Q.; and Ward, T. 2021. Generative adversarial networks in time series: A survey and taxonomy. *arXiv preprint arXiv:2107.11098*.

Coluccia, A.; D'Alconzo, A.; and Ricciato, F. 2013. Distribution-based anomaly detection via generalized likelihood ratio test: A general maximum entropy approach. *Computer Networks*, 57(17): 3446–3462.

Cuturi, M.; and Blondel, M. 2017. Soft-dtw: a differentiable loss function for time-series. In *International conference on machine learning*, 894–903. PMLR.

Di Mattia, F.; Galeone, P.; De Simoni, M.; and Ghelfi, E. 2019. A survey on gans for anomaly detection. *arXiv preprint arXiv:1906.11632*.

Geiger, A.; Liu, D.; Alnegheimish, S.; Cuesta-Infante, A.; and Veeramachaneni, K. 2020. Tadgan: Time series anomaly detection using generative adversarial networks. In *2020 IEEE International Conference on Big Data (Big Data)*, 33–43. IEEE.

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

Gu, Y.; and Jazizadeh, F. 2022. DEGAN: Time Series Anomaly Detection using Generative Adversarial Network Discriminators and Density Estimation. *arXiv preprint arXiv:2210.02449*.

Gulati, M.; and Arjunan, P. 2022. LEAD1. 0: a large-scale annotated dataset for energy anomaly detection in commercial buildings. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 485–488.

Himeur, Y.; Ghanem, K.; Alsalemi, A.; Bensaali, F.; and Amira, A. 2021. Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives. *Applied Energy*, 287: 116601.

Katipamula, S.; and Brambley, M. R. 2005. Methods for fault detection, diagnostics, and prognostics for building systems—a review, part I. *Hvac&R Research*, 11(1): 3–25.

Li, D.; Chen, D.; Jin, B.; Shi, L.; Goh, J.; and Ng, S.-K. 2019. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International conference on artificial neural networks*, 703–716. Springer.

Parzen, E. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3): 1065–1076.

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

Roth, K. W.; Westphalen, D.; Feng, M. Y.; Llana, P.; and Quartararo, L. 2005. Energy impact of commercial building controls and performance diagnostics: market characterization, energy impact of building faults and energy savings potential. *Prepared by TAIX LLC for the US Department of Energy. November. 412pp (Table 2–1)*.

Schein, J.; Bushby, S. T.; Castro, N. S.; and House, J. M. 2006. A rule-based fault detection method for air handling units. *Energy and buildings*, 38(12): 1485–1492.

Schlegl, T.; Seeböck, P.; Waldstein, S. M.; Schmidt-Erfurth, U.; and Langs, G. 2017. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, 146–157. Springer.

Zhu, G.; Zhao, H.; Liu, H.; and Sun, H. 2019. A novel LSTM-GAN algorithm for time series anomaly detection. In *2019 prognostics and system health management conference (PHM-Qingdao)*, 1–6. IEEE.