*IJECBE* (2024), **2**, **3**, 284–301
Received (1 June 2024) / Revised (10 July 2024)
Accepted (11 July 2024) / Published (30 September 2024)
https://doi.org/10.62146/ijecbe.v2i3.56
https://ijecbe.ui.ac.id
ISSN 3026-5258

**International Journal of Electrical, Computer and Biomedical Engineering**

RESEARCH ARTICLE

# Anomaly Detection in Imbalance Secure Water Treatment Dataset Using LSTM-DC-Wasserstein Generative Adversarial Network with Gradient Penalty

J. M. Kevin* and N. Raharya

Department of Electrical Engineering, Universitas Indonesia, Depok, Indonesia
*Corresponding author. Email: jonathan.marshell@ui.ac.id

**Abstract**

In modern industrial systems, particularly with the advancement of the Internet of Things (IoT), industry players can record machine and system data for comprehensive analysis. This capability is crucial for detecting anomalies and taking necessary corrective actions. However, it is common for manufacturers to lack recorded anomaly datasets, especially for newly operational systems. In this paper, we develop a model to detect anomalies in an imbalanced dataset from the Secure Water Treatment (SWaT) system. The performance of the proposed model is compared with previous works, demonstrating significant improvements in anomaly detection capabilities where it achieves accuracy of 0.9546, precision of 0.9086, recall of 0.6654, and F1 score of 0.7681

**Keywords:** wasserstein gan, deep convolutional neural network, long-short term memory, anomaly detection, multivariate time series

## 1. Introduction

The Internet of Things (IoT) is accelerating the digitization of industrial control systems (ICS), leading to significant improvements in the efficiency of industrial processes. Cyber-Physical Systems (CPS), a key component of ICS, serve dual purposes: they enable precise control and facilitate extensive data collection for subsequent analysis. The integration of advanced systems—such as sensors for monitoring machine performance [1], enhanced data acquisition and storage technologies [2],

and cutting-edge artificial intelligence (AI) algorithms [3] has made the development of data-driven analytics increasingly feasible. By combining human creativity with machine-generated analytics, organizations can achieve superior outcomes in problem-solving, strategic planning, and operational enhancements. This synergistic approach leverages the strengths of both human insight and automated data processing for optimal data-driven results. [4].

The implementation of data-driven analytics has been explored from various perspectives, one of which is anomaly detection. An ICS generates a large volume of multivariate time series data, which may encompass both "normal" system behavior and "anomalous" behavior caused by factors such as sensor faults, human errors, or cyber attacks [5]. Various methods have been investigated to identify the most effective approach for detecting anomalies in datasets. Filovonov *et al.* in [6] and [7] developed a neural network-based forecasting approach for anomaly detection in their studies, while Matteson *et al.* proposed a non-parametric change point detection method for the same purpose [8]. Goodfellow *et al.* proposed Generative Adversarial Networks [9] to handle image related tasks. It is then enhance to handle task beside image [10], one of the is anomaly detection. The original model of GAN is powerful, but it is hard to train. So, there are improved model developed to stabilize the training, for instance from Arjovsky *et al.* [11] by improving GAN's loss metric and from Gulrajani *et al.* [12] by adding gradient penalty to avoid undesired behaviour during training. Improved performance of GAN can be achieved by combining it with other deep models, for example with Long-Short Term Memory (LSTM) [13] by Xu *et al.* Generative DL model is used for fault detection, especially when the data has imbalance ratio between "normal" and "anomalous" data, for instance in [14], [15] and [16].

Imbalanced datasets are a common issue in industrial environments. There are two primary approaches to address this imbalance: data-level methods and algorithm-level methods [17]. At the data level, one popular oversampling technique is Synthetic Minority Over-sampling Technique (SMOTE) [18] which operates based on the k-nearest neighbor method. Recently, researchers have begun using Generative Adversarial Networks (GANs) for data oversampling, demonstrating promising results in generating synthetic data or images [19]. The application of GANs for oversampling has been explored in various domains, including crash prediction [18], credit card fraud detection [15] and motor fault diagnosis [20]. At the algorithm level, methods are designed to adjust classifiers to better handle imbalanced data [17]. For example, Wu *et al.* proposed the Easy-SMT method to address imbalanced datasets from an algorithmic perspective [21].

We address the challenge of finding anomaly in imbalanced industrial data by proposing a Long Short-Term Memory - Deep Convolutional Wasserstein GAN with Gradient Penalty (LSTM-DC-WGAN-GP). The Generator is trained to generate synthetic time series data that reflects normal system behavior and Critic (Discriminator) is trained to distinguish feature of real data and synthetic data. The synthetic data generated by the trained model will be compared to real testing data and its result will be classified based on a combination of reconstruction error and feature matching error to detect anomalies. The SWaT dataset, as described in [22], will be

used for training and testing our model.

The main contributions of this paper are as follows: (1) The design of an LSTM-DC-WGAN-GP model to generate multivariate synthetic time series data for detecting anomalies in imbalanced datasets, and (2) A comparative analysis demonstrating that the developed model outperforms other existing models and algorithms in terms of performance.

## 2. Anomaly Detection

In industrial environments, anomalies are defined as deviations in recorded data that differ from the usual behavior of tools, machines, or equipment [23]. Detecting anomalies is essential for manufacturers to maintain production efficiency and prevent unnecessary downtime. Anomaly detection involves identifying regions or time periods where data deviates from the norm. However, achieving satisfactory results in anomaly detection presents several challenges, such as unclear boundaries between normal and anomalous behavior, indistinct anomalies resulting from malicious actions, evolving patterns of normal and anomalous data, the need for labeled data for training and validation, varying anomaly behaviors across different domains, and noise in recorded data [23]. Additionally, class-imbalanced data, where normal condition data is more prevalent than anomalous data, is a common issue in real-world cases [24]. The Secure Water Treatment (SWaT) dataset from iTrust, Center for Research in Cyber Security at the Singapore University of Technology and Design [22] is a prime example of such imbalanced data.

### *2.1 Imbalance Data*

Having a balance data is important factor in building an anomaly detection model based on machine learning. Imbalance data cause lower accuracy of generated model [25]. But, it turns out its very common for industry player not to have a balance recorded data. This topic has been addressed as one obstacle in training an anomaly detection model [23]. As this is a common situation in CPS, Goh *et al.* present a realistic dataset from a fully operational, scaled-down water treatment plant [22].

The dataset comprises six main processes, each involving 51 parameters, a combination of 25 sensors, and 26 actuators [5]. The SWaT dataset is divided into two subsets: training data and testing data. All parameters are recorded once per second over an 11-day period. The training data covers seven days of normal operations, while the remaining six days include a combination of normal and attacked conditions. Notably, the data collection process began from an empty state, and it took approximately five hours for the system to stabilize [22].

There are 4 types of attacked recorded in SWaT, they are:

1. Single Stage Single Point (SSSP): single attack on one point in the system. There are 23 recorded attacks. For example: attack on AIT-202 where its value set to 6 suddenly.
2. Single Stage Multi Point (SSMP): single attack aiming to multiple points in the system. There are 6 recorded attacks. For example: attack on MV-101 and LIT-101 where status of MV-101 kept on and value of LIT-101 set to 700mm.

3. Multi Stage Single Point (MSSP): similar to SSSP, but this attack performed on multiple stages. There are 4 recorded attacks. For example: attack on FIT-401 and AIT-502 where value of FIT-401 set to 0.5 and value of AIT-502 set to 140 mV.
4. Multi Stage Multi Point (MSMP): similar to SSMP, but this attack performed on more than one stages. There are 3 recorded attacks. For example: attack on UV-401, AIT-502 and P-501 where UV-401 is stopped, value of AIT-502 set to 150 and force P-501 to remain on.

For more details on attack scenarios, attack points, impacts, and other information, refer to Table I in [26] and [22]. Based on the attack data, three distinct attack behaviors were identified:

1. Sudden Changes Beyond Normal Range: Measurements abruptly deviate beyond the recorded normal data range. For instance, in the LIT-301 attack, the normal data range is between 132.8185 and 1014.724, but during the attack, the recorded data spiked to 1200.
2. Sudden Changes Within Normal Range: Measurements abruptly change but remain within the recorded normal data range. An example is the FIT-401 attack, where the normal data range is between 0 and 1.747862. During the attack, the data was set to 0.7 for a while before dropping to 0.
3. Gradual Changes: Measurements change gradually over time. For instance, in the LIT-101 attack, the recorded data increased by 1mm per second during the attack.

The dataset includes 36 attack scenarios, though not all were successful. According to the shared dataset from iTrust [22], there is a class imbalance, with only 5.7% of the data being anomalous.

### 2.2 Generative Adversarial Networks (GAN) Algorithm and Architecture

Generative Adversarial Networks (GANs), initially proposed by Goodfellow *et al*., are designed for image sample generation. In a GAN, the Generator network denoted as Generator $G$ is trained to map data from a latent space or noise $z$ and create synthetic data samples denoted as $p_g$. The Discriminator network, denoted as $D$ is trained to output the probability that a given data sample comes from the real data distribution $x$ rather than the synthetic distribution $p_g$. The interplay between $D$ and $G$ forms a two-player minimax game, with the objective function $V(G, D)$ defined as follows [9]:

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

where $x$ represents the real data, $z$ represents the synthetic data, $D(x)$ represents the probability $x$ came from real data, $D(G(z))$ represents the probability that $z$ came from synthetic data. This model trained to maximize the probability of D assigning correct labels to the data [9].

Generative Adversarial Networks (GANs) have proven successful not only for image generation, as mentioned in the seminal work by [9], but also for video prediction [27]. Researchers have expanded the capabilities of GANs to handle various use cases. Schlegl *et al.* employed GANs for anomaly detection using unsupervised medical image data [16], while Shafqat *et al.* utilized GANs to over-sample minority class data for recommendation systems [24].

However, training GAN models to reach Nash equilibrium remains a challenge [28]. Weight updates pose one of the difficulties. Weight clipping is a common approach to ensure that the Discriminator's weights $W$ lie within a compact space. But, this method has drawbacks [11]. If the clipping parameter is large, training time increases, while using a small clipping parameter can lead to gradient vanishing issues. To improve stability during training, various techniques have been introduced by Salimans *et al.* [28], for example: Feature matching (instead of directly maximizing the output of the Discriminator, the Generator aims to generate data that matches the statistics of real data) and virtual batch normalization (Input examples (x) are normalized based on statistics collected from a reference batch of examples).

In [11], a new divergence proposing a new divergence to be minimized during the training of GAN, that is Earth-Mover (or called Wasserstein-1) distance, denote as $W(q, p)$. It is the cost to transform distribution of q into distribution of p. Also, Discriminator role is now called by Critic, as it is not function as classifier anymore. Based on this adjustment, WGAN lost function to obtain [12]:

$$\min_{G} \max_{D \in \mathbf{D}} E_{x \sim \mathbf{p}_r}[D(x)] - E_{z \sim \mathbf{p}_g}[D(z)] \tag{2}$$

where $D$ is the set of 1-Lipschitz function and $P_g$ is the distribution defined by $z = G(z), z \sim p(z)$. In [12], Gulrajani *et al.* proposed an alternative to enforce the Lipschitz constraint by adding gradient penalty portion on the loss function of WGAN. The objective function of WGAN-GP is [12]:

$$L = E_{z \sim \mathbf{p}_g}[D(z)] - E_{x \sim \mathbf{p}_r}[D(x)] + \lambda E_{z \sim \mathbf{p}_z}[(\|\nabla_z D(z)\|_2 - 1)^2] \tag{3}$$

where λ is the weight decay of gradient penalty, $(\|\nabla_z D(z)\|_2 - 1)^2$ is the function to enforce Lipschitz constraint [12].

New proposed gradient penalty on the loss function helps Critic in the training process, as the optimal Critic forms straight lines with gradient norm 1 when connecting two points from $P_r$ and $P_g$.

Beside various algorithm of GAN, researcher also explore architecture of Discriminator and Generator. For example, Bashar *et al.* used 3 layer of LSTM network with depth 3 and 100 hidden units for Generator and 1 layer of LSTM with depth 1 for Discriminator [29]. Gulrajani *et al.* used Deep Convolutional GAN (DCGAN) for both Discriminator and Generator with various size [12]. Another enhancement of GAN by Lu *et al.* in [30] is cooperative GAN. In usual GAN algorithm, Discriminator and Generator are against each other. In his model, Discriminator and Generator are sharing the same purpose. It is founded that this appoarch help to stabilize the training.

### 2.3 GAN Based Anomaly Detection

The use of Generative Adversarial Networks (GANs) in anomaly detection has gained popularity. There are several approaches to leveraging GANs for anomaly detection. First, synthetic data generation. GANs can generate synthetic data to address imbalanced datasets by creating synthetic minority class data. This approach is demonstrated in credit card fraud detection by Fiore *et al.* [15], who used GANs to generate synthetic anomalous transactions. Second, Discriminator as Anomaly Detector: The Discriminator in a GAN, trained on normal data, can produce distinct values when fed anomalous test data, as described by Jiang *et al.* in [14]. However, this method requires careful assessment because the Discriminator is originally trained to differentiate between real and artificial data. This training helps the Generator create data resembling real signals from the latent space, but it does not inherently detect anomalies. Third, combining Discriminator and Generator. Schlegl *et al.* implemented this approach in Anomaly GAN (AnoGAN) [16]. To identify anomalies, they combined residual loss (the visual dissimilarity between generated data and actual test data) with Discriminator loss (the feature extraction loss between real and generated data). This method is useful when the training data consists of normal data. However, in practice, it is less straightforward than using the Discriminator alone, as the output of the Generator is not an anomaly score.

## 3. Proposed Design and Methodology

In this section, we present the background of the proposed anomaly detection architecture based on LSTM-DC-WGAN-GP and its application in detecting anomalies in an imbalanced SWaT dataset. This model utilizes the WGAN-GP algorithm, a stable GAN method, to train the Critic and Generator, which are then used to identify anomalies in the dataset. The Critic and Generator architectures are built using a combination of long-short term memory (LSTM) and deep convolutional neural network (DC) methods.

LSTM was first introduced by Hochreiter *et al.* to improve the performance of recurrent neural networks (RNNs) by addressing two main issues in the back-propagation process: high error signals causing oscillation in weight updates and vanishing error signals leading to no changes in weights [31]. LSTM has been successfully applied to time series data, demonstrating its capability in various applications such as language modeling, speech-to-text transcription, machine-based translation [32], predicting the remaining usage life for UVLEDs [33], and predicting traffic volume [34]. Given its ability to handle time series data, LSTM was chosen for the SWaT dataset, which also comprises time series data. Detecting anomalies in SWaT requires a model that can identify continuity in the data of each feature, a task well-suited to LSTM.

The second component of our anomaly detection model is DC. DC, a subset of deep learning, is capable of recognizing patterns in datasets [35]. Although DC is predominantly applied to image tasks, its ability to study data patterns makes it applicable to anomaly detection, such as in the SWaT dataset. As previously mentioned in Section 2.1, the anomaly behavior in SWaT is not always a sudden drastic change to a value outside the normal range; it can also be a gradual change in the recorded data

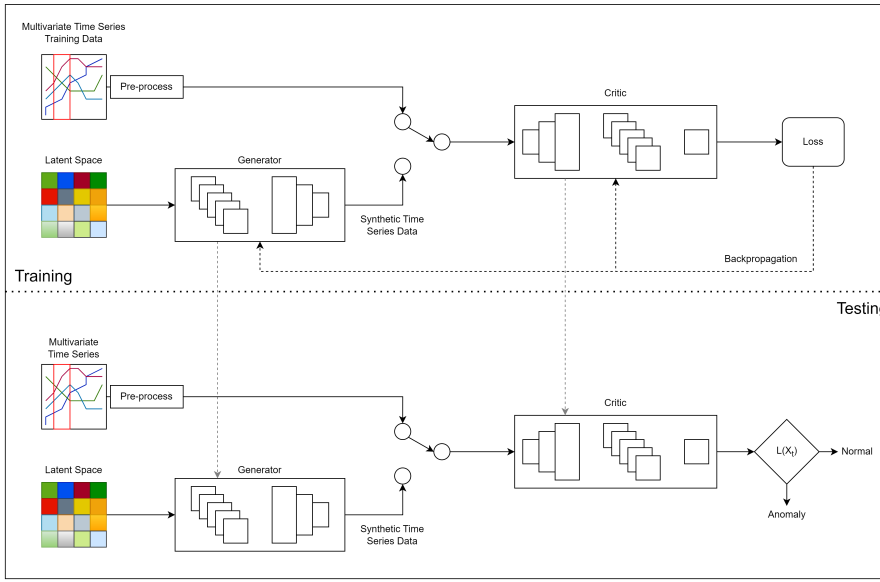


**Figure 1.** Process Flow

(a change in pattern). During the learning phase, DC will study the normal patterns in the data to recognize any deviations.

We implemented the LSTM-DC-WGAN-GP using PyTorch, a popular deep-learning library in Python. The experiment flow is shown in Figure 1. In summary, the process begins with pre-processing both the training and testing data. The training data is then used to train the LSTM-DC-WGAN-GP model. After training, the model is applied to the testing data to generate anomaly detection results. These results are then compared with those obtained using other methods and algorithms based on standard metrics (Precision, Recall, and F1 scores).

### *3.1 Data Preprocessing*

Training data starts being recorded from an empty state and requires 5 hours to stabilize [22]. Consequently, the first 21,600 rows are removed, leaving 475,200 rows of stable data. This data is then subdivided into smaller time series segments, with each window having a length of 30 seconds. Within these time windows, the median value of the data is taken as a representative measure. This approach is applied to both training and testing data, reducing the dataset to 15,840 rows for training. The data is then batched, with each batch having a shape of (51, 20), where 51 represents the number of features and 20 represents the time length, corresponding to 600 seconds of real data. The window capture is shifted by 10 to capture the next batch. As a final step, each data feature is normalized to a range of 0 to 1 using the min-max method. This procedure is applied to the testing data as well, with the difference being that for testing data, each window length is 5 seconds, reducing the dataset from 449,919

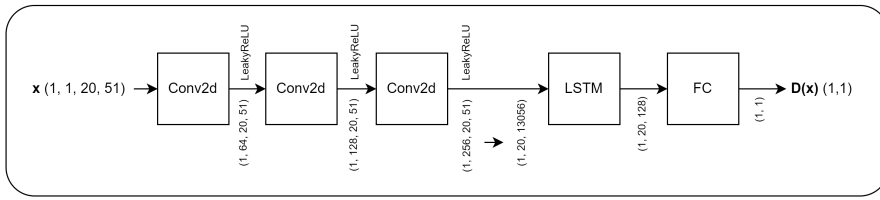


**Figure 2.** Critic Structure

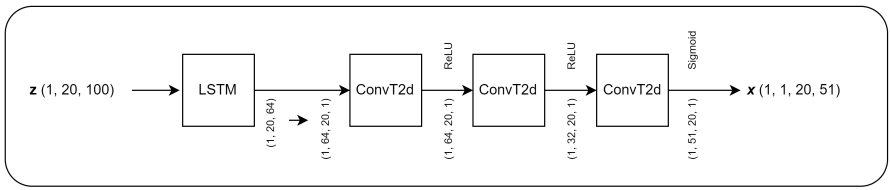


**Figure 3.** Generator Structure

rows to 89,983 rows. The batch size for the test data remains (51, 20), where 20 now represents 100 seconds of real data. The window capture is shifted by 20 to capture the next batch, and the same min-max values used for normalizing the training data are applied to the test data, under the assumption that the data range in the training set covers all normal operations of the SWaT system.

### *3.2 System Architecture*

In this study, we use combination of LSTM and convolutions layer. For Critic, input data first connected to 3 2-dimension convolution layer. Each of this layer will duplicated the channels: 1 to 64 on first layer, 64 to 128 on second layer, 128 to 256 on third layer. This arrangement help model to understand features of the data. Convolution layer then followed by a LSTM layer. A LSTM layer help model to understand temporal dependencies as the data is time series [36]. At every convolution layer, spectral normalization is used. Spectral normalization applied in Critics to help to stabilize the training by normalizing the spectral norm of the weight matrices [37]. And, activation function LeakyReLU with negative slope of 0.02 added.

For Generator, latent space is feed into a LSTM layer, then followed with 3 layer of convolution transpose layer. First two convolution transpose layers followed by batch normalization to help model stabilize the training process and ReLU activation function. At the end of third convolution layer, a Sigmoid activation function is applied as the data need to be normalized to 0 to 1.

Initial weights for all layers are using normal distribution with range 0.0 to 0.02 and biases are using 0. Graph of Critic and Generator structure can be seen in Figure 2 and Figure 3.

On the system algorithm, we are proposing LSTM-DC-WGAN-GP approach,

**Algorithm 1** LSTM-DC-WGAN with gradient penalty. Parameters used: $m$ = 1582, epoch size = 100, $\alpha = 1e^{-5}$, $\beta_1 = 0.5$, $\beta_2 = 0.99$, $n_{\text{Critic}} = 5$, $\lambda = 10$

**Require:** Batch size $m$, epoch size, the Adam hyper-parameters $\alpha$, $\beta_1$, $\beta_2$, the number of Critic iterations per Generator iteration $n_{\text{Critic}}$, initial Generator parameters $\theta_0$, initial Critic parameters $w_0$, $\lambda$.

**for** $t = 1, ...,$ epoch **do**
  **for** $i = 1, ..., m$ **do**
    **for** $i = 1, ..., n_{\text{Critic}}$ **do**
      Sample real data $\mathbf{x}^{(i)}$
      Sample random noise $\mathbf{z}^{(i)}$
      Sample a random number $\epsilon \sim U[0, 1]$
      $\hat{\mathbf{x}}^{(i)} \leftarrow \epsilon \mathbf{x}^{(i)} + (1 - \epsilon) G_\theta(\mathbf{z}^{(i)})$
      $L_C^{(i)} \leftarrow D_w(\hat{\mathbf{x}}^{(i)}) - D_w(\mathbf{x}^{(i)}) + \lambda(\|\nabla_{\hat{\mathbf{x}}^{(i)}} D_w(\hat{\mathbf{x}}^{(i)})\|_2 - 1)^2$
      $w_C \leftarrow \text{Adam}(L_C^{(i)}, w_{C_t}, \alpha, \beta_1, \beta_2)$
    **end for**
    Sample random noise $\mathbf{z}^{(i)}$
    $L_G^{(i)} \leftarrow -D_w(G_\theta(\mathbf{z}^{(i)}))$
    $w_G \leftarrow \text{Adam}(L_G^{(i)}, w_{G_t}, \alpha, \beta_1, \beta_2)$
  **end for**
**end for**

where Critic and Generator will against each other in the training process. In the development of the LSTM-DC-WGAN-GP model, we follow a specific training procedure. Initially, a batch of real data is sampled from the dataset, which is then loaded into the Critic model alongside synthetic data generated by the Generator. To enforce gradient penalty, we sample random numbers between 0 and 1 to create interpolated data points between the real and synthetic data. The Critic's loss is computed based on its scores for real data, synthetic data, and interpolated data. This process is iterated $n_{\text{Critic}}$ times, with the Critic adjusting its weights during each iteration. Finally, the Generator updates its weights, and the entire process is repeated across multiple epochs More details on the algorithm and parameters can be found below in Algorithm 1.

### *3.3 Anomaly Detection Method*

Method used to detect anomaly in this study is a combination of 2 loss calculation. These loss calculation will process further the output value of Critic and also Generator. First is inspired by [16]. This method used Critic loss or feature mapping of Critic, which defined as:

$$\mathbf{L}_C(\mathbf{z}_\gamma) = 0.5 * (\mathbf{C}(\mathbf{x}_{query}) - \mathbf{C}(\mathbf{G}(\mathbf{z}_\gamma))^2 \tag{4}$$

MSE is used in calculating Critic loss. The trained Critic is not determined whether generated data fits to normal data, but instead it compares features of $\mathbf{x}_{query}$ with the generated data $\mathbf{G}(\mathbf{z}_\gamma)$. Critic is act more on feature extractor instead of classifier.

Second method is using Dynamic Time Warping (DTW) proposed by Berndt *et al.* in [38]. DTW is a method to evaluate similarities between two time series. DTW

**Algorithm 2** Soft-DTW

**Require:** $X$, $Y$, *smoothing* $\gamma \geq 0$, squared Euclidean distance $\delta$.
1: $r_{0,0} = 0$; $r_{i,0} = r_{0,j} = \infty$; $i \in [m]$, $j \in [n]$
2: **for** $j = 1, ..., n$ **do**
3: **for** $i = 1, ..., m$ **do**
4: $r_{(i,j)} = \delta(X_i, Y_j) + min^{\gamma}(r_{i-1,j-1}, r_{i-1,j}, r_{i,j-1})$
5: **end for**
6: **end for**

techniques is often used compare to general loss function such as mean squared error (MSE) when it comes to time series data. How DTW works is not directly measure distance between one value to another but its compares overall shapes of the data [39]. Cuturi *et al.* enhance DTW feature to be Soft-DTW. Given there are two time series data x = $(x_1, x_2, \ldots, x_n)$ and y = $(y_1, y_2, \ldots, m)$, the Soft-DTW distance is defined in Algorithm 2. As part of the algorithm, squared Euclidean distance is calculated with below formula [40]:

$$\delta(X, Y) = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{5}$$

where $x \in X$, $y \in Y$ and $n$ are data shape of X.

Meanwhile, soft minimum distance (notated as $min^{\gamma}$) from $(r_{i-1,j-1}, r_{i-1,j}, r_{i,j-1})$ is calculated with below formula [41]:

$$min^{\gamma}(r_{i-1,j-1}, r_{i-1,j}, r_{i,j-1}) = -\gamma((\max z) + log \sum_{i=1}^{3} e^{z_i - \max z}) \tag{6}$$

where $z_1 = -\frac{r_{i-1,j-1}}{\gamma}$, $z_2 = -\frac{r_{i-1,j}}{\gamma}$, $z_3 = -\frac{r_{i,j-1}}{\gamma}$, $\max z = \max\{-\frac{r_{i-1,j-1}}{\gamma}, -\frac{r_{i-1,j}}{\gamma}, -\frac{r_{i,j-1}}{\gamma}\}$, $\gamma$ is smoothing parameter.

Soft-DTW distance which notated as **D** between $X$ and $Y$ can be found at $r(m, n)$. In this research, Soft-DTW will be used to calculate distance between synthetic data generated by trained Generator with the actual testing data.

Both error cannot directly combined. There are several steps to combine those as proposed by Geiger *et al.* in [42]. First, normalized Loss of Critic $L_C$, then multiply it with Soft-DTW distance. as shown in function below:

$$\mathbf{L}(\mathbf{x}_t) = \|\mathbf{L}_C(\mathbf{x}_t)\| * \mathbf{D}(\mathbf{x}_t, \mathbf{G}(\mathbf{z}_\gamma)) \tag{7}$$

where $\mathbf{L}_C(\mathbf{z}_\gamma)$ is the Critic loss and $\mathbf{D}(\mathbf{x}_{query}, \mathbf{G}(\mathbf{z}_\gamma))$ is the soft-DTW distance between $\mathbf{x}_{query}$ and $\mathbf{G}(\mathbf{z}_\gamma)$.

After the anomaly score for each time step is calculated, threshold techniques are applied to identify anomalous sequences. This method requires two parameters: window size and step size, as mentioned in [42]. The window size refers to the sliding window used to compare thresholds, while the step size represents the window size

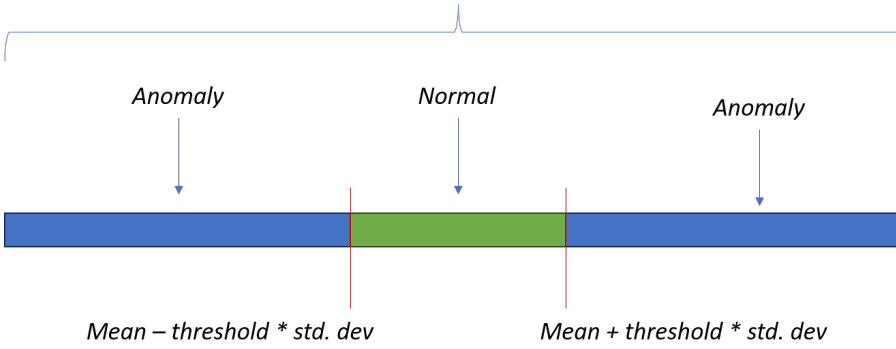


**Figure 4.** Anomaly Evaluation Method

per batch. In our research, we used a window size of 249 and a step size of 18. The value of 18 was determined based on our testing to achieve the best results, with trials conducted using step sizes of 14, 16, 18, 21, 24, 29, 37, 49, 74, and 149.

Within each sliding window, the anomaly score is compared with the mean and standard deviation of the sliding window. If the anomaly score falls within one standard deviation from the window's mean, it is categorized as normal data. This process is repeated for all anomaly scores. Figure 4 illustrates how anomalies are predicted.

### *3.4 Evaluation Metrics*

Standard metrics are being used to evaluate anomaly detection result of this model [43].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{11}$$

where TP is the correct detected anomaly (detection result labeled real anomaly correctly), FP is incorrect detected anomaly (detection result label anomaly for real normal data) and FN is incorrect detected normal (detection result label normal for real anomaly data).

The second metric used to evaluate the model's performance is the Receiver Operating Characteristics (ROC) curve. The ROC curve is utilized to assess the performance of a binary classification model by comparing it to a random classifier, represented by the line $y = x$ as a reference [44]. If the model's ROC points are located towards the northwest of the graph, it indicates that the model is producing good results and is performing better than the random classifier.

## 4. Evaluation

This section presented the evaluation of proposed LSTM-DC-WGAN-GP based model to detect anomaly on SWaT dataset. This model will be evaluated based on evaluation metrics in Section 3.4 and compare it with other techniques as well. Hyper parameters setting are also important to be carefully asses, as they directly affect performance of the model.

### *4.1 Model Training*

During the training phase, several architectures for the Critic and Generator were tested. Different architectures can be found in Table 1.

First model, the Critic had a simple architecture with 2 DC layers and 1 LSTM layer, while the Generator had a complex architecture with 1 LSTM layer and 4 DC layers. This model resulted in the Generator learning quickly, as indicated by the loss moving in a negative direction. However, the Critic was unable to challenge the Generator effectively, as indicated by its fluctuating loss and increasing value.

Second model, the Critic was made complex with 3 DC layers and 2 LSTM layers, while the Generator was made simple with 1 LSTM layer and 2 DC layers. In this model, the Critic was able to distinguish real and fake data features faster, as indicated by its loss moving towards the negative direction. However, the Generator struggled to keep up, with its fluctuating loss moving towards the positive direction.

Third model, both the Critic and Generator were made complex, with the Critic having 4 DC layers and 2 LSTM layers, and the Generator having 2 LSTM layers and 4 DC layers. This complexity did not result in stable training for the model

Stability was achieved with a balanced architecture, where the Critic had 3 DC layers and 1 LSTM layer, and the Generator had 1 LSTM layer and 3 DC layers. This balanced architecture allowed for stable training and prevented the model from overfitting.

Figure 5 shows the loss of the Critic and Generator during the training phase. A total of 1,582 datasets were used per iteration, with each epoch taking approximately 15 minutes to complete. The configurations are illustrated in Figure 2 and Figure 3, with hyper-parameters detailed in Algorithm 1.

### *4.2 Model Testing*

After 100 epochs, the models for the Critic and Generator are loaded into the test environment, and a dataset comprising 4,499 batches is processed, with each batch containing 100 seconds of data. The model is tasked with detecting any anomalies in the batched data. Figure 6 shows the resulting confusion matrix.

**Table 1.** Comparison of Critic and Generator Loss During Training

| No | Critic | Generator | Critic Loss | Generator Loss |
|---|---|---|---|---|
| 1 | Simple (3) | Complex (5) | Fluctuate, move to positive direction | Move to negative direction |
| 2 | Complex (5) | Simple (3) | Move to negative direction | Fluctuate, move to positive direction |
| 3 | Complex (6) | Complex (6) | Fluctuate | Fluctuate |
| 4 | Simple (4) | Simple (4) | Stable | Stable, move to negative direction |

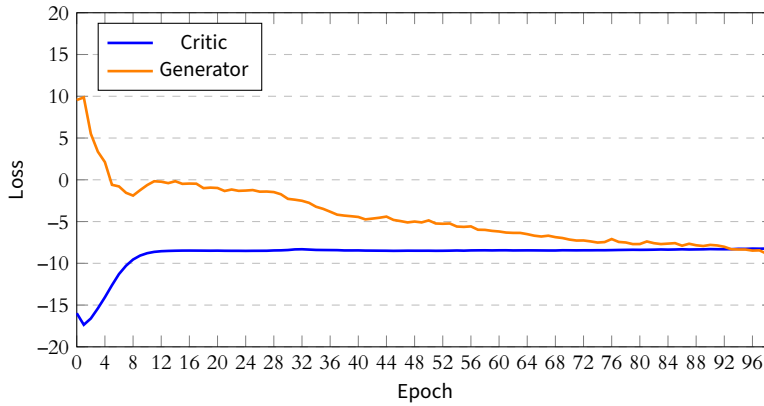


**Figure 5.** Loss of Critic and Generator on Training

Based on the results, the model achieved an accuracy of 0.9546, a precision of 0.9086, a recall of 0.6654, and an F1 score of 0.76818. Two components to highlight here are false positives (FP) and false negatives (FN). On the FP side, we observed that sensor measurements require time to stabilize after an attack, although iTrust has already marked them as normal. For example, during an attack on FIT-401, as shown in Figure 7, the attacker changed the recorded value to 0. After the attack, the sensor data continued to show 0 for 40 more seconds before returning to normal. This causes the model to mark it as an anomaly, even though the data is flagged as normal.

Different attack behaviors were observed, such as in Attacks 12 and 36, where the level sensor readings decreased over time. This type of error is missed by our model because the data is batched in 30-second intervals, and the median value is taken as representative. This makes gradual decreases difficult to detect, as the model may interpret them as expected behavior. Another type of attack behavior involves directly changing the sensor and actuator values outside their normal range, which is easier to catch compared to cases where the values remain stagnant within the normal range, contributing to FN predictions.

In addition to standard metrics, the model's performance is evaluated using the ROC curve. To plot the ROC curve, the threshold value is adjusted to achieve true

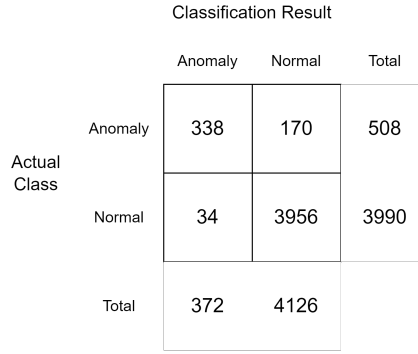


**Figure 6.** Confusion Matrix

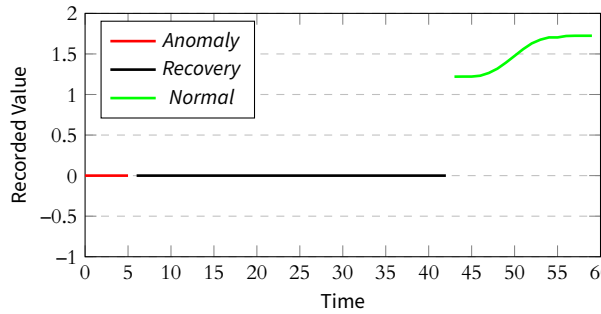


**Figure 7.** Attack on FIT-401

positive rate (TPR) and false positive rate (FPR) values approaching 0 and 1. For our model, we used 12 points on the graph, changing the threshold value by 0.5 starting at 0. As shown in Figure 8, all points from our model are located above the $y=x$ line, which represents a random classifier. This indicates that our model performs better than a random classifier, as stated in [44].

### *4.3 Model Comparison*

Research on finding anomalies in SWaT datasets have been done several times, namely: K-Nearest Neighbors (KNN) [26], Feature Bagging (FB) [26], Autoencoders (AE) [26], EGAN [26] and GAN-AD [45]. Comparison of evaluation result of proposed model with the other model shown in Table 2. Comparing to above model LSTM-DC-WGAN-GP model able to peform better in F1 Score, Precision and Result.

K-Nearest Neighbors (KNN) operates based on the distance between the test data and the n-nearest members of a class [46]. This method requires a sufficient amount of anomaly class data to effectively train the KNN model. In the case of SWaT, there is no anomaly data in the training set, which explains why KNN did not perform well. Additionally, the Fast Fourier Transform (FB) did not perform well, as it was initially designed for uni-variate data [26], while SWaT is multivariate.

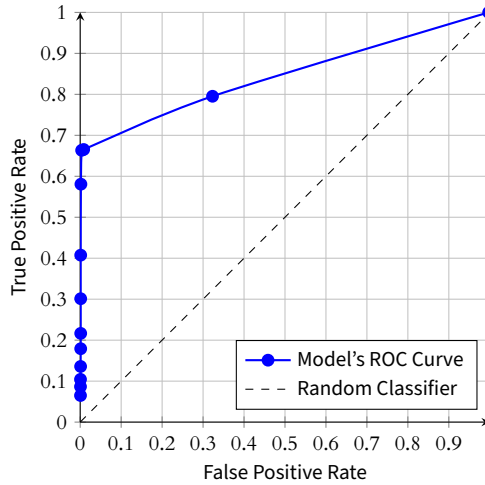


**Figure 8.** ROC Curve

**Table 2.** Comparison of anomaly detection result on SWaT dataset with different methods

| Method | F1 Score | Precision | Recall |
|---|---|---|---|
| KNN [26] | 0.350 | 0.348 | 0.348 |
| FB [26] | 0.360 | 0.358 | 0.358 |
| AE [26] | 0.520 | 0.516 | 0.516 |
| EGAN [26] | 0.510 | 0.405 | **0.677** |
| GAN-AD [45] | 0.750 | 0.857 | 0.636 |
| LSTM-DC-WGAN-GP | **0.768** | **0.909** | 0.665 |

Improvement can be seen with models using deep learning methods, such as Autoencoders (AE), Enhanced GAN (EGAN), and GAN-based Anomaly Detection (GAN-AD). AEs are capable of handling multivariate data and measure anomalies based on reconstruction error [45]. GAN-based methods also show improvements in their scores. However, compared to these models, the use of LSTM-DC in the architecture and WGAN-GP in the algorithm provides better results. The gradient penalty component helps keep the Critic's loss within Lipschitz continuity, resulting in a stable learning process, which leads to more effective weight updates. As shown in Figure 5, the Critic's loss value stabilizes starting from epoch 13.

## 5. Conclusion

In our research, we present an LSTM-DC-WGAN-GP model designed for detecting anomalies in unbalanced multivariate time series data. We evaluate this model using the SWaT dataset. The architecture of our model is based on two key components: Long Short-Term Memory (LSTM) for capturing sequence data correlations and Deep Convolutions (DC) layers for learning feature correlations in multivariate

data. Additionally, we employ the Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) algorithm, which has demonstrated effectiveness in anomaly detection scenarios.

Our approach involves training both the Critic and Generator components with normal data. During training, the Generator learns to create synthetic data that aligns with the distribution of normal data, while the Critic focuses on understanding the value distribution of each feature. Consequently, when anomalous data is introduced to the model, it can accurately detect and label it as an anomaly, leveraging its understanding of the normal data distribution.

Compared to previous approaches, our LSTM-DC-WGAN-GP model exhibits improvements across key evaluation metrics. It achieves accuracy of 0.9546, precision of 0.9086, recall of 0.6654, and F1 score of 0.76818, outperforming other methods such as K-Nearest Neighbors (KNN), Feature-Based (FB) approaches, Autoencoders (AE), Energy-based GANs (EGAN), and GAN-based Anomaly Detection (GAN-AD).

Looking ahead, further investigations can explore training on smaller time frames to capture shorter anomaly duration. Additionally, experimenting with hyperparameters and model architecture adjustments can enhance feature extraction capabilities and enable the model to provide probabilities indicating which features contribute to anomalies.

## Acknowledgement

The authors want to thank iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design for presenting the SWaT dataset.

## References

[1] Shunfeng Cheng et al. "A Wireless Sensor System for Prognostics and Health Management". In: *IEEE Sensors Journal* 10.4 (2010), pp. 856–862. DOI: 10.1109/JSEN.2009.2035817.

[2] Daoqu Geng et al. "Big Data-Based Improved Data Acquisition and Storage System for Designing Industrial Data Platform". In: *IEEE Access* 7 (2019), pp. 44574–44582. DOI: 10.1109/ACCESS.2019.2909060.

[3] Weiting Zhang, Dong Yang, and Hongchao Wang. "Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey". In: *IEEE Systems Journal* 13.3 (2019), pp. 2213–2227. DOI: 10.1109/JSYST.2019.2905565.

[4] Andrea Bonci, Massimiliano Pirani, and Sauro Longhi. "Tiny Cyber-Physical Systems for Performance Improvement in the Factory of the Future". In: *IEEE Transactions on Industrial Informatics* 15.3 (2019), pp. 1598–1608. DOI: 10.1109/TII.2018.2855747.

[5] Dmitry Shalyga, Pavel Filonov, and Andrey Lavrentyev. *Anomaly Detection for Water Treatment System based on Neural Network with Automatic Architecture Optimization*. 2018. arXiv: 1807.07282 [cs.LG].

[6] Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. *Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model*. 2016. arXiv: 1612.06676 [cs.LG].

[7] Pavel Filonov, Fedor Kitashov, and Andrey Lavrentyev. *RNN-based Early Cyber-Attack Detection for the Tennessee Eastman Process*. 2017. arXiv: 1709.02232 [cs.CR].

[8] David S. Matteson and Nicholas A. James. *A Nonparametric Approach for Multiple Change Point Analysis of Multivariate Data*. 2013. arXiv: 1306.4933 [stat.ME].

[9] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[10] Yong jun Hong et al. "How Generative Adversarial Networks and Their Variants Work: An Overview". In: *ACM Computing Surveys* 52.1 (Feb. 2019), pp. 1–43. ISSN: 1557-7341. DOI: 10.1145/3301282. URL: http://dx.doi.org/10.1145/3301282.

[11] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[12] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028 [cs.LG].

[13] Peng Xu, Rui Du, and Zhongbao Zhang. "Predicting pipeline leakage in petrochemical system through GAN and LSTM". In: *Knowledge-Based Systems* 175 (2019), pp. 50–61. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2019.03.013. URL: https://www.sciencedirect.com/science/article/pii/S0950705119301340.

[14] Wenqian Jiang et al. "A GAN-Based Anomaly Detection Approach for Imbalanced Industrial Time Series". In: *IEEE Access* 7 (2019), pp. 143608–143619. DOI: 10.1109/ACCESS.2019.2944689.

[15] Ugo Fiore et al. "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection". In: *Information Sciences* 479 (2019), pp. 448–455. ISSN: 0020-0255. DOI: 10.1016/j.ins.2017.12.030. URL: https://www.sciencedirect.com/science/article/pii/S0020025517311519.

[16] Thomas Schlegl et al. *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*. 2017. arXiv: 1703.05921 [cs.CV].

[17] Mikel Galar et al. "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting, and Hybrid-Based Approaches". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4 (2012), pp. 463–484. DOI: 10.1109/TSMCC.2011.2161285.

[18] Cheuk Ki Man et al. "Wasserstein Generative Adversarial Network to Address the Imbalanced Data Problem in Real-Time Crash Risk Prediction". In: *IEEE Transactions on Intelligent Transportation Systems* 23.12 (2022), pp. 23002–23013. DOI: 10.1109/TITS.2022.3207798.

[19] Octavio Loyola-González et al. "Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases". In: *Neurocomputing* 175 (2016), pp. 935–947. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2015.04.120. URL: https://www.sciencedirect.com/science/article/pii/S0925231215015908.

[20] Yong Oh Lee, Jun Jo, and Jongwoon Hwang. "Application of deep neural network and generative adversarial network to industrial maintenance: A case study of induction motor fault detection". In: *2017 IEEE International Conference on Big Data (Big Data)*. 2017, pp. 3248–3253. DOI: 10.1109/BigData.2017.8258307.

[21] Zhenyu Wu, Wenfang Lin, and Yang Ji. "An Integrated Ensemble Learning Model for Imbalanced Fault Diagnostics and Prognostics". In: *IEEE Access* 6 (2018), pp. 8394–8402. DOI: 10.1109/ACCESS.2018.2807121.

[22] Jonathan Goh et al. "A Dataset to Support Research in the Design of Secure Water Treatment Systems". In: *Critical Information Infrastructures Security*. Springer International Publishing, 2017, pp. 88–99. ISBN: 978-3-319-71368-7.

[23] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM Comput. Surv.* 41.3 (July 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882.

[24] Wafa Shafqat and Yung-Cheol Byun. "A Hybrid GAN-Based Approach to Solve Imbalanced Data Problem in Recommendation Systems". In: *IEEE Access* 10 (2022), pp. 11036–11047. DOI: 10.1109/ACCESS.2022.3141776.

[25] Kanika and Jimmy Singla. "Class Balancing Methods for Fraud Detection using Deep Learning". In: *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*. 2022, pp. 395–400. DOI: 10.1109/ICAIS53314.2022.9742836.

[26] Dan Li et al. *MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks*. 2019. arXiv: 1901.04997 [cs.LG].

[27] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. *Generating Videos with Scene Dynamics*. 2016. arXiv: 1609.02612 [cs.CV].

[28] Tim Salimans et al. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG].

[29] Md Abul Bashar and Richi Nayak. *ALGAN: Time Series Anomaly Detection with Adjusted-LSTM GAN*. 2023. arXiv: 2308.06663 [cs.LG].

[30] Sidi Lu et al. *CoT: Cooperative Training for Generative Modeling of Discrete Data*. 2019. arXiv: 1804.03782 [cs.LG].

[31] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[32] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306. ISSN: 0167-2789. DOI: 10.1016/j.physd.2019.132306. URL: http://dx.doi.org/10.1016/j.physd.2019.132306.

[33] Zhou Jing et al. "Lifetime Prediction of Ultraviolet Light-Emitting Diodes Using a Long Short-Term Memory Recurrent Neural Network". In: *IEEE Electron Device Letters* 41.12 (2020), pp. 1817–1820. DOI: 10.1109/LED.2020.3034567.

[34] Yuguang Chen et al. "Improved Long Short-Term Memory-Based Periodic Traffic Volume Prediction Method". In: *IEEE Access* 11 (2023), pp. 103502–103510. DOI: 10.1109/ACCESS.2023.3305398.

[35] Xujuan Zhou et al. "A New Deep Convolutional Neural Network Model for Automated Breast Cancer Detection". In: *2020 7th International Conference on Behavioural and Social Computing (BESC)*. 2020, pp. 1–4. DOI: 10.1109/BESC51023.2020.9348322.

[36] F.A. Gers, J. Schmidhuber, and F. Cummins. "Learning to forget: continual prediction with LSTM". In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. Vol. 2. 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218.

[37] Takeru Miyato et al. *Spectral Normalization for Generative Adversarial Networks*. 2018. arXiv: 1802.05957 [cs.LG].

[38] Donald J. Berndt and James Clifford. "Using Dynamic Time Warping to Find Patterns in Time Series". In: *KDD Workshop*. 1994. URL: https://api.semanticscholar.org/CorpusID:929893.

[39] Hardik Prabhu, Jayaraman Valadi, and Pandarasamy Arjunan. *Generative Adversarial Network with Soft-Dynamic Time Warping and Parallel Reconstruction for Energy Time Series Anomaly Detection*. 2024. arXiv: 2402.14384 [cs.LG].

[40] Thokozani Shongwe, Theo G. Swart, and Hendrik C. Ferreira. "Distance-Preserving Mapping with Euclidean Distance for 4-ary PAM". In: *2018 11th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP)*. 2018, pp. 1–6. DOI: 10.1109/CSNDSP.2018.8471854.

[41] Marco Cuturi and Mathieu Blondel. *Soft-DTW: a Differentiable Loss Function for Time-Series*. 2018. arXiv: 1703.01541 [stat.ML].

[42] Alexander Geiger et al. *TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks*. 2020. arXiv: 2009.07769 [cs.LG].

[43] Jason Brownlee. *How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification*. MachineLearningMastery.com. Accessed: 09-07-2024. URL: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/.

[44] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (2006), pp. 861–874. DOI: 10.1016/j.patrec.2005.10.010. URL: https://www.sciencedirect.com/science/article/pii/S016786550500303X.

[45] Dan Li et al. *Anomaly Detection with Generative Adversarial Networks for Multivariate Time Series*. 2019. arXiv: 1809.04758 [cs.LG].

[46] G. Parthasarathy and B.N. Chatterji. "A class of new KNN methods for low sample problems". In: *IEEE Transactions on Systems, Man, and Cybernetics* 20.3 (1990), pp. 715–718. DOI: 10.1109/21.57285.