Spring 2024

# Time series anomaly detection using generative adversarial networks

Shyam Sundar Saravanan
*Missouri University of Science and Technology*



**Department:**

# TIME SERIES ANOMALY DETECTION USING GENERATIVE ADVERSARIAL NETWORKS

by

SHYAM SUNDAR SARAVANAN

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

COMPUTER SCIENCE

2023

Approved by:

Tony Luo, Advisor
Venkata Sriram Siddhardh Nadendla
Nan Cen

## PUBLICATION THESIS OPTION

This thesis consists of the following article, formatted in the style used by the Missouri University of Science and Technology.

Paper I: Pages 3-31 have been submitted to IEEE International Conference on Computer Communications (INFOCOM) 2023. TSI-GAN: Unsupervised Time Series Anomaly Detection using Convolutional Cycle-Consistent Generative Adversarial Networks.

# ABSTRACT

Anomaly detection is widely used in network intrusion detection, autonomous driving, medical diagnosis, credit card frauds, etc. However, several key challenges remain open, such as lack of ground truth labels, presence of complex temporal patterns, and generalizing over different datasets. In this work, we propose TSI-GAN, an unsupervised anomaly detection model for time-series that can learn complex temporal patterns automatically and generalize well, i.e., no need for choosing dataset-specific parameters, making statistical assumptions about underlying data, or changing model architectures. To achieve these goals, we convert each input time-series into a sequence of 2D images using two encoding techniques with the intent of capturing temporal patterns and various types of deviance. Moreover, we design a reconstructive GAN that uses convolutional layers in an encoder-decoder network and employs *cycle-consistency loss* during training to ensure that inverse mappings are accurate as well. In addition, we also instrument a *Hodrick-Prescott filter* in post-processing to mitigate false positives. We evaluate TSI-GAN using 250 well-curated and harder-than-usual datasets and compare with 8 state-of-the-art baseline methods. The results demonstrate the superiority of TSI-GAN to all the baselines, offering an overall performance improvement of 13% and 31% over the second-best performer MERLIN and the third-best performer LSTM-AE, respectively.

# ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Tony Luo, and the Intelligent Systems Center (ISC) of Missouri S&T for their generous support. My appreciation also goes out to my family and friends, without their support and understanding it would have been impossible to accomplish what I have.

# TABLE OF CONTENTS

Page

# LIST OF ILLUSTRATIONS

# LIST OF TABLES

Table Page

PAPER I

# 1. INTRODUCTION

The aim of anomaly detection is to identify subsequences that are considered abnormal in a given context. Several applications rely upon accurate and automated anomaly detection, such as network security, smart manufacturing, and autonomous driving. In almost every application domain, time-series data is ubiquitous; therefore, time-series anomaly detection has been studied for years, especially recently using machine learning. There are three key challenges to this task, including the rarity of labels for abnormal data, presence of noise and complex temporal patterns, as well as requiring domain knowledge.

To address the first challenge, we take a reconstruction-based unsupervised learning approach. Reconstruction-based methods employ a model to learn a low-dimensional representation of the input time-series signal, reconstruct that signal using the representation, and then compare the reconstructed signal to the real input to calculate a reconstruction error. A large reconstruction error usually indicates the presence of an anomaly. In order to address the second challenge, we adopt a deep learning approach to harness deep neural networks' superior capability to recognize complex temporal correlations and hidden patterns in data. As a preprocessing step, we encode 1D time series into 2D images and exploit convolutional neural networks (CNN) for their superior performance on computer vision tasks. For the final challenge, we take a nonparametric approach throughout our design pipeline. As a result, the model doesn't make any assumptions about the underlying data, and it is not required to change parameters or model architectures for each dataset.

Thus finally, in this work, we propose a novel generative adversarial network (GAN) architecture called TSI-GAN for unsupervised time series anomaly detection. First, we transform 1D time series to 2D images using two encoding techniques to capture the temporal correlation and various types of deviance present in the time series. Second, we design a GAN with two critics and two generators that consist of convolutional layers in order to reconstruct encoded images and obtain effective reconstruction errors. In addition,

GAN-based methods typically solve an optimization problem using gradient descent as a separate step during *inference* to find the latent representation for *each sample* which is highly efficient. In contrast, we train an encoder-decoder network in our GAN with *cycle consistency loss* to obtain the inverse mapping of the latent representation automatically and immediately, making our inference almost instantaneous. Third, as a further enhancement, we address false positives (alarms) by post-processing the reconstruction errors and then combining the errors from two encoding channels using a weighted sum. This way, we obtain a reliable anomaly score vector which leads to reduced false positives.

# PAPER

## I. TSI-GAN: UNSUPERVISED TIME SERIES ANOMALY DETECTION USING CONVOLUTIONAL CYCLE-CONSISTENT GENERATIVE ADVERSARIAL NETWORKS

Shyam Sundar Saravanan, Tony Luo
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri 65401
Email: {ssdmw,tluo}@mst.edu

Mao V. Ngo
Singapore University of Technology and Design, Singapore
Email: vanmao_ngo@mymail.sutd.edu.sg

## ABSTRACT

Anomaly detection is widely used in network intrusion detection, autonomous driving, medical diagnosis, credit card frauds, etc. However, several key challenges remain open, such as lack of ground truth labels, presence of complex temporal patterns, and generalizing over different datasets. This paper proposes TSI-GAN, an unsupervised anomaly detection model for time-series that can learn complex temporal patterns automatically and generalize well, i.e., no need for choosing dataset-specific parameters, making statistical assumptions about underlying data, or changing model architectures. To achieve these goals, we convert each input time-series into a sequence of 2D images using two encoding techniques with the intent of capturing temporal patterns and various types of deviance. Moreover, we design a reconstructive GAN that uses convolutional layers in an encoder-decoder network and employs *cycle-consistency loss* during training to ensure that inverse

mappings are accurate as well. In addition, we also instrument a *Hodrick-Prescott filter* in post-processing to mitigate false positives. We evaluate TSI-GAN using 250 well-curated and harder-than-usual datasets and compare with 8 state-of-the-art baseline methods. The results demonstrate the superiority of TSI-GAN to all the baselines, offering an overall performance improvement of 13% and 31% over the second-best performer MERLIN and the third-best performer LSTM-AE, respectively.

**Keywords:** Anomaly detection, unsupervised learning, time series, neural networks, generative adversarial networks

## 1. INTRODUCTION

Anomaly detection aims to identify sub-sequences of various lengths that are considered abnormal within a context represented by data. Accurate and automated anomaly detection is crucial to a wide range of applications including network security, smart manufacturing, autonomous driving, and digital healthcare. Time-series data is ubiquitous in almost all application domains; hence, time-series anomaly detection has been actively studied for years, especially recently using machine learning. However, it remains a very challenging task for three key reasons: (i) labels for abnormal data are often rare, preventing proper training of supervised learning models; (ii) real-world time-series data is often subject to noise and characterized by complex temporal patterns that are difficult to identify; (iii) different datasets have different properties and thus often require a specific choice of parameters (e.g., using domain knowledge) for anomaly detectors to work well, making them hard to generalize.

To address the first challenge, we take an unsupervised learning approach. Modern approaches to unsupervised time series anomaly detection are typically *prediction-based* or *reconstruction-based*. Prediction-based methods [1, 2] forecast time-series signal and compare the real data to the predicted signal to calculate a prediction error. Reconstruction-based methods [3, 4, 5] employ a model to learn a low-dimensional representation of the

input time-series signal, reconstruct that signal using the representation, and then compare the reconstructed signal to the real input to calculate a reconstruction error. A large prediction or reconstruction error usually indicates the presence of an anomaly.

To address the second challenge, we adopt a deep learning approach to reap deep neural nets' superior capability of recognizing complex temporal correlations and hidden patterns from data [4, 6, 7]. In particular, we exploit convolutional neural networks (CNN) for their nifty performance in *computer vision* tasks, and in order to bridge this gap to time series, we encode 1D time series into 2D images as a preprocessing step, which also explains part of our approach, TSI, which stands for *Time Series to Images*.

For the third challenge, we take a nonparametric approach throughout our design pipeline. The resulting model does not make any assumptions about the underlying data and does not require choosing parameters for each dataset, or altering model architectures like [8].

Thus finally, in this paper, we propose a novel generative adversarial network (GAN) architecture called TSI-GAN for unsupervised time series anomaly detection. First, we perform 1D-to-2D using two encoding techniques to capture the temporal correlation and any type of deviance present in the time series. This encoding also allows us to leverage GAN's outstanding performance on tasks of image generation [9] and image-to-image translation [10]. Second, we design a GAN with two critics and two generators that consist of convolutional layers in order to reconstruct encoded images and obtain effective reconstruction errors. The aim of the GAN is to learn a generalized distribution of normal samples such that it produces reconstruction errors that are (i) large on anomalous inputs and (ii) small on normal data even in the presence of noise and time non-stationarity.

In addition, GAN-based methods typically solve an optimization problem using gradient descent as a separate step during *inference* to find the latent representation for *each sample* [11, 12]. This is highly inefficient on large datasets and impractical for real-time

applications as proven by [8]. In contrast, we train an encoder-decoder network in our GAN with *cycle consistency loss* to obtain the inverse mapping of the latent representation automatically and immediately, making our inference almost instantaneous.

Third, as a further enhancement we address false positives (alarms), which are often a pain point in existing anomaly detection methods. To this end, we post-process the reconstruction errors using the *Hodrick-Prescott filter* [13] and then combine the errors from two encoding channels using a weighted sum. This way, we obtain a reliable anomaly score vector which leads to reduced false positives.

In summary, this paper makes the following contributions:

- We introduce TSI-GAN, a novel convolutional cycle-consistent GAN architecture that learns to reconstruct 2D-encoded complex 1D time-series data and produces reliable reconstruction errors for detecting non-trivial time series anomalies without any labels, and in real-time.

- We address the challenge of model generalization by taking a fully nonparametric approach throughout our design pipeline. As a result, our method makes no assumptions about underlying data and requires no manual parameter choice, or changing model architectures.

- We mitigate false alarms as a common issue in anomaly detection, by post-processing the reconstruction error using a filtering technique and a weighting strategy.

- We benchmark TSI-GAN against eight state-of-the-art baseline methods on 250 well-curated and harder-than-usual datasets. The results validate our approach as the best performer overall, with a large winning margin over other methods.

# 2. RELATED WORK

Anomalies can generally be classified into three categories: point anomalies, collective anomalies, and contextual anomalies [14]. Point anomalies are single instances of data that are considered abnormal or irregular; collective anomalies are windows or sequences of anomalous data; contextual anomalies are data that are only considered anomalous (point or collective) depending on context such as time and space.

Due to the importance of anomaly detection in many applications, research in this field has been active for years. While statistical methods are classical, machine learning and especially deep learning based approaches have recently received increasingly more attention due to their attractive performance.

Proximity-based methods classify a data point as a point anomaly or a sub-sequence as a collective anomaly when its locality is sparsely populated. These methods can be further classified into *cluster-based methods* such as k-means clustering [15], *distance-based methods* such as k-nearest neighbors [16], and *density-based methods* such as DBSCAN [17]. The main drawback of these methods when applied to time series anomaly detection is that they require the number of anomalies to be known a priori and are unable to capture temporal patterns.

Under the above category, *Time-series discord discovery* is a recently proposed *distance-based* method for anomaly detection. Time series discords refer to very unusual time series subsequences, i.e., those that are maximally different from other subsequences in the same time series. Nakamura et al. introduced MERLIN [18] which works by measuring the similarities between subsequences of all possible lengths and selecting the anomalous ones using a discord refinement algorithm. MERLIN is considered to be the state-of-the-art for anomaly detection in univariate time series, and is included in our experiments as a baseline.

Prediction-based methods try to predict future values of a given time series by learning a predictive model; a data point is classified as an anomaly if the predicted value differs from real data by more than a specified threshold. Time series forecasting methods such as ARIMA [1] can be used for this but often require extensive examination and preprocessing of data and are parameter sensitive. Several deep learning approaches have been proposed to overcome these limitations. Notably, Hundman et al. [2] proposed an LSTM model with Dynamic Thresholding (LSTM-DT) to make predictions and introduce dynamic thresholding to reduce false positives.

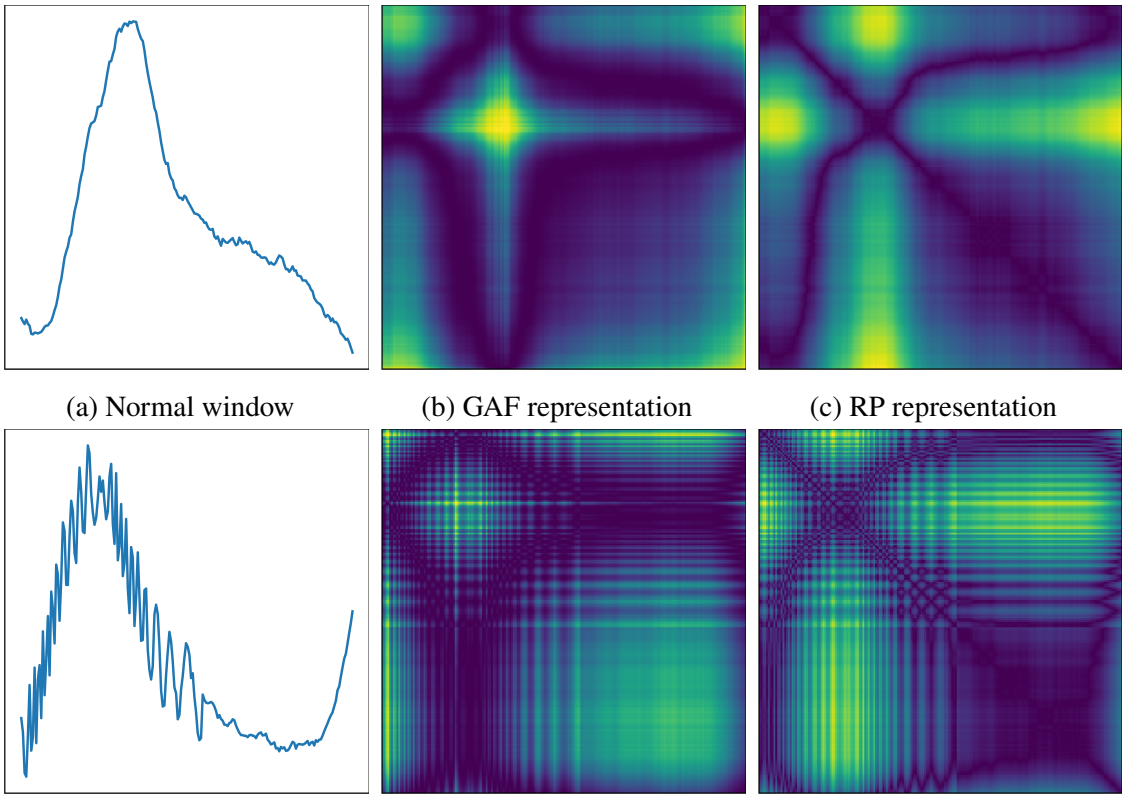
Reconstruction-based methods learn a latent low-dimensional representation of the input time-series data and try to reconstruct the input based on the representation. The assumption is that anomalies will lose information when mapped to the latent space and thus will not be reconstructed accurately, producing a larger reconstruction error. Hence, reconstruction error is measured at each time step and thresholding techniques are applied to detect the anomalies. Several deep learning approaches have been proposed including LSTM based Autoencoder (LSTM-AE) [19], Dense Autoencoder (DENSE-AE) [20], DONUT [21] which uses a Variational Autoencoder (VAE), and GAN-based methods [4, 11, 12, 22]. TadGAN [4] presents a recent study using GAN to perform this task. However, TadGAN requires the sampling interval of input data to be known for data preprocessing; otherwise, anomalies that do not have high amplitude will not be detected. This is a notable limitation because most anomalies in the real world are complicated rather than just simple amplitude spikes. Another related work is T2IVAE [6], which transforms time series to images and uses VAE to reconstruct the input time series. However, VAEs are prone to overfitting and often reconstruct anomalous samples quite accurately, resulting in unreliable reconstruction errors. Even though T2IVAE attempts to reduce this risk by employing an adversarial training strategy in the last five training epochs, the overfitting effect remains rather prominent.

We take a GAN-based approach instead of VAE because we find that GAN is strongly averse to the overfitting phenomenon when it comes to infrequent anomalous samples. Unlike TadGAN, however, we use convolutional neural networks in our GAN to learn feature maps as if learning from images. This way, we are able to reconstruct a more effective 2D-encoded time series, which allows us to obtain more reliable anomaly scores based on reconstruction errors.

There are also commercial tools including Microsoft Azure Anomaly Detector [23] and LinkedIn Luminol [24]. Azure uses spectral residual (SR) from the saliency detection domain [23] and CNN to learn a discriminating threshold. The output is a sequence of labels indicating if a particular timestamp is anomalous or not. Luminol uses the Bitmap detector algorithm [25] which divides input time series into chunks and calculates the frequency of similar chunks to calculate anomaly scores. These commercial tools are included as baselines in our experiments as well.

## 3. ENCODING TIME-SERIES TO IMAGES

To begin with, TSI-GAN transforms 1D time series input to 2D images in order to capture the complex temporal correlation in the data. Consider an input time series $\mathring{X} = \{x_1, x_2, ..., x_T\}$, where $T$ is the time series length. We use a sliding window with window size $W$ and step size $S$ to divide $\mathring{X}$ into $N$ overlapping sub-sequences, $\mathring{X}_k = \{x_{k+1}, x_{k+2}, \ldots, x_{k+W}\}$, where $k = 0, \ldots, N-1$ and $N = \lfloor \frac{T-W}{S} \rfloor$. We set $W = 64$ and $S = 1$ and convert each window of size 64 into a two-channel image of size $64 \times 64 \times 2$, using two time-series encoding techniques: Gramian Angular Field (GAF) [26] and Recurrence Plot (RP) [27].



(a) Normal window (b) GAF representation (c) RP representation

(d) Anomalous window (e) GAF representation (f) RP representation

Figure 1. Illustration of normal and anomalous windows encoded by GAF and RP for a sub-sequence of time series taken from the InternalBleeding dataset

### 3.1. GRAMIAN ANGULAR FIELD (GAF)

The core idea behind employing GAF transformation is that, when anomalous values exist in a time series, that transformation will *amplify* the difference between the normal samples and anomalous samples in the 2D image (see Fig. 1) and as such our GAN model will produce a large reconstruction error for the anomalous samples.

Given a sub-sequence $\mathring{X}_k = \{x_{k+i}\}_{i=1}^{W}$ at time step $k$, GAF rescales all the observations into the interval $[-1, 1]$ and calculates $\bar{X}_k = \{\bar{x}_{k+i}\}_{i=1}^{W}$, where

$$\bar{x}_{k+i} = \frac{(x_{k+i} - \max(\mathring{X}_k)) + (x_{k+i} - \min(\mathring{X}_k))}{\max(\mathring{X}_k) - \min(\mathring{X}_k)} \tag{1}$$

Next, we represent each rescaled $\bar{X}_k$ using polar coordinates, as radius $r = t_{k+i}/W$ where $t_{k+i} \in \mathbb{N}$ is the timestamp, and angular $\phi = \arccos(\bar{x}_{k+i}) \in [0, \pi]$. This polar conversion produces a one-to-one mapping with a unique inverse function, and preserves absolute temporal relation (as opposed to Cartesian coordinates). Thus, we can identify the temporal correlation at different time intervals by calculating the trigonometric sum between each point within the sub-sequence:

$$
\begin{aligned}
X_k^{GAF} &= \begin{bmatrix} \cos(\phi_1 + \phi_1) & \cdots & \cos(\phi_1 + \phi_W) \\ \cos(\phi_2 + \phi_1) & \cdots & \cos(\phi_2 + \phi_W) \\ \vdots & \ddots & \vdots \\ \cos(\phi_W + \phi_1) & \cdots & \cos(\phi_W + \phi_W) \end{bmatrix} \\
&= (\bar{X}_k)^T \otimes \bar{X}_k - \left(\sqrt{I - (\bar{X}_k)^2}\right)^T \otimes \sqrt{I - (\bar{X}_k)^2},
\end{aligned}
\tag{2}
$$

where $X_k^{GAF}$ is a $W \times W$ matrix, $I$ is the unit row vector $[1, 1, ..., 1]$ ($\bar{X}_k$ is a row vector too), and $\otimes$ represents outer product. We can easily derive (2) as follows:

$$
\begin{aligned}
&\cos(\phi_a + \phi_b) = \cos(\phi_a) \cdot \cos(\phi_b) - sin(\phi_a) \cdot sin(\phi_b) \\
&= \cos(\phi_a) \cdot \cos(\phi_b) - \sqrt{1 - (cos(\phi_a))^2} \cdot \sqrt{1 - (cos(\phi_b))^2} \\
&= \bar{x}_a \cdot \bar{x}_b - \sqrt{1 - \bar{x}_a^2} \cdot \sqrt{1 - \bar{x}_b^2}
\end{aligned}
\tag{3}
$$

The $X_k^{GAF}$ matrix has several desired properties: it preserves the temporal dependency since time increases from top-left to bottom-right; its diagonal retains the original angular and value information ($\cos(2\phi_a) = 2(\cos(\phi_a))^2 - 1 = 2\bar{x}_a^2 - 1$); its off-diagonal values $x^{GAF}_{k,(a,b);\ |a-b|=d} = \cos(\phi_a + \phi_b)$ represent relative temporal correlations by a trigonometric sum of the angular directions ($\phi_a$ and $\phi_b$) with respect to a time interval $d$. Fig. 1b depicts a GAF-encoded normal window and Fig. 1e shows a GAF-encoded anomalous

window. We can observe that the anomalous window is substantially different from the normal window. Since normal windows occur with much higher probability than abnormal ones in the training set, our GAN model will converge towards normal windows, thus able to produce a larger reconstruction error when encountering an anomaly.

### 3.2. RECURRENCE PLOT (RP)

A recurrence plot (RP) [27] is an image that represents the distance between observations extracted from a sub-sequence time series. Given a sub-sequence window $\mathring{X}_k = \{x_{k+i}\}_{i=1}^{W}$, we calculate a RP matrix $X_k^{RP}$ of dimension $W \times W$ where each element at row $a$ and column $b$ is defined as

$$x_{k,(a,b)}^{RP} = \Theta(\epsilon - \|x_{k+a} - x_{k+b}\|), \ \forall a, b \in \{1, \cdots, W\} \tag{4}$$

where $\Theta(\cdot) : \mathbb{R} \rightarrow \{0, 1\}$ is a Heaviside function, and $\epsilon$ is a predefined distance threshold. In this work, instead of using binary representation, we use raw distances $\|x_{k+a} - x_{k+b}\|$ (without the need for choosing $\epsilon$ or $\Theta(\cdot)$) to construct the RP matrix; the resulting 2D image will thus have more granularity scales of the distances. In order to align RP images with GAF images on the same scale, we scale the RP matrix into the range $[-1, 1]$ before further processing.

If any time step $k + a$ is anomalous, then both row $k + a$ and column $k + a$ in the RP image will be significantly different from other normal pixels because using multiple observations amplify the distance, as illustrated in Fig. 1f.

### 3.3. COMBINING TWO CHANNELS

After encoding the series using GAF and RP, respectively, we treat them as two channels and stack them along the channel axis to obtain

$$X_k = Stack(X_k^{GAF}, X_k^{RP}). \tag{5}$$

Since we have divided the original time series $\mathring{X}$ into $N$ overlapping sub-sequence windows of size $W$, and encoded each window as a 2-channel image of shape $[W \times W \times 2]$, we thus finally obtain a sequence of images $\mathbb{X} = \{X_k\}_{k=1}^{N}$.

## 4. THE TSI-GAN MODEL

### 4.1. MODEL ARCHITECTURE

Reconstruction-based anomaly detection methods learn a model that maps input data (in our case, an image with two channels) to the latent low-dimensional space and then reconstructs the input using the latent representation. The objective is to train a model that captures a generalized latent representation of the *normal* patterns, such that anomalies will *not* be reconstructed accurately and hence result in a larger reconstruction error. In our proposed method, we learn two mapping functions, $\mathcal{E} : \mathcal{X} \rightarrow \mathcal{Z}$ and $\mathcal{G} : \mathcal{Z} \rightarrow \mathcal{X}$, where $\mathcal{X}$ represents the input domain, $\mathcal{Z}$ represents the latent domain for which Gaussian distribution $\mathcal{N}(0, 1)$ is used. For any given input image at time step $k$, denoted by $X_k$, the model tries to reconstruct it as $X_k \rightarrow \mathcal{E}(X_k) \rightarrow \mathcal{G}(\mathcal{E}(X_k)) \approx \hat{X}_k$.

The entire model architecture is presented in Fig. 2. We model the above mapping functions as Generators, where $\mathcal{E}$ acts as an encoder which maps the input image to the latent space using convolution layers, and $\mathcal{G}$ acts as a decoder which transforms the latent representation to a reconstructed input image using transposed convolution. We use two Critics $C_x$ and $C_z$: $C_x$ regulates the decoder $\mathcal{G}$ by trying to distinguish real images $X$

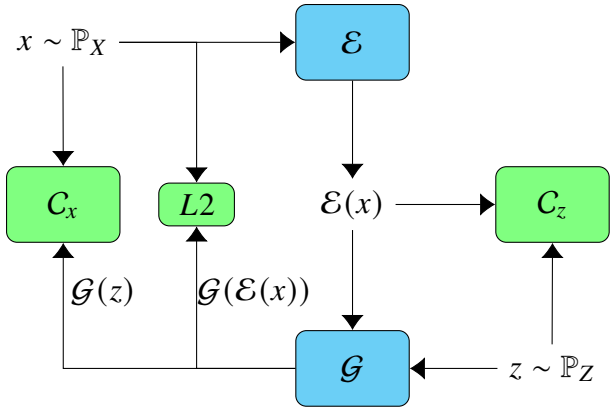



Figure 2. TSI-GAN model architecture

from the reconstructed images $\mathcal{G}(\mathcal{E}(x))$; $C_z$ regulates the encoder $\mathcal{E}$ by trying to liken the latent representation $\mathcal{E}(x)$ to the Gaussian noise $z$. The $L_2$-norm will be used in our cycle consistency loss which we describe later in Section 4.2.2.

## 4.2. LOSS FUNCTION AND TRAINING STRATEGY

We use two loss functions: (1) Wasserstein loss, to match the distribution of generated images with the distribution of input images, and (2) cycle consistency loss, to ensure the desired mapping route $X_k \rightarrow Z_k \rightarrow \hat{X}_k$.

**4.2.1. Wasserstein Loss.** Training a GAN-based model is a min-max game between the generator $\mathcal{G}$ and its critic $C_x$ with a vanilla adversarial loss function:

$$\min_{\mathcal{G}} \max_{C_x} \mathop{\mathbb{E}}_{x \sim \mathbb{P}_X} [\log C_x(x)] + \mathop{\mathbb{E}}_{z \sim \mathbb{P}_Z} [\log(1 - C_x(\mathcal{G}(z)))] \tag{6}$$

where $C_x$ outputs a probability score in the range $[0, 1]$ indicating the realness of the input image. However, the above objective suffers from two problems: *mode collapse* and *vanishing gradients*. In adversarial learning, mode collapse occurs when a Generator is trapped onto only a few particular samples that can fool the Critic well, thereby failing to capture the entire distribution. Vanishing gradients refer to the situation where, as the

**Algorithm 1:** Training TSI-GAN

**Input** Batch size $m$, learning rate $\alpha$, no. of $epochs$

**:**

**for** $n = 1, ..., epochs$ **do**

  Sample $\{(X_k)\}_{k=1}^{m}$ from input data $\mathbb{X}$

  Sample $\{(Z_k)\}_{k=1}^{m}$ from Gaussian distribution

  $g_{c_x} \leftarrow \nabla_{w_{c_x}} [\frac{1}{m} \sum_{k=1}^{m} (C_x(X_k) - C_x(\mathcal{G}(Z_k))) +$

    $\lambda_{gp}(X_k, \mathcal{G}(Z_k))]$

  $w_{c_x} \leftarrow w_{c_x} + \alpha \cdot RMSProp(w_{c_x}, g_{c_x})$

  $g_{c_z} \leftarrow \nabla_{w_{c_z}} [\frac{1}{m} \sum_{k=1}^{m} (C_z(Z_k) - C_z(\mathcal{E}(X_k))) +$

    $\lambda_{gp}(Z_k, \mathcal{E}(X_k))]$

  $w_{c_z} \leftarrow w_{c_z} + \alpha \cdot RMSProp(w_{c_z}, g_{c_z})$

  $g_{\mathcal{E}} \leftarrow \nabla_{w_{\mathcal{E}}} [\frac{1}{m} \sum_{k=1}^{m} (C_z(Z_k) - C_z(\mathcal{E}(X_k)) +$

    $\|X_k - \mathcal{G}(\mathcal{E}(X_k))\|_2)]$

  $w_{\mathcal{E}} \leftarrow w_{\mathcal{E}} + \alpha \cdot RMSProp(w_{\mathcal{E}}, g_{\mathcal{E}})$

  $g_{\mathcal{G}} \leftarrow \nabla_{w_{\mathcal{G}}} [\frac{1}{m} \sum_{k=1}^{m} (C_x(X_k) - C_x(\mathcal{G}(Z_k)) +$

    $\|X_k - \mathcal{G}(\mathcal{E}(X_k))\|_2)]$

  $w_{\mathcal{G}} \leftarrow w_{\mathcal{G}} + \alpha \cdot RMSProp(w_{\mathcal{G}}, g_{\mathcal{G}})$

Critic improves over time, the gradients approach zero and thus the loss function (6) starts to stagnate, ceasing to provide feedback to the Generator. To overcome these two issues, we use Wasserstein loss [9] in place of (6) to train our GAN:

$$\min_{\mathcal{G}} \max_{C_x \in \mathbf{C}_x} V_X(C_x, \mathcal{G}) \triangleq \mathbb{E}_{x \sim \mathbb{P}_X}[C_x(x)] - \mathbb{E}_{z \sim \mathbb{P}_Z}[C_x(\mathcal{G}(z))] \tag{7}$$

where $\mathbf{C}_x$ is the set of all the 1-Lipschitz functions. To ensure that Wasserstein Loss approximates the *Earth Mover's Distance* [28], the Critic needs to be 1-Lipschitz continuous. This can be enforced by adding a *gradient penalty* [9] regularization term $\lambda_{gp}(x, \mathcal{G}(z)) = \mathbb{E}\left(\|\nabla(C_x(\hat{x}))\|_2 - 1\right)^2$ to (7) where $\hat{x} = \beta x + (1-\beta)\mathcal{G}(z)$ and $\beta$ is a random number generated in the interval $[0, 1]$. This term penalizes gradients not equal to 1.

Similarly, for Encoder $\mathcal{E}$ and its Critic $C_z$, the loss function as in the objective is

$$\min_{\mathcal{E}} \max_{C_z \in \mathbf{C}_z} V_Z(C_z, \mathcal{E}) \triangleq \mathbb{E}_{z \sim \mathbb{P}_Z}[C_z(z)] - \mathbb{E}_{x \sim \mathbb{P}_X}[C_z(\mathcal{E}(x))] \tag{8}$$

where the Critic $C_z$ tries to distinguish between $z \sim \mathbb{P}_z$ and the encoded images $\mathcal{E}(x)$. Again, we add a gradient penalty regularization term $\lambda_{gp}(z, \mathcal{E}(x)) = \mathbb{E}(\|\nabla(C_z(\hat{z}))\|_2 - 1)^2$ where $\hat{z} = \beta z + (1-\beta)\mathcal{E}(x)$, to (8) to ensure a 1-Lipschitz continuous Critic. The complete architecture is given in Figure 2.

**4.2.2. Cycle Consistency Loss.** The GAN model described above is able to map $X_k$ to a desired $Z_k$. However, the inverse mapping of $Z_k$ back to $\hat{X}_k$ is not guaranteed by training with just vanilla adversarial losses or Wasserstein losses alone. This is because those losses only ensures distribution similarity but not instance similarity. To this end, Schlegl et al. [11] proposed an iterative approach via backpropagation to find the best $Z_k$ in the latent space that would generate $\mathcal{G}(\mathcal{E}(X_k))$ that is similar to the input image $X_k$. However, this method suffers from large search space and is inefficient for large datasets and real-time applications, as shown by Zenati et al. [8]. Hence, we use *cycle consistency loss* [29] to train the generators $\mathcal{G}$ and $\mathcal{E}$ by minimizing the $L_2$-norm distance between the input image and the reconstructed image:

$$\min_{\{\mathcal{E},\mathcal{G}\}} V_{L2}(\mathcal{E},\mathcal{G}) \triangleq \mathbb{E}_{x\sim\mathbb{P}_X}[\|x - \mathcal{G}(\mathcal{E}(x))\|_2] \tag{9}$$

**4.2.3. Final Objective.** Combining the objectives (7), (8), (9) we arrive at the final objective:

$$\min_{\{\mathcal{E},\mathcal{G}\}} \max_{\{C_x\in\mathbf{C}_x, C_z\in\mathbf{C}_z\}} V_X(C_x,\mathcal{G}) + V_Z(C_z,\mathcal{E}) + V_{L2}(\mathcal{E},\mathcal{G}) \tag{10}$$

The complete training procedure is presented in Algorithm 1, where $w_{c_x}, w_{c_z}, w_{\mathcal{G}}, w_{\mathcal{E}}$ are the weights for critic $C_x$, $C_z$, decoder $\mathcal{G}$ and encoder $\mathcal{E}$, respectively, and $g_{c_x}, g_{c_z}, g_{\mathcal{G}}, g_{\mathcal{E}}$ are their corresponding gradients.

### 4.3. POST-PROCESSING AND ANOMALY DETECTION

After training the above GAN model, we calculate the sum of squared errors (SSE) between the input image $X_k$ and the reconstructed image $\hat{X}_k$ to get the reconstruction error $\epsilon_{rec}$. We then extract the reconstruction error for each channel as $\epsilon_{gaf}$ and $\epsilon_{rp}$. Calculating thresholds directly on the raw reconstruction error will lead to many false positives. To mitigate this, we smooth the reconstruction error to suppress frequently occurring minor error peaks which are usually caused by normal behavior rather than anomalies. We use the *Hodrick-Prescott filter* [13] because of its excellent capability of removing short-term fluctuations in data since we are only concerned with peaks that persist for a sustained period of time. It extracts a smooth trend $r$ from a given sequence $\epsilon$ of length $N$ by solving:

$$\min_{r}\left(\sum_{k=1}^{N}(\epsilon_k - r_k)^2 + \lambda \sum_{k=2}^{N-1}[(r_{k+1}-r_k)-(r_k-r_{k-1})]^2\right) \tag{11}$$

After smoothing $\epsilon_{gaf}$ and $\epsilon_{rp}$, we find the local (neighborhood) peaks in each channel and sort them in descending order to calculate a confidence level $\sigma \in [1,2]$ for each channel:

$$\sigma = \frac{peaks[0] - peaks[1]}{peaks[0]} + 1,$$

where $peaks[0]$ and $peaks[1]$ are the first and the second highest peaks in the smoothed reconstruction errors, respectively. The idea is that when the difference between these two peaks is large, that channel is assumed to be more confident about its detection of the anomaly and hence weighed higher in the final anomaly score. This score is defined by combining the two reconstruction errors $\epsilon_{gaf}$ and $\epsilon_{rp}$ using their respective confidence level:

$$score\_vec = \sigma_{gaf} \times \epsilon_{gaf} + \sigma_{rp} \times \epsilon_{rp}. \tag{12}$$

**Algorithm 2:** Anomaly Detection using TSI-GAN

// Post-processing for reliable anomaly scores
**for** $k = 1, ..., N$ **do**
  $\hat{X}_k \leftarrow \mathcal{G}(\mathcal{E}(X_k))$
  $\hat{X}_k^{GAF}, \hat{X}_k^{RP} \leftarrow$ extract GAF & RP channels from $\hat{X}_k$
  $\epsilon_{gaf_k} \leftarrow \sum_{i=1}^{W} \sum_{j=1}^{W} (X_{k,i,j}^{GAF} - \hat{X}_{k,i,j}^{GAF})^2$
  $\epsilon_{rp_k} \leftarrow \sum_{i=1}^{W} \sum_{j=1}^{W} (X_{k,i,j}^{RP} - \hat{X}_{k,i,j}^{RP})^2$
**for** $ch \in \{gaf, rp\}$ **do**
  $\epsilon_{ch} \leftarrow \text{HP}(\epsilon_{ch})$ // HP: Hodrick-Prescott filter
  $peaks_{ch} \leftarrow \text{find\_peaks}(\epsilon_{ch})$
  $\sigma_{ch} \leftarrow \frac{(peaks_{ch}[0] - peaks_{ch}[1])}{peaks_{ch}[0]} + 1$
$score\_vec \leftarrow \sigma_{gaf} \times \epsilon_{gaf} + \sigma_{rp} \times \epsilon_{rp}$
// Detect anomalies
$mean \leftarrow \text{mean}(score\_vec)$
**for** $k = 1, ..., N$ **do**
  **if** $score\_vec_k > mean$ **then**
    $pred_k = true$
  **else**
    $pred_k = false$
Group consecutive $pred_k$'s into $\{seq_i\}_{i=1}^{L}$
// Pruning to reduce false alarms
$\{m_i\} \leftarrow \max(\{seq_i\})$
$\{m_i\} \leftarrow \text{sort}(\{m_i\}, descending = true)$
sort $\{seq_i\}$ in the same order of $\{m_i\}$
**for** $i = 1, ..., L$ **do**
  $p_i \leftarrow (m_{i-1} - m_i)/m_{i-1}$
  **if** $p_i < \theta$ **then**
    reclassify $\{seq_j\}_{j=i}^{L}$ as normal
    break;

Here it is defined as a vector (of length $N$) because each of the $N$ windows will have an anomaly score. For the weight $\sigma$, if there are multiple anomalies, the difference between $peaks[0]$ and $peaks[1]$ will be small and thus $\sigma$ will be smaller than the other channel if the other channel detects a single outlier, which is desired since outliers are rare by definition and thus single outliers are more likely than multiple. Otherwise, if both channels detect multiple, they will be weighted by similar $\sigma$'s.

After obtaining the anomaly score for each window, we calculate the mean anomaly score over all the windows and any window that exceeds this threshold is flagged as an anomaly. Following that, consecutive anomalous windows will be grouped together to form a sequence (i.e., collective anomaly). To further mitigate false positives, we use an anomaly pruning approach introduced by Hundman et al. [2]. Given a set of predicted anomalous sequences $\{seq_1, seq_2, ..., seq_L\}$, where $seq_i = (seq_{start(i)}, ..., seq_{end(i)})$ denotes a set of points from the start to the end of an anomalous sequence $i$, we find the maximum score of each sequence to obtain $\{m_1, m_2, ..., m_L\}$. Sort these values in descending order (indexed by $i'$) and compute a descent rate $p_{i'} = (m_{i'-1} - m_{i'})/m_{i'-1}$. Then, look for the first sequence that satisfies the condition $p_{i'} < \theta$ (by default $\theta = 0.1$), this and its subsequent sequences are reclassified as normal.

The above post-processing and detection procedures are formulated in Algorithm 2.

## 5. PERFORMANCE EVALUATION

### 5.1. DATASETS

We use the UCR 2021 anomaly detection dataset[1] which contains 250 sub-datasets collected from a variety of sources. Unlike commonly used datasets such as Yahoo, Numenta, and NASA which are found to have numerous flaws [30] including incorrect ground truth labels, triviality of the anomalies and unrealistic anomaly density, this UCR dataset is carefully curated, harder to detect and is much more reliable. Moreover, this dataset contains a combination of point, collective, and contextual anomalies as well as amplitude, seasonal, and trend anomalies, which offers a good variety for evaluation.

[1] http://bit.ly/3V2n6FY

Table 1. Statistics of Datasets used in our experiments.

| Property | Dataset | | | | | | |
|---|---|---|---|---|---|---|---|
| | AirTemperature | PowerDemand | InternalBleeding | EPG | NASA T-1 | Noise | All datasets |
| # Sub-datasets | 14 | 8 | 26 | 12 | 10 | 16 | 250 |
| # Data Points | 98208 | 239448 | 194992 | 359304 | 113488 | 629494 | 19353766 |
| # Anomalous Points | 398 | 1688 | 3018 | 1292 | 644 | 3134 | 49363 |
| # (% tot.) | 0.004% | 0.007% | 0.015% | 0.003% | 0.005% | 0.004% | 0.002% |

We choose a total of 6 categories from this dataset and each category contains 4-13 original sub-datasets; each original sub-dataset comes with a distorted duplicate by adding artificial fluctuations. Therefore, the number of sub-datasets is doubled. The only exception is the `Noise` category in which the sub-datasets are chosen from multiple other categories with Gaussian noise added. A brief description of each category is as follows: `AirTemperature` consists of hourly air temperature between 03/01 and 03/31 from 2009 to 2019, collected from CIMIS station 44 in Riverside, CA. `PowerDemand` consists of Italian power demand data between 1/1/1995 and 5/31/1998. `InternalBleeding` consists of the arterial blood pressure measurements of pigs. `EPG` is collected from an insect known as Asian Citrus Psyllid, recorded using an Electropalatography (EPG) apparatus. `NASA T-1` is collected from NASA Mars Science Laboratory (MSL) dataset that consists of spacecraft telemetry signals. Detailed statistics of each category and all the datasets is presented in Table 1.

## 5.2. BASELINES

We compare our model against eight baselines.

LSTM-DT (Prediction-based). Similar to Hundman et al. [2], we use two layers of LSTM with 80 units each and a subsequent dense layer with one unit to predict the value at the next time step; point-wise prediction errors are then computed to detect anomalies.

Table 2. Architecture and Hyperparameters of the proposed TSI-GAN.

| Operation | Kernel | Strides | Units | BatchNorm? | Activation |
|---|---|---|---|---|---|
| **Encoder** | | | | | |
| Convolution | $7 \times 7$ | $3 \times 3$ | 48 | ✓ | ReLU |
| Convolution | $5 \times 5$ | $3 \times 3$ | 96 | ✓ | ReLU |
| Convolution | $4 \times 4$ | $2 \times 2$ | 192 | ✓ | ReLU |
| Convolution | $2 \times 2$ | $1 \times 1$ | z_dim | × | – |
| **Decoder** | | | | | |
| Transposed Convolution | $2 \times 2$ | $1 \times 1$ | 192 | ✓ | LeakyReLU |
| Transposed Convolution | $4 \times 4$ | $2 \times 2$ | 96 | ✓ | LeakyReLU |
| Transposed Convolution | $5 \times 5$ | $3 \times 3$ | 48 | ✓ | LeakyReLU |
| Transposed Convolution | $7 \times 7$ | $3 \times 3$ | 2 | × | Tanh |
| **Critic X** | | | | | |
| Convolution | $7 \times 7$ | $3 \times 3$ | 48 | LayerNorm | LeakyReLU |
| Convolution | $5 \times 5$ | $3 \times 3$ | 96 | LayerNorm | LeakyReLU |
| Convolution | $4 \times 4$ | $2 \times 2$ | 192 | LayerNorm | LeakyReLU |
| Convolution | $2 \times 2$ | $1 \times 1$ | 1 | × | – |
| **Critic Z** | | | | | |
| Fully Connected | | | 50 | LayerNorm | ReLU |
| Fully Connected | | | 25 | LayerNorm | ReLU |
| Fully Connected | | | 1 | × | – |
| z_dim | 100 | Learning rate ($\alpha$) | 1e-4 | | |
| Optimizer | RMSProp | Weight decay ($\lambda_{wd}$) | 1e-4 | | |
| Iterations | 5000 | Batch Size | 128 | | |

Autoencoder (Reconstruction-based). We consider two versions of autoencoders: one is a dense autoencoder consisting of three fully connected layers with 60, 20, and 60 units respectively; the other is an LSTM autoencoder which contains two LSTM layers each with 60 units. Similar to our approach, these models try to reconstruct the input sample at each time step and then use the reconstruction error to detect the anomalies.

TadGAN [4] (Reconstruction-based). The generators in TadGAN use a 1-layer BiLSTM with 100 hidden units and a 2-layer BiLSTM with 64 hidden units. Additionally, TadGAN uses a 1D (whereas we use 2D) convolutional layer for the critics to capture

temporal correlation. Their best method of combining reconstruction error and critic score, as reported by their paper, is Critic $\times$ DTW, which we adopt in our experiments to detect anomalies for TadGAN.

DONUT [21] (Reconstruction-based) uses a Variational Autoencoder (VAE) to detect anomalous seasonal KPIs in web applications. It computes reconstruction errors at each time step and subsequently applies thresholding to detect anomalies.

Luminol [24] is an open source anomaly detection Python library. It uses the Bitmap detector algorithm by default which divides the input time series into chunks and calculates the frequency of similar chunks to calculate the anomaly scores.

MERLIN [18] (Distance-based) is an anomaly detection algorithm that measures the similarities between subsequences of all possible lengths and flags the anomalous ones using a discord refinement algorithm.

Microsoft Azure Anomaly Detector [23] (Commercial tool). Azure uses spectral residuals (SR) adopted from a preprocessing step in computer vision called saliency detection, as well as CNN to learn a discriminating threshold, and outputs a sequence of labels indicating if a particular timestamp is anomalous.

## 5.3. PERFORMANCE METRICS

In real world application scenarios, most anomalies happen in the form of collective anomalies and hence we use the window based rules introduced by Hundman et al. [2]: (1) If an anomalous window overlaps any predicted window, a true positive (TP) is recorded; (2) If a predicted window does not overlap with any anomalous window, a false positive (FP) is recorded; (3) If an anomalous window does not overlap with any predicted window, a false negative (FN) is recorded. Based on this set of rules, we calculate *Precision* and *F1-Score* as the performance metrics.

Table 3. Average F1-Score on original and distorted datasets for each category, as well as F1-Score and Precision averaged over all the 250 datasets.

| **Model** | AirTemperature | | PowerDemand | | InternalBleeding | | EPG | | NASA T-1 | | Noise | All 250 datasets | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Orig. | Distor. | Orig. | Distor. | Orig. | Distor. | Orig. | Distor. | Orig. | Distor. | | F1 | Precision |
| TSI-GAN | **1.0** | **0.833** | **0.667** | **0.667** | 0.846 | 0.474 | **0.5** | **0.556** | **0.933** | 0.267 | 0.479 | **0.468** | **0.445** |
| MERLIN | 0.054 | 0.18 | 0.04 | 0.071 | **0.926** | **0.721** | 0.354 | 0.191 | 0.613 | **0.6** | **0.49** | 0.414 | 0.402 |
| LSTM-AE | 0.389 | 0.611 | 0.375 | 0.583 | 0.654 | 0.308 | 0.222 | 0.444 | 0.533 | 0.333 | 0.208 | 0.355 | 0.301 |
| DONUT | 0.611 | 0.444 | 0.083 | 0.1 | 0.59 | 0.564 | 0.278 | 0.167 | 0.333 | 0.533 | 0.458 | 0.351 | 0.325 |
| LSTM-DT | 0.778 | **0.833** | 0.25 | 0.5 | 0.615 | 0.449 | 0.222 | 0.222 | 0.6 | **0.6** | 0.271 | 0.32 | 0.289 |
| DENSE-AE | 0.194 | 0.111 | 0.0 | 0.0 | 0.231 | 0.077 | 0.222 | 0.222 | 0.2 | 0.0 | 0.271 | 0.159 | 0.136 |
| TadGAN | 0.0 | 0.133 | 0.0 | 0.0 | 0.282 | 0.24 | 0.233 | 0.189 | 0.267 | 0.2 | 0.171 | 0.131 | 0.092 |
| Azure | 0.181 | 0.199 | 0.083 | 0.196 | 0.099 | 0.176 | 0.167 | 0.167 | 0.007 | 0.017 | 0.084 | 0.05 | 0.037 |
| Luminol | 0.022 | 0.021 | 0.078 | 0.089 | 0.118 | 0.046 | 0.037 | 0.088 | 0.009 | 0.014 | 0.019 | 0.049 | 0.021 |

## 5.4. EXPERIMENTAL RESULTS

Table 3 reports the average F1-Score on the original and distorted datasets for each category, and in the last column, the F1-Score and Precision averaged over all the 250 datasets.

Overall, it is observed that TSI-GAN achieves an F1-Score of 0.468 and Precision of 0.445, outperforming all the baseline methods. More specifically, TSI-GAN offers an improvement of 13% and 31% on F1-score over the second and the third best methods, MERLIN (0.414) and LSTM-AE (0.355), respectively. We note that 85-95% of the improvement was attributed to GAN, while 5-15% was attributed to post-processing. When the individual categories are considered, TSI-GAN performs the best on AirTemperature, PowerDemand, EPG for both original and distorted datasets and wins over other methods by a significant margin; it also offers competitive performance on other categories (InternalBleeding, NASA-T1 Distorted, and Noise) as well.

Using MERLIN as a benchmark, we measure the performance difference between each method and MERLIN in Fig. 4. It indicates that TSI-GAN is the only one that offers a positive performance improvement while all the other methods underperform MERLIN. Next, we present an in-depth analysis of MERLIN.

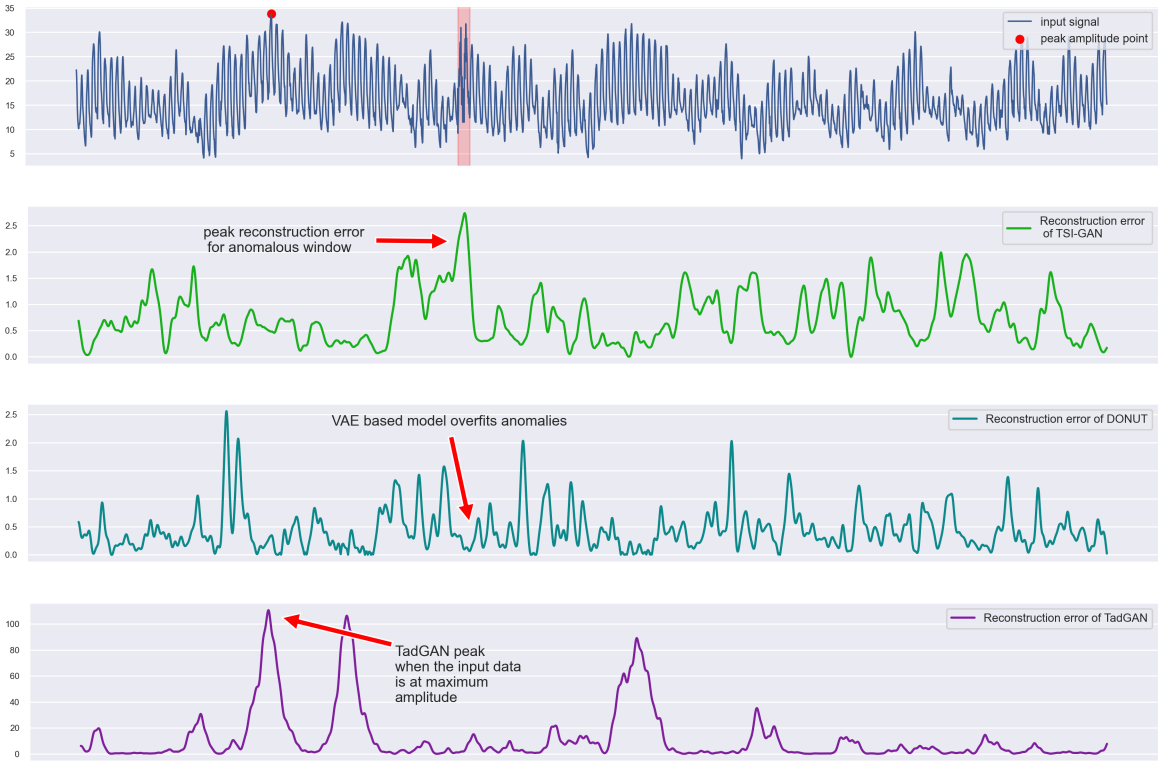



Figure 3. TSI-GAN vs. DONUT vs. TadGAN when applied to an example time series. The translucent red interval depicts the ground-truth anomaly. DONUT as a VAE-based model tends to overfit anomalies, while TadGAN tends to only detect high-amplitude data points as anomalies since it does not use any feature engineering to capture other types of anomalies. TSI-GAN, however, encodes input time series using GAF and RP and thus is able to detect various types of deviance.

**5.4.1. Comparison with MERLIN.** Notably, MERLIN is a non-deep-learning based method, but it is well-designed and outperforms other deep learning based methods as shown by Table 3. On the other hand, MERLIN underperforms when compared to TSI-GAN (0.414 vs 0.468) in terms of F1 score averaged over all the 250 datasets. After an in-depth investigation, we identify the following important limitations of MERLIN.

First, MERLIN predicts multiple integer positions surrounding each detected anomaly, and those positions wander in an uncertain range. For example, a sub-dataset in the `AirTemperature` category has an anomaly on the interval $[5703, 5727]$ (ground truth), but MERLIN's prediction are "5673, 5672, 5671, 5670, 5669, 5668, 5667, 5675, 5675, 5664, 5674, 5674,

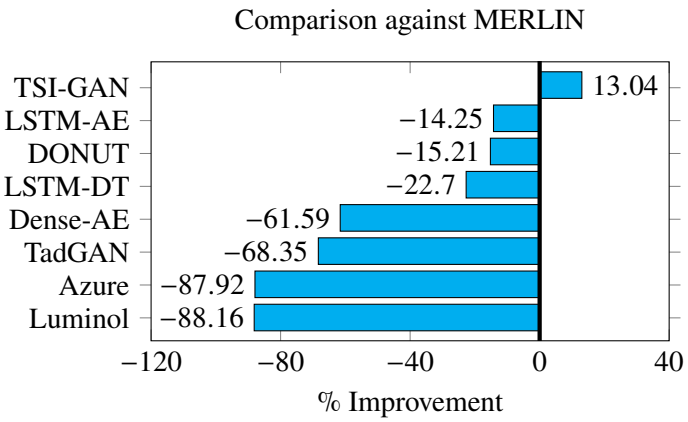



Figure 4. Comparing all anomaly detection methods against MERLIN in terms of F1 score averaged across all 250 datasets, expressed as a percentage of improvement.

5673, 5681, 5680, 5679, 5673, 5673, 5672, 5671, 5672, 5669, 2193, 2193, 2192, 2192, 2192, 2191, 2191, 2190, 2190, 2189, 2188, 2187, 2186, 2186, 2185, 2184, 2183, 2182, 2181, 2180, 2179, 2179, 2178, 2177".

Because of this, MERLIN requires a *relaxed* scoring function to reap good performance, where we deem its prediction $p$ correct if $(anomaly_{start} - 100) < p < (anomaly_{end} + 100)$. However, if we reduce 100 to 5, it suffers from a large performance downgrade from 0.414 to 0.129, a 68% decrease, when averaged across all 250 datasets. Our experiments has adopted 100 in favor of MERLIN.

Second, MERLIN builds on top of the DRAG algorithm [31] which requires a user-defined parameter $r$, and the algorithm could fail depending on its value and the user has to re-run it with a different value. MERLIN claims to have overcome this by trying many $r$ values in a somewhat brute-force manner, where it sets an upper bound and gradually decreases it by tiny bits until the algorithm returns success. However, our experiments observed that this approach still fails multiple times; the reason is that the upper bound depends on a user provided value $L$ which requires knowing the length of each anomaly *a priori*.

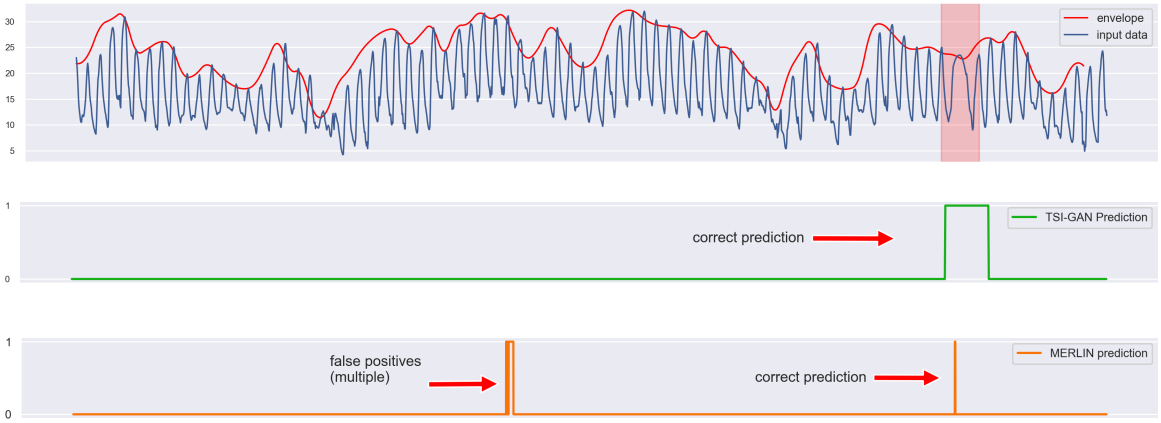



Figure 5. Illustration TSI-GAN vs. MERLIN when applied to an example time series. The translucent red interval denotes the ground-truth anomaly. MERLIN as a distance-based method tends to produce multiple false positives when the envelope of the time series varies randomly without a clear pattern. On the other hand, TSI-GAN learns a good representation from normal data and does not produce any false positives.

Third, when the envelope of the input time series varies randomly without a clear pattern, MERLIN tends to produce multiple false positives (see Figure 5) due to its distance-based approach.

TSI-GAN does not have any such limitations, since it operates by windows, does not require dataset-specific parameters, and takes a reconstruction-based approach.

**5.4.2. Comparison with Deep Learning Based Methods.** Among all the deep learning based methods, LSTM-AE performs the best, with an average F1-Score of 0.355; DONUT comes in second with a slightly lower score 0.351. We examine the possible reasons for their shortfall as compared to TSI-GAN and how our approach overcomes them. As we mentioned earlier, autoencoder based methods carry the risk of overfitting anomalies during training, by reconstructing anomalous samples just as accurately as normal samples. DONUT which employs VAE has this tendency as can be seen in Fig. 3. This is also a plausible reason for underperformance of other autoencoder based models such as LSTM-AE.

Table 4. Training and Inference time of TSI-GAN

| | **AirTemperature** | | | **PowerDemand** | | |
|---|---|---|---|---|---|---|
| | # of Samples | Total time | Per-window time | # of Samples | Total time | Per-window time |
| **Training** | 7996 | 251s | 0.06s | 29772 | 246s | 0.01s |
| **Inference** | 4083 | 10s | 0.002s | 11862 | 27s | 0.002s |

In contrast, TSI-GAN uses an adversarial training strategy which makes our model largely immune to this behaviour. However, while TadGAN and many others alike also use adversarial learning, they are unable to capture anomalies that are not amplitude spikes unless dataset-specific parameters such as sampling interval is known. The reason is that they do not instrument feature engineering to capture anomalies that deviates in seasonality, trend, etc., and therefore tend to only detect high amplitude points in the input as can be observed in Fig. 3. This is a main reason why TadGAN only performs well on datasets in which all anomalies are amplitude spikes; such anomalies, however, are *trivial* to detect as pointed out by [30]. On the other hand, TSI-GAN uses GAF and RP encoding which substantially enhances its ability to detect various types of non-trivial deviance, as can be observed in Fig. 3.

**5.4.3. Time Efficiency.** We report the training and inference time of TSI-GAN in Table 4. The times are measured on a NVIDIA RTX 3070 GPU with 8GB of VRAM along with AMD Ryzen 7 5800H @ 3.20 GHz CPU. We can see that the training time remains almost constant irrespective of the number of training samples, the reason is that we train for iterations and not epochs. More importantly, the inference time per window is only two milliseconds, which signifies that TSI-GAN is well suited for use on rapidly arriving streaming data.

## 6. CONCLUSION

In this paper, we introduce TSI-GAN, a novel convolutional cycle-consistent GAN architecture that learns to reconstruct 2D-encoded time-series data and produces effective and reliable reconstruction errors for detecting time series anomalies. We also address the challenge of mitigating false alarms by post-processing the reconstruction error using a filtering technique and computing a reliable score using a weighted combination of channels. Our extensive experimental results demonstrate that TSI-GAN outperforms 8 state-of-the-art baseline methods and achieves the highest averaged F1-Score over all the 250 datasets (that are well-curated and non-trivial). We also provide an in-depth analysis of the baselines' limitations and how our model addresses them. TSI-GAN is unsupervised and generalizes well without the need for parameter calibration, enabling it to be applicable to most applications that involve time series, such as network intrusion detection, cyber defense, IoT and edge computing.

## REFERENCES

[1] E. H. Pena, M. V. de Assis, and M. L. Proença, "Anomaly detection using forecasting methods arima and hwds," in *2013 32nd international conference of the chilean computer science society (sccc)*, pp. 63–66, IEEE, 2013.

[2] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, 2018.

[3] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

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

[5] A. Vahdat and J. Kautz, "Nvae: A deep hierarchical variational autoencoder," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19667–19679, 2020.

[6] L. Xu, L. Zheng, W. Li, Z. Chen, W. Song, Y. Deng, Y. Chang, J. Xiao, and B. Yuan, "Nvae-gan based approach for unsupervised time series anomaly detection," *arXiv preprint arXiv:2101.02908*, 2021. not peer-reviewed or published, code not available.

[7] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 4027–4035, 2021.

[8] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *2018 IEEE International conference on data mining (ICDM)*, pp. 727–736, IEEE, 2018.

[9] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

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

[12] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *International conference on artificial neural networks*, pp. 703–716, Springer, 2019.

[13] R. J. Hodrick and E. C. Prescott, "Postwar us business cycles: an empirical investigation," *Journal of Money, credit, and Banking*, pp. 1–16, 1997.

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, jul 2009.

[15] S. Chawla and A. Gionis, "k-means–: A unified approach to clustering and outlier detection," in *Proceedings of the 2013 SIAM international conference on data mining*, pp. 189–197, SIAM, 2013.

[16] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European conference on principles of data mining and knowledge discovery*, pp. 15–27, Springer, 2002.

[17] M. Çelik, F. Dadaşer-Çelik, and A. Ş. Dokuz, "Anomaly detection in temperature data using dbscan algorithm," in *2011 international symposium on innovations in intelligent systems and applications*, pp. 91–95, IEEE, 2011.

[18] T. Nakamura, M. Imamura, R. Mercer, and E. Keogh, "Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives," in *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 1190–1195, IEEE, 2020.

[19] H. Nguyen, K. P. Tran, S. Thomassey, and M. Hamad, "Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management," *International Journal of Information Management*, vol. 57, p. 102282, 2021.

[20] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pp. 4–11, 2014.

[21] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pp. 187–196, International World Wide Web Conferences Steering Committee, 2018.

[22] P. C. Ngo, A. A. Winarto, C. K. L. Kou, S. Park, F. Akram, and H. K. Lee, "Fence gan: Towards better anomaly detection," in *2019 IEEE 31St International Conference on tools with artificial intelligence (ICTAI)*, pp. 141–148, IEEE, 2019.

[23] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3009–3017, 2019.

[24] Linkedin, "Luminol," 2015. Access 15 May 2022.

[25] L. Wei, N. Kumar, V. N. Lolla, E. J. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Assumption-free anomaly detection in time series.," in *SSDBM*, vol. 5, pp. 237–242, 2005.

[26] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, 2015.

[27] J.-P. Eckmann, S. O. Kamphorst, D. Ruelle, *et al.*, "Recurrence plots of dynamical systems," *World Scientific Series on Nonlinear Science Series A*, vol. 16, pp. 441–446, 1995.

[28] E. Levina and P. Bickel, "The earth mover's distance is the mallows distance: Some insights from statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision (ICCV)*, vol. 2, pp. 251–256, IEEE, 2001.

[29] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[30] R. Wu and E. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[31] D. Yankov, E. Keogh, and U. Rebbapragada, "Disk aware discord discovery: finding unusual time series in terabyte sized datasets," *Knowledge and Information Systems*, vol. 17, no. 2, pp. 241–262, 2008.

# SECTION

## 2. UNPUBLISHED CONTENT

Building on top of the previous paper by exploiting Denoising Diffusion Probabilistic Models (DDPM) for anomaly detection. DDPMs have shown to outperform GANs in image synthesis and thus it should translate to better anomaly detection performance as well. However, due to time constraints this study will not be finished by the time of this publication.

# 3. SUMMARY AND CONCLUSIONS

This thesis presented TSI-GAN, a novel convolutional cycle-consistent GAN architecture that learns to reconstruct 2D-encoded complex 1D time-series data and produces reliable reconstruction errors for detecting non-trivial time series anomalies without any labels, and in real-time. Through a fully nonparametric design pipeline, we mitigate false alarms, which are a common issue in anomaly detection, by post-processing reconstruction error with a filtering technique and weighting strategy. We benchmark TSI-GAN against eight state-of-the-art baseline methods on 250 well-curated and harder-than-usual datasets. With a large winning margin over other methods, our approach is shown to be the best performer overall.

# VITA

Shyam Sundar Saravanan was born and raised in Chennai, India and obtained his Secondary School Leaving Certificate (SSLC) from BCS Jain Matriculation School on March 2013 and Higher Secondary School Certificate (HSC) from Kalaimagal Vidyalaya Matriculation Higher Secondary School on March 2015. Shyam attended MNM Jain Engineering College, earning a Bachelor of Engineering in Electronics and Communication Engineering (ECE) in November 2019. Shyam interned as a Data Science Intern at App Orchid Inc. during Spring 2022 and worked on Computer Vision and Natural Language Processing projects. Shyam earned a Master of Science in Computer Science from Missouri University of Science and Technology in May 2023.