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

Motivation

The core strength of Generative Adversarial Networks (GANs) lies in learning and mimicking complex data distributions, which allows for generating synthetic data points indistinguishable from the original. GANs could utilize this ability to detect anomalies. This approach differs from traditional likelihood-based models. GANs learn a mapping from a simple prior in latent space to the data distribution and do not explicitly model the data distribution. Popular methods involve recreating the original data from noise and then using this reconstruction to calculate an anomaly score by comparison.

Our paper focuses on performing gradient descent in the input space to reverse the generator's mapping. This process aims to find a specific noise that can effectively reconstruct the data. A commonly known drawback of this approach is the slow test runtime, which we aim to overcome by performing gradient descent operations in parallel. This allows the inversion of multiple data points to noise space, which are then reconstructed by passing the noise vectors through the generator. The reconstructed data is then compared with the original data points to detect anomalies.

Contributions

- 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 [?] 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.

Problem Statment and Dataset

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 LEAD1.0 dataset [?] 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.

Preprocessing

Train-test segments: First, the time series data for individual buildings' electricity consumption is partitioned into segments. We divide each series into contiguous, non-overlapping time segments. The segments contain pointwise anomaly annotations. The segments which have no anomalous points are used in training. Each segment is normalized to be in the range [-1,1].

Model Input: A segment of time series data is 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.

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

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

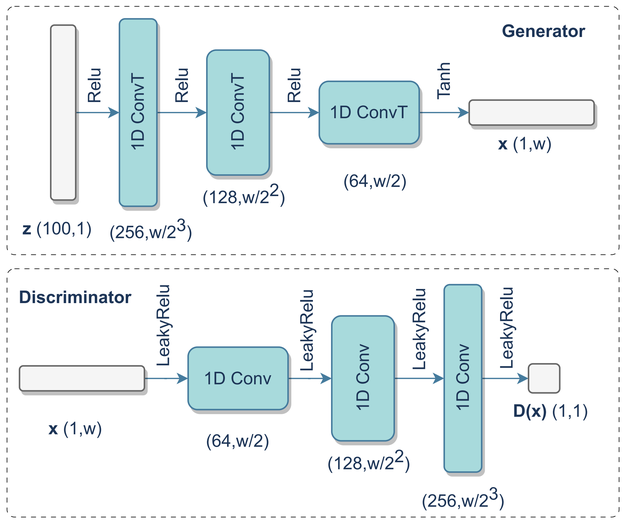

Figure 1. 1-D DCGAN architecture used for anomaly detection

The generator and discriminator in the DCGAN architecture, as shown in Figure 1, each consists of three symmetrical hidden layers. The generator upsamples an input noise vector into a 1D time series output using convolutional transpose layers, starting with 100 input channels and progressing through layers and *Tanh* activation at the end.

Reconstruction Loss

Soft-DTW : A Differentiable Loss for Reconstruction

We use Soft-DTW [?] 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:

$$\begin{aligned} D[i,j] =& \text{softmin}_\gamma \left(D[i-1,j], D[i,j-1], D[i-1,j-1]\right) \\ &+ d(x_i, y_j) \end{aligned} \tag{2}$$

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 [?]. The Soft-DTW distance between the two sequences X and Y is found at $D[n, m]$.

Parallel Computation

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.

$$L(G(\mathbf{Z}), \mathbf{X}) = \begin{bmatrix} L(G(Z_1), X_1) \\ L(G(Z_2), X_2) \\ L(G(Z_3), X_3) \\ \vdots \\ L(G(Z_k), X_k) \end{bmatrix} \tag{3}$$

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

$$\mathbf{Z} = \mathbf{Z} - \alpha \nabla_{\mathbf{Z}} \frac{1}{k} \sum_{i=1}^{k} (L(G(Z_i), X_i)) \tag{4}$$

Reconstruction and Anomaly Score

After training the GAN network, 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 gradient descent. 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. 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.

$$\text{Anomaly score}(X) = \alpha * \text{Soft-DTW}(X, G(Z)) + \beta * \|Z\|_2 \tag{5}$$

Where α and β regulate the influence of each subpart.

Anomaly Detection Using KDE

The anomaly is detected at the subsequence level. To align with annotated timestamp data, these detections are then translated into specific anomalous timestamps. In this regard, we use a similar method proposed in the prior work done by Gu et al. [?]. 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) 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.

Evaluation

To evaluate, True Positives (TP), False Negatives (FN), and False Positives (FP) calculations are adjusted with the introduction of tolerance.

$$\begin{aligned} TP &: |d - p_{\text{closest}}| \leq r_t, \\ FN &: |d - p_{\text{closest}}| > r_t, \\ FP &: |p - d_{\text{closest}}| > r_t \end{aligned} \tag{6}$$

Where d is the ground truth anomaly location, and p_{closest} is the nearest predicted anomaly. Ground truth anomalies near a prediction are True Positives (TP), and far are False Negatives (FN). False Positives (FP) occur when a prediction (p) has no real anomaly nearby, with d_{closest} as the nearest true anomaly.