# TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks

Alexander Geiger*
*MIT*
Cambridge, USA
geigera@mit.edu

Dongyu Liu*
*MIT*
Cambridge, USA
dongyu@mit.edu

Sarah Alnegheimish
*MIT*
Cambridge, USA
smish@mit.edu

Alfredo Cuesta-Infante
*Universidad Rey Juan Carlos*
Madrid, Spain
alfredo.cuesta@urjc.es

Kalyan Veeramachaneni
*MIT*
Cambridge, USA
kalyanv@mit.edu



***Abstract*—Time series anomalies can offer information relevant to critical situations facing various fields, from finance and aerospace to the IT, security, and medical domains. However, detecting anomalies in time series data is particularly challenging due to the vague definition of anomalies and said data's frequent lack of labels and highly complex temporal correlations. Current state-of-the-art unsupervised machine learning methods for anomaly detection suffer from scalability and portability issues, and may have high false positive rates. In this paper, we propose TadGAN, an unsupervised anomaly detection approach built on Generative Adversarial Networks (GANs). To capture the temporal correlations of time series distributions, we use LSTM Recurrent Neural Networks as base models for *Generators* and *Critics*. TadGAN is trained with cycle consistency loss to allow for effective time-series data reconstruction. We further propose several novel methods to compute reconstruction errors, as well as different approaches to combine reconstruction errors and *Critic* outputs to compute anomaly scores. To demonstrate the performance and generalizability of our approach, we test several anomaly scoring techniques and report the best-suited one. We compare our approach to 8 baseline anomaly detection methods on 11 datasets from multiple reputable sources such as NASA, Yahoo, Numenta, Amazon, and Twitter. The results show that our approach can effectively detect anomalies and outperform baseline methods in most cases (6 out of 11). Notably, our method has the highest averaged F1 score across all the datasets. Our code is open source and is available as a benchmarking tool.**

***Index Terms*—Anomaly detection, Generative adversarial network, Time series data**

## I. Introduction

The recent proliferation of temporal observation data has led to an increasing demand for time series anomaly detection in many domains, from energy and finance to healthcare and cloud computing. A time series anomaly is defined as a time point or period where a system behaves unusually [1]. Broadly speaking, there are two types of anomalies: A *point anomaly* is a single data point that has reached an unusual value, while a *collective anomaly* is a continuous sequence of data points that are considered anomalous as a whole, even if the individual data points may not be unusual [1].

Time series anomaly detection aims to isolate *anomalous* subsequences of **varied lengths** within time series. One of the simplest detection techniques is *thresholding*, which detects data points that exceed a normal range. However, many anomalies do not exceed any boundaries – for example, they may have values that are purportedly "normal," but are unusual at the specific time that they occur (i.e., *contextual anomalies*). These anomalies are harder to identify because the context of a signal is often unclear [1], [2].

Various statistical methods have been proposed to improve upon thresholding, such as Statistical Process Control (SPC) [3], in which data points are identified as anomalies if they fail to pass statistical hypothesis testing. However, a large amount of human knowledge is still required to set prior assumptions on the models.

Researchers have also studied a number of unsupervised machine learning-based approaches to anomaly detection. One popular method consists of segmenting a time series into subsequences (overlapping or otherwise) of a certain length and applying clustering algorithms to find outliers. Another learns

| Deep learning based method | Outperforms ARIMA, 1970 [4] |
|---|---|
| LSTM AutoEncoder, 2016 [5] | 5 |
| LSTM, 2018 [6] | 5 |
| MAD-GAN, 2019 [7] | 0 |
| MS Azure, 2019 [8] | 0 |
| DeepAR, 2019 [9] | 6 |
| TadGAN | **8** |

TABLE I
THE NUMBER OF WINS OF A PARTICULAR METHOD COMPARED WITH ARIMA, THE TRADITIONAL TIME SERIES FORECASTING MODEL, AGAINST AN APPROPRIATE METRIC (F1 SCORE) ON 11 REAL DATASETS.

* The two authors make equal contributions to this work. D. Liu and K. Veeramachaneni are the co-corresponding authors. 

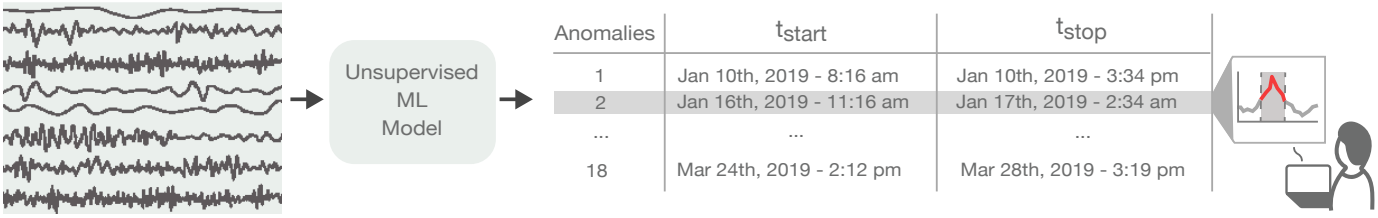


Fig. 1. An illustration of time series anomaly detection using unsupervised learning. Given a multivariate time series, the goal is to find out a set of anomalous time segments that have unusual values and do not follow the expected temporal patterns.

a model that either predicts or reconstructs a time series signal, and makes a comparison between the real and the predicted or reconstructed values. High prediction or reconstruction errors suggest the presence of anomalies.

Deep learning methods [10] are extremely capable of handling non-linearity in complex temporal correlations, and have excellent learning ability. For this reason, they have been used in a number of time series anomaly detection methods [6], [11], [12], including tools created by companies such as Microsoft [8]. Generative Adversarial Networks (GANs) [13] have also been shown to be very successful at generating time series sequences and outperforming state-of-the-art benchmarks [14]. Such a proliferation of methods invites the question: Do these new, complex approaches actually perform better than a simple baseline statistical method? To evaluate the new methods, we used 11 datasets (real and synthetic) that collectively have 492 signals and thousands of known anomalies to set-up a benchmarking system (see the details in Section VI and Table IV). We implemented 5 of the most recent deep learning techniques introduced between 2016 and 2019, and compared their performances with that of a baseline method from the 1970s, ARIMA. While some methods were able to beat ARIMA on 50% of the datasets, two methods failed to outperform it at all (c.f. Table I).

One of the foundational challenges of deep learning-based approaches is that their remarkable ability to fit data carries the risk that they could fit anomalous data as well. For example, autoencoders, using $L2$ objective function, can fit and reconstruct data extremely accurately - thus fitting the anomalies as well. On the other hand, GANs may be ineffective at learning the generator to fully capture the data's hidden distribution, thus causing false alarms. Here, we mix the two methods, creating a more nuanced approach. Additionally, works in this domain frequently emphasize improving the deep learning model itself. However, as we show in this paper, improving post-processing steps could aid significantly in reducing the number of false positives.

In this work, we introduce a novel GAN architecture, TadGAN, for the time series domain. We use TadGAN to reconstruct time series and assess errors in a contextual manner to identify anomalies. We explore different ways to compute anomaly scores based on the outputs from *Generators* and *Critics*. We benchmark our method against several well-known classical- and deep learning-based methods on eleven time series datasets. The detailed results can be found in Table IV.

The key contributions of this paper are as follows:

- We propose a novel unsupervised GAN-reconstruction-based anomaly detection method for time series data. In particular, we introduce a cycle-consistent GAN architecture for time-series-to-time-series mapping.
- We identify two time series similarity measures suitable for evaluating the contextual similarity between original and GAN-reconstructed sequences. Our novel approach leverages GAN's *Generator* and *Critic* to compute robust anomaly scores at every time step.
- We conduct an extensive evaluation using 11 time-series datasets from 3 reputable entities (NASA, Yahoo, and Numenta), demonstrating that our approach outperforms 8 other baselines. We further provide several insights into anomaly detection for time series data using GANs.
- We develop a benchmarking system for time series anomaly detection. The system is open-sourced and can be extended with additional approaches and datasets[1]. At the time of this writing, the benchmark includes 9 anomaly detection pipelines, 13 datasets, and 2 evaluation mechanisms.

The rest of this paper is structured as follows. We formally lay out the problem of time series anomaly detection in Section II. Section III presents an overview of the related literature. Section IV introduces the details of our GAN model. We describe how to use GANs for anomaly detection in Section V, and evaluate our proposed framework in Section VI. Finally, Section VII summarizes the paper and reports our key findings.

## II. Unsupervised Time Series Anomaly Detection

Given a time series $\mathbf{X} = (x^1, x^2, \cdots, x^T)$, where $x^i \in \mathbf{R}^{M \times 1}$ indicates $M$ types of measurements at time step $i$, the goal of *unsupervised* time series anomaly detection is to find a set of anomalous time segments $\boldsymbol{A_{seq}} = \{\boldsymbol{a^1_{seq}}, \boldsymbol{a^2_{seq}}, \cdots, \boldsymbol{a^k_{seq}}\}$, where $\boldsymbol{a^i_{seq}}$ is a continuous sequence of data points in time that show anomalous or unusual behaviors (Figure 1) – values within the segment that appear not to comply with the expected temporal behavior of the signal. A few aspects of this problem make it both distinct from and more difficult than time series classification [15] or supervised time series anomaly detection [16], as well as more pertinent to many industrial applications. We highlight them here:

[1]The software is available at github (https://github.com/signals-dev/Orion)

– No a priori knowledge of anomalies or possible anomalies: Unlike with supervised time series anomaly detection, we do not have any previously identified "known anomalies" with which to train and optimize the model. Rather, we train the model to to learn the time series patterns, ask it to detect anomalies, and then check whether the detector identified anything relevant to end users.
– Non availability of "normal baselines" : For many real-world systems, such as wind turbines and aircraft engines, simulation engines can produce a signal that resembles normal conditions, which can be tweaked for different control regimes or to account for degradation or aging. Such simulation engines are often physics-based and provide "*normal baselines*," which can be used to train models such that any deviations from them are considered anomalous. Unsupervised time series anomaly detection strategies do not rely on the availability of such baselines, instead learning time series patterns from real-world signals – signals that may themselves include anomalies or problematic patterns.
– Not all detected anomalies are problematic: Detected "*anomalies*" may not actually indicate problems, and could instead result from external phenomena (such as sudden shifts in environmental conditions), auxiliary information (such as the fact that a test run is being performed), or other variables that the algorithm did not consider, such as regime or control setting changes. Ultimately, it is up to the end user, the domain expert, to assess whether the anomalies identified by the model are problematic. Figure 1 highlights how a trained unsupervised machine learning model can be used in real time for the incoming data.
– No clear segmentation possible: Many signals, such as those associated with periodic time series, can be segmented – for example, an electrocardiogram signal (ECG) can be separated into similar segments that pertain to periods [16], [17]. The resulting segment clusters may reveal different collective patterns, along with anomalous patterns. We focus on signals that cannot be clearly segmented, making these approaches unfeasible. The length of $\boldsymbol{a^i}$ is also variable and is not known a priori, which further increases the difficulty.
– How do we evaluate these competing approaches? For this, we rely on several datasets that contain "known anomalies", the details of which are introduced in Section VI-A. Presumably, the "*anomalies*" are time segments that have been manually identified as such by some combination of algorithmic approaches and human expert annotation. These "*anomalies*" are used to evaluate the efficacy of our proposed unsupervised models. More details about this can be found in Section VI-B3.

## III. Related Work

Over the past several years, the rich variety of anomaly types, data types and application scenarios has spurred a range of anomaly detection approaches [1], [18]–[20]. In this section, we discuss some of the unsupervised approaches. The simplest of these are out-of-limit methods, which flag regions where values exceed a certain threshold [21], [22]. While these methods are intuitive, they are inflexible and incapable of detecting contextual anomalies. To overcome this, more advanced techniques have been proposed, namely proximity-based, prediction-based, and reconstruction-based anomaly detection (Table II).

| Methodology | Papers |
|---|---|
| Proximity | [23]–[25] |
| Prediction | [2], [6], [26], [27] |
| Reconstruction | [5], [28]–[30] |
| Reconstruction (GANs) | [7], [14], [31] |

TABLE II
UNSUPERVISED APPROACHES TO TIME SERIES ANOMALY DETECTION.

### A. Anomaly Detection for Time Series Data.

**Proximity-based methods** first use a distance measure to quantify similarities between objects – single data points for point anomalies, or fixed length sequences of data points for collective anomalies. Objects that are distant from others are considered anomalies. This detection type can be further divided into distance-based methods, such as K-Nearest Neighbor (KNN) [24] – which use a given radius to define neighbors of an object, and the number of neighbors to determine an anomaly score – and density-based methods, such as Local Outlier Factor (LOF) [23] and Clustering-Based Local Outlier Factor [25], which further consider the density of an object and that of its neighbors. There are two major drawbacks to applying proximity-based methods in time series data: (1) a priori knowledge about anomaly duration and the number of anomalies is required; (2) these methods are unable to capture temporal correlations.

**Prediction-based methods** learn a predictive model to fit the given time series data, and then use that model to predict future values. A data point is identified as an anomaly if the difference between its predicted input and the original input exceeds a certain threshold. Statistical models, such as ARIMA [26], Holt-Winters [26], and FDA [27], can serve this purpose, but are sensitive to parameter selection, and often require strong assumptions and extensive domain knowledge about the data. Machine learning-based approaches attempt to overcome these limitations. [2] introduce Hierarchical Temporal Memory (HTM), an unsupervised online sequence memory algorithm, to detect anomalies in streaming data. HTM encodes the current input to a hidden state and predicts the next hidden state. A prediction error is measured by computing the difference between the current hidden state and the predicted hidden state. Hundman et al. [6] propose Long Short Term Recurrent Neural Networks (LSTM RNNs), to predict future time steps and flag large deviations from predictions.

**Reconstruction-based methods** learn a model to capture the latent structure (low-dimensional representations) of the

given time series data and then create a synthetic reconstruction of the data. Reconstruction-based methods assume that anomalies lose information when they are mapped to a lower dimensional space and thereby cannot be effectively reconstructed; thus, high reconstruction errors suggest a high chance of being anomalous.

Principal Component Analysis (PCA) [28], a dimensionality-reduction technique, can be used to reconstruct data, but this is limited to linear reconstruction and requires data to be highly correlated and to follow a Gaussian distribution [29]. More recently, deep learning based techniques have been investigated, including those that use Auto-Encoder (AE) [30], Variational Auto-Encoder (VAE) [30] and LSTM Encoder-Decoder [5].

However, without proper regularization, these reconstruction-based methods can easily become overfitted, resulting in low performance. In this work, we propose the use of adversarial learning to allow for time series reconstruction. We introduce an intuitive approach for regularizing reconstruction errors. The trained *Generators* can be directly used to reconstruct more concise time series data – thereby providing more accurate reconstruction errors – while the *Critics* can offer scores as a powerful complement to the reconstruction errors when computing an anomaly score.

### B. Anomaly Detection Using GANs.

Generative adversarial networks can successfully perform many image-related tasks, including image generation [13], image translation [32], and video prediction [33], and researchers have recently demonstrated the effectiveness of GANs for anomaly detection in images [34], [35].

**Adversarial learning for images.** Schlegl et al. [36] use the Critic network in a GAN to detect anomalies in medical images. They also attempt to use the reconstruction loss as an additional anomaly detection method, and find the inverse mapping from the data space to the latent space. This mapping is done in a separate step, after the GAN is trained. However, Zenati et al. [37] indicate that this method has proven impractical for large data sets or real-time applications. They propose a bi-directional GAN for anomaly detection in tabular and image data sets, which allows for simultaneous training of the inverse mapping through an encoding network.

The idea of training both encoder and decoder networks was developed by Donahue et al. [38] and Dumoulin et al. [39], who show how to achieve bidirectional GANs by trying to match joint distributions. In an optimal situation, the joint distributions are the same, and the Encoder and Decoder must be inverses of each other. A cycle-consistent GAN was introduced by Zhu et al. [32], who have two networks try to map into opposite dimensions, such that samples can be mapped from one space to the other and vice versa.

**Adversarial learning for time series.** Prior GAN-related work has rarely involved time series data, because the complex temporal correlations within this type of data pose significant challenges to generative modeling. Three works published in 2019 are of note. First, to use GANs for anomaly detection in time series, Li et al. [7] propose using a vanilla GAN model to capture the distribution of a multivariate time series, and using the Critic to detect anomalies. Another approach in this line is BeatGAN [31], which is a Encoder and Decoder GAN architecture that allows for the use of the reconstruction error for anomaly detection in heartbeat signals. More recently, Yoon et al. [14] propose a time series GAN which adopts the same idea but introduces temporal embeddings to assist network training. However, their work is designed for time series representation learning instead of anomaly detection.

To the best of our knowledge, we are the first to introduce cycle-consistent GAN architectures for time series data, such that *Generators* can be directly used for time series reconstructions. In addition, we systematically investigate how to utilize *Critic* and *Generator* outputs for anomaly score computation. A complete framework of time series anomaly detection is introduced to work with GANs.

## IV. Adversarial Learning for Time Series Reconstruction

The core idea behind reconstruction-based anomaly detection methods is to learn a model that can encode a data point (in our case, a segment of a time series) and then decode the encoded one (i.e., reconstructed one). An effective model should not be able to reconstruct anomalies as well as "normal" instances, because anomalies will lose information during encoding. In our model, we learn two mapping functions between two domains $X$ and $Z$, namely $\mathcal{E} : X \rightarrow Z$ and $\mathcal{G} : Z \rightarrow X$ (Fig. 2). $X$ denotes the input data domain, describing the given training samples $\{(x_i^{1\cdots t})\}_{i=1}^{N}$, $x_i^{1\cdots t} \in X$. $Z$ represents the latent domain, where we sample random vectors $z$ to represent white noise. We follow a standard multivariate normal distribution, i.e., $z \sim \mathbb{P}_Z = \mathcal{N}(0, 1)$. For notational convenience we use $x_i$ to denote a time sequence of length $t$ starting at time step $i$. With the mapping functions, we can reconstruct the input time series: $x_i \rightarrow \mathcal{E}(x_i) \rightarrow \mathcal{G}(\mathcal{E}(x_i)) \approx \hat{x}_i$.

We propose leveraging adversarial learning approaches to obtain the two mapping functions $\mathcal{E}$ and $\mathcal{G}$. As illustrated in Fig. 2, we view the two mapping functions as *Generators*. Note that $\mathcal{E}$ is serving as an Encoder, which maps the time series sequences into the latent space, while $\mathcal{G}$ is serving as a Decoder, which transforms the latent space into the reconstructed time series. We further introduce two adversarial *Critics* (aka discriminators) $\mathcal{C}_x$ and $\mathcal{C}_z$. The goal of $\mathcal{C}_x$ is to distinguish between the real time series sequences from $X$ and the generated time series sequences from $\mathcal{G}(z)$, while $\mathcal{C}_z$ measures the performance of the mapping into latent space. In other words, $\mathcal{G}$ is trying to fool $\mathcal{C}_x$ by generating real-looking sequences. Thus, our high-level objective consists of two terms: (1) *Wasserstein losses* [40], to match the distribution of generated time series sequences to the data distribution in the target domain; and (2) *cycle consistency losses* [32], to prevent the contradiction between $\mathcal{E}$ and $\mathcal{G}$.

### A. Wasserstein Loss

The original formulation of GAN that applies the standard adversarial losses (Eq. 1) suffers from the mode collapse problem.

$$\mathcal{L} = \mathbb{E}_{x\sim\mathbb{P}_X}[\log\ \mathcal{C}_x(x)] + \mathbb{E}_{z\sim\mathbb{P}_Z}[\log(1-\mathcal{C}_x(\mathcal{G}(z)))] \quad (1)$$

where $\mathcal{C}_x$ produces a probability score from 0 to 1 indicating the realness of the input time series sequence. To be specific, the *Generator* tends to learn a small fraction of the variability of the data, such that it cannot perfectly converge to the target distribution. This is mainly because the *Generator* prefers to produce those samples that have already been found to be good at fooling the *Critic*, and is reluctant to produce new ones, even though new ones might be helpful to capture other "modes" in the data.

To overcome this limitation, we apply Wasserstein loss [40] as the adversarial loss to train the GAN. We make use of the Wasserstein-1 distance when training the *Critic* network. Formally, let $\mathbb{P}_X$ be the distribution over $X$. For the mapping function $\mathcal{G} : Z \rightarrow X$ and its *Critic* $C_x$, we have the following objective:

$$\min_{\mathcal{G}} \max_{\mathcal{C}_x\in\mathbf{C_x}} V_X(\mathcal{C}_x, \mathcal{G}) \quad (2)$$

with

$$V_X(\mathcal{C}_x, \mathcal{G}) = \mathbb{E}_{x\sim\mathbb{P}_X}[\mathcal{C}_x(x)] - \mathbb{E}_{z\sim\mathbb{P}_Z}[\mathcal{C}_x(\mathcal{G}(z)))] \quad (3)$$

where $\mathcal{C}_x \in \mathbf{C_x}$ which denotes the set of 1-Lipschitz continuous functions. K-Lipschitz continuous functions are defined as follows: $\|f(x_1) - f(x_2)\| \leq K\|x_1 - x_2\|, \forall x_1, x_2 \in dom\, f$. The Lipschitz continuous functions constrain the upper bound of the function, further smoothing the function. Therefore, the weights will not change dramatically when updated with gradient descent methods. This reduces the risk of gradient explosion, and makes the model training more stable and reliable. In addition, to enforce the 1-Lipschitz constraint during training, we apply a gradient penalty regularization term as introduced by Gulrajani et al. [41], which penalizes gradients not equal to 1 (cf. line 5).

Following a similar approach, we introduce a Wasserstein loss for the mapping function $\mathcal{E} : X \rightarrow Z$ and its *Critic* $C_z$. The objective is expressed as:

$$\min_{\mathcal{E}} \max_{\mathcal{C}_z\in\mathbf{C_z}} V_Z(\mathcal{C}_z, \mathcal{E}) \quad (4)$$

The purpose of the second *Critic* $C_z$ is to distinguish between random latent samples $z \sim \mathbb{P}_Z$ and encoded samples $\mathcal{E}(x)$ with $x \sim \mathbb{P}_X$. We present the model type and architecture for $\mathcal{E}$, $\mathcal{G}$, $\mathcal{C}_x$, $\mathcal{C}_z$ in section VI-B.

### B. Cycle Consistency Loss

The purpose of our GAN is to reconstruct the input time series: $x_i \rightarrow \mathcal{E}(x_i) \rightarrow \mathcal{G}(\mathcal{E}(x_i)) \approx \hat{x}_i$. However, training the GAN with adversarial losses (i.e., Wasserstein losses) alone cannot guarantee mapping individual input $x_i$ to a desired output $z_i$ which will be further mapped back to $\hat{x}_i$. To reduce the possible mapping function search space, we adapt cycle consistency loss to time series reconstruction, which was first introduced by Zhu et al. [32] for image translation tasks. We train the generative network $\mathcal{E}$ and $\mathcal{G}$ with the adapted cycle consistency loss by minimizing the L2 norm of the difference between the original and the reconstructed samples:

$$V_{L2}(\mathcal{E}, \mathcal{G}) = \mathbb{E}_{x\sim\mathbb{P}_X}[\|x - \mathcal{G}(\mathcal{E}(x))\|_2] \quad (5)$$

Considering that our target is anomaly detection, we use the L2 norm instead of L1 norm (the one used by Zhu et al. [32] for image translation) to emphasize the impacts of anomalous values. In our preliminary experiments, we observed that adding the backward consistency loss (i.e., $\mathbb{E}_{z\sim\mathbb{P}_z}[\|z - \mathcal{E}(\mathcal{G}(z))\|_2]$) did not improve performance.

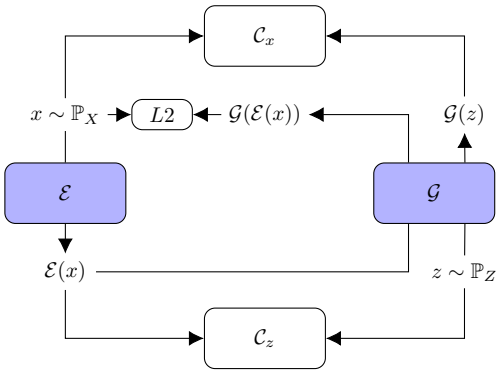


Fig. 2. Model architecture: *Generator* $\mathcal{E}$ is serving as an Encoder which maps the time series sequences into the latent space, while *Generator* $\mathcal{G}$ is serving as a Decoder that transforms the latent space into the reconstructed time series. *Critic* $\mathcal{C}_x$ is to distinguish between real time series sequences from $X$ and the generated time series sequences from $\mathcal{G}(z)$, whereas *Critic* $\mathcal{C}_z$ measures the goodness of the mapping into the latent space.

### C. Full Objective

Combining all of the objectives given in (3),(4) and (5) leads to the final MinMax problem:

$$\min_{\{\mathcal{E},\mathcal{G}\}} \max_{\{\mathcal{C}_x\in\mathbf{C_x},\mathcal{C}_z\in\mathbf{C_z}\}} V_X(\mathcal{C}_x, \mathcal{G}) + V_Z(\mathcal{C}_z, \mathcal{E}) + V_{L2}(\mathcal{E}, \mathcal{G}) \quad (6)$$

The full architecture of our model can be seen in Figure 2. The benefits of this architecture with respect to anomaly detection are twofold. First, we have a *Critic* $\mathcal{C}_x$ that is trained to distinguish between real and fake time series sequences, hence the score of the *Critic* can directly serve as an anomaly measure. Second, the two *Generators* trained with cycle consistency loss allow us to encode and decode a time series sequence. The difference between the original sequence and the decoded sequence can be used as a second anomaly detection measure. For detailed training steps, please refer to the pseudo code (cf. line 1–14). The following section will introduce the details of using TadGAN for anomaly detection.

## V. Time-series GAN for anomaly detection (TadGAN)

Let us assume that the given time series is $\mathbf{X} = (x^1, x^2, \cdots, x^T)$, where $x^i \in \mathbf{R}^{M\times 1}$ indicates $M$ types of measurements at time step $i$. For simplicity, we use $M = 1$

**Algorithm 1:** TadGAN

**Require:** $m$, batch size.
$epoch$, number of iterations over the data.
$n_{critic}$, number of iterations of the critic per epoch.
$\eta$, step size.

1 **for** *each epoch* **do**
2  **for** $\kappa = 0, \ldots, n_{critic}$ **do**
3  Sample $\{(x_i^{1\cdots t})\}_{i=1}^m$ from real data.
4  Sample $\{(z_i^{1\cdots k})\}_{i=1}^m$ from random.
5  $g_{w_{\mathcal{C}_x}} = \nabla_{w_{\mathcal{C}_x}}[\frac{1}{m}\sum_{i=1}^m \mathcal{C}_x(x_i) - \frac{1}{m}\sum_{i=1}^m \mathcal{C}_x(\mathcal{G}(z_i)) + gp(x_i, \mathcal{G}(z_i))]$
6  $w_{\mathcal{C}_x} = w_{\mathcal{C}_x} + \eta \cdot \texttt{adam}(w_{\mathcal{C}_x}, g_{w_{\mathcal{C}_x}})$
7  $g_{w_{\mathcal{C}_z}} = \nabla_{w_{\mathcal{C}_z}}[\frac{1}{m}\sum_{i=1}^m \mathcal{C}_z(z_i) - \frac{1}{m}\sum_{i=1}^m \mathcal{C}_z(\mathcal{E}(x_i)) + gp(z_i, \mathcal{E}(x_i))]$
8  $w_{\mathcal{C}_z} = w_{\mathcal{C}_z} + \eta \cdot \texttt{adam}(w_{\mathcal{C}_z}, g_{w_{\mathcal{C}_z}})$
9  **end**
10  Sample $\{(x_i^{1\cdots t})\}_{i=1}^m$ from real data.
11  Sample $\{(z_i^{1\cdots k})\}_{i=1}^m$ from random.
12  $g_{w_{\mathcal{G},\mathcal{E}}} = \nabla_{w_{\mathcal{G}},w_{\mathcal{E}}}[\frac{1}{m}\sum_{i=1}^m \mathcal{C}_x(x_i) - \frac{1}{m}\sum_{i=1}^m \mathcal{C}_x(\mathcal{G}(z_i)) + \frac{1}{m}\sum_{i=1}^m \mathcal{C}_z(z_i) - \frac{1}{m}\sum_{i=1}^m \mathcal{C}_z(\mathcal{E}(x_i)) + \frac{1}{m}\sum_{i=1}^m \|x_i - \mathcal{G}(\mathcal{E}(x_i))\|_2]$
13  $w_{\mathcal{G},\mathcal{E}}, = w_{\mathcal{G},\mathcal{E}} + \eta \cdot \texttt{adam}(w_{\mathcal{G},\mathcal{E}}, g_{w_{\mathcal{G},\mathcal{E}}})$
14 **end**
15 $X = \{(x_i^{1\cdots t})\}_{i=1}^n$
16 **for** $i = 1, \ldots, n$ **do**
17  $\hat{x}_i = \mathcal{G}(\mathcal{E}(x_i))$;
18  $RE(x_i) = f(x_i, \hat{x}_i)$;
19  $score = \alpha Z_{RE}(x_i) + (1-\alpha) Z_{\mathcal{C}_x}(\hat{x}_i)$
20 **end**

in the later description. Therefore, $\mathbf{X}$ is now a univariate time series and $x^i$ is a scalar. The same steps can be applied for multivariate time series (i.e., when $M > 1$).

To obtain the training samples, we introduce a sliding window with window size $t$ and step size $s$ to divide the original time series into N sub-sequences $X = \{(x_i^{1\cdots t})\}_{i=1}^N$, where $N = \frac{T-t}{s}$. In practice, it is difficult to know the ground truth, and anomalous data points are rare. Hence, we assume all the training sample points are normal. In addition, we generate $Z = \{(z_i^{1\cdots k})\}_{i=1}^N$ from a random space following normal distribution, where $k$ denotes the dimension of the latent space. Then, we feed $X$ and $Z$ to our GAN model and train it with the objective defined in (6). With the trained model, we are able to compute anomaly scores (or likelihoods) at every time step by leveraging the reconstruction error and *Critic* output (cf. line 15–20).

### A. Estimating Anomaly Scores using Reconstruction Errors

Given a sequence $x_i^{1\cdots t}$ of length $t$ (denoted as $x_i$ later), TadGAN generates a reconstructed sequence of the same length: $x_i \rightarrow \mathcal{E}(x_i) \rightarrow \mathcal{G}(\mathcal{E}(x_i)) \approx \hat{x}_i$. Therefore, for each time point $j$, we have a collection of reconstructed values $\{\hat{x}_i^q, i+q=j\}$ We take the median from the collection as the final reconstructed value $\hat{x}^j$. Note that in the preliminary experiments, we found that using the median achieved a better performance than using the mean. Now, the reconstructed time series is $(\hat{x}^1, \hat{x}^2, \cdots, \hat{x}^T)$. Here we propose three different types of functions (cf. line 18) for computing the reconstruction errors at each time step (assume the interval between neighboring time steps is the same).

**Point-wise difference.** This is the most intuitive way to define the reconstruction error, which computes the difference between the true value and the reconstructed value at every time step:

$$s_t = \left| x^t - \hat{x}^t \right| \tag{7}$$

**Area difference.** This is applied over windows of a certain length to measure the similarity between local regions. It is defined as the average difference between the areas beneath two curves of length $l$:

$$s_t = \frac{1}{2*l}\left| \int_{t-l}^{t+l} x^t - \hat{x}^t \, dx \right| \tag{8}$$

Although this seems intuitive, it is not often used in this context – however, we will show in our experiments that this approach works well in many cases. Compared with the point-wise difference, the area difference is good at identifying the regions where small differences exist over a long period of time. Since we are only given fixed samples of the functions, we use the trapezoidal rule to calculate the definite integral in the implementation.

**Dynamic time warping (DTW).** DTW aims to calculate the optimal match between two given time sequences [42] and is used to measure the similarity between local regions. We have two time series $X = (x_{t-1}, x_{t-l+1}, \ldots, x_{t+l})$ and $\hat{X} = (\hat{x}_{t-1}, \hat{x}_{t-l+1}, \ldots, \hat{x}_{t+l})$ and let $W \in \mathbf{R}^{2*l \times 2*l}$ be a matrix such that the $(i,j)$-th element is a distance measure between $x_i$ and $\hat{x}_j$, denoted as $w_k$. We want to find the warp path $W^* = (w_1, w_2, \ldots, w_K)$ that defines the minimum distance between the two curves, subject to boundary conditions at the start and end, as well as constraints on continuity and monotonicity. The DTW distance between time series $X$ and $\hat{X}$ is defined as follows:

$$s_t = W^* = \text{DTW}(X, \hat{X}) = \min_W \left[ \frac{1}{K} \sqrt{\sum_{k=1}^K w_k} \right] \tag{9}$$

Similar to area difference, DTW is able to identify the regions of small difference over a long period of time, but DTW can handle time shift issues as well.

### B. Estimating Anomaly Scores with Critic Outputs

During the training process, the *Critic* $C_x$ has to distinguish between real input sequences and synthetic ones. Because we use the Wasserstein-1 distance when training $C_x$, the outputs can be seen as an indicator of how real (larger value) or fake (smaller value) a sequence is. Therefore, once the *Critic* is

| | NASA | | Yahoo S5 | | | | NAB | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Property** | SMAP | MSL | A1 | A2 | A3 | A4 | Art | AdEx | AWS | Traf | Tweets |
| # SIGNALS | 53 | 27 | 67 | 100 | 100 | 100 | 6 | 5 | 17 | 7 | 10 |
| # ANOMALIES | 67 | 36 | 178 | 200 | 939 | 835 | 6 | 11 | 30 | 14 | 33 |
| point $(len = 1)$ | 0 | 0 | 68 | 33 | 935 | 833 | 0 | 0 | 0 | 0 | 0 |
| collective $(len > 1)$ | 67 | 36 | 110 | 167 | 4 | 2 | 6 | 11 | 30 | 14 | 33 |
| # ANOMALY POINTS | 54696 | 7766 | 1669 | 466 | 943 | 837 | 2418 | 795 | 6312 | 1560 | 15651 |
| # out-of-dist | 18126 | 642 | 861 | 153 | 21 | 49 | 123 | 15 | 210 | 86 | 520 |
| (% tot.) | 33.1% | 8.3% | 51.6% | 32.8% | 2.2% | 5.9% | 5.1% | 1.9% | 3.3% | 5.5% | 3.3% |
| # DATA POINTS | 562800 | 132046 | 94866 | 142100 | 168000 | 168000 | 24192 | 7965 | 67644 | 15662 | 158511 |
| IS SYNTHETIC? | | | | ✓ | ✓ | ✓ | ✓ | | | | |

TABLE III
DATASET SUMMARY: OVERALL THE BENCHMARK DATASET CONTAINS A TOTAL OF 492 SIGNALS AND 2349 ANOMALIES.

trained, it can directly serve as an anomaly measure for time series sequences.

Similar to the reconstruction errors, at time step $j$, we have a collection of *Critic* scores $(c_i^q, i + q = j)$. We apply kernel density estimation (KDE) on the collection and then take the maximum value as the smoothed value $c^j$ Now the *Critic* score sequence is $(c^1, c^2, \ldots, c^T)$. We show in our experiments that it is indeed the case that the *Critic* assigns different scores to anomalous regions compared to normal regions. This allows for the use of thresholding techniques to identify the anomalous regions.

### C. Combining Both Scores

The reconstruction errors $RE(x)$ and *Critic* outputs $\mathcal{C}_x(x)$ cannot be directly used together as anomaly scores. Intuitively, the larger $RE(x)$ and the smaller $\mathcal{C}_x(x)$ indicate higher anomaly scores. Therefore, we first compute the mean and standard deviation of $RE(x)$ and $\mathcal{C}_x(x)$, and then calculate their respective z-scores $Z_{RE}(x)$ and $Z_{\mathcal{C}_x}(x)$ to normalize both. Larger z-scores indicate high anomaly scores.

We have explored different ways to leverage $Z_{RE}(x)$ and $Z_{\mathcal{C}_x}(x)$. As shown in Table V (row 1–4), we first tested three types of $Z_{RE}(x)$ and $Z_{\mathcal{C}_x}(x)$ individually. We then explored two different ways to combine them (row 5 to the last row). First, we attempt to merge them into a single value $\boldsymbol{a}(x)$ with a convex combination (cf. line 19) [7], [36]:

$$\boldsymbol{a}(x) = \alpha Z_{RE}(x) + (1 - \alpha) Z_{\mathcal{C}_x}(x) \tag{10}$$

where $\alpha$ controls the relative importance of the two terms (by default $alpha = 0.5$). Second, we try to multiply both scores to emphasize the high values:

$$\boldsymbol{a}(x) = \alpha Z_{RE}(x) \odot Z_{\mathcal{C}_x}(x) \tag{11}$$

where $\alpha = 1$ by default. Both methods result in robust anomaly scores. The results are reported in Section VI-C.

### D. Identifying Anomalous Sequences

**Finding anomalous sequences with locally adaptive thresholding:** Once we obtain anomaly scores at every time step, thresholding techniques can be applied to identify anomalous sequences. We use sliding windows to compute thresholds, and empirically set the window size as $\frac{T}{3}$ and the step size as $\frac{T}{3*10}$. This is helpful to identify contextual anomalies whose contextual information is usually unknown. The sliding window size determines the number of historical anomaly scores to evaluate the current threshold. For each sliding window, we use a simple static threshold defined as 4 standard deviations from the mean of the window. We can then identify those points whose anomaly score is larger than the threshold as anomalous. Thus, continuous time points compose into anomalous sequences (or windows): $\{\boldsymbol{a}_{seq}^i, i = 1, 2, \ldots, K\}$, where $\boldsymbol{a}_{seq}^i = (\boldsymbol{a}_{start(i)}, \ldots, \boldsymbol{a}_{end(i)})$ .

**Mitigating false positives:** The use of sliding windows can increase recall of anomalies but may also produce many false positives. We employ an anomaly pruning approach inspired by Hundman et al. [6] to mitigate false positives. At first, for each anomalous sequence, we use the maximum anomaly score to represent it, obtaining a set of maximum values $\{\boldsymbol{a}_{max}^i, i = 1, 2, \ldots, K\}$. Once these values are sorted in descending order, we can compute the decrease percent $p^i = (\boldsymbol{a}_{max}^{i-1} - \boldsymbol{a}_{max}^i)/\boldsymbol{a}_{max}^{i-1}$. When the first $p^i$ does not exceed a certain threshold $\theta$ (by default $\theta = 0.1$), we reclassify all subsequent sequences (i.e., $\{\boldsymbol{a}_{seq}^j, i \leq j \leq K\}$) as normal.

## VI. EXPERIMENTAL RESULTS

### A. Datasets

To measure the performance of TadGAN, we evaluate it on multiple time series datasets. In total, we have collected 11 datasets (a total of 492 signals) across a variety of application domains. We use **spacecraft telemetry** signals provided by NASA[2], consisting of two datasets: Mars Science Laboratory (MSL) and Soil Moisture Active Passive (SMAP). In addition, we use **Yahoo S5** which contains four different sub-datasets [3] The A1 dataset is based on real production traffic to Yahoo computing systems, while A2, A3 and A4 are all synthetic datasets. Lastly, we use **Numenta Anomaly Benchmark**

[2]Spacecraft telemetry data: https://s3-us-west-2.amazonaws.com/telemanom/data.zip

[3]Yahoo S5 data can be requested here: https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70

**(NAB)**. NAB [43] includes multiple types of time series data from various application domains[4] We have picked five datasets: Art, AdEx, AWS, Traf, and Tweets.

Datasets from different sources contain different numbers of signals and anomalies, and locations of anomalies are known for each signal. Basic information for each dataset is summarized in Table III. For each dataset, we present the total number of signals and the number of anomalies pertaining to them. We also observe whether the anomalies in the dataset are single "point" anomalies, or one or more collections. In order to suss out the ease of anomaly identification, we measure how out-of-the-ordinary each anomaly point is by categorizing it as "*out-of-dist*" if it falls $4$ standard deviations away from the mean of all the data for a signal. As each dataset has some quality that make detecting its anomalies more challenging, this diverse selection will help us identify the effectiveness and limitations of each baseline.

### *B. Experimental setup*

*1) Data preparation:* For each dataset, we first normalize the data betweeen $[-1, 1]$. Then we find a proper interval over which to aggregate the data, such that we have several thousands of equally spaced points in time for each signal. We then set a window size $t = 100$ and step size $s = 1$ to obtain training samples for TadGAN. Because many signals in the Yahoo datasets contain linear trends, we apply a simple detrending function (which subtracts the result of a linear least-squares fit to the signal) before training and testing.

*2) Architecture:* In our experiments, inputs to TadGAN are time series sequences of length 100 (domain $X$), and the latent space (domain $Z$) is 20-dimensional. We use a 1-layer bidirectional Long Short-Term Memory (LSTM) with 100 hidden units as *Generator* $\mathcal{E}$, and a 2-layer bidirectional LSTM with 64 hidden units each as *Generator* $\mathcal{G}$, where dropout is applied. We add a 1-D convolutional layer for both *Critics*, with the intention of capturing local temporal features that can determine how anomalous a sequence is. The model is trained on a specific signal from one dataset for 2000 iterations, with a batch size of 64.

*3) Evaluation metrics:* We measure the performance of different methods using the commonly used metrics Precision, Recall and F1-Score. In many real-world application scenarios, anomalies are rare and usually window-based (i.e. a continuous sequence of points—see Sec. V-D). From the perspective of end-users, the best outcome is to receive timely true alarms without too many false positives (FPs), as these may waste time and resources. To penalize high FPs and reward the timely true alarms, we present the following window-based rules: (1) If a known anomalous window overlaps any predicted windows, a **TP** is recorded. (2) If a known anomalous window does not overlap any predicted windows, a **FN** is recorded. (3) If a predicted window does not overlap any labeled anomalous region, a **FP** is recorded. This method is also used in Hundman et al's work [6].

[4]NAB data: https://github.com/numenta/NAB/tree/master/data

*4) Baselines:* The baseline methods can be divided into three categories: prediction-based methods, reconstruction-based methods, and online commercial tools.

**ARIMA** (Prediction-based). An autoregressive integrated moving average (ARIMA) model is a popular statistical analysis model that learns autocorrelations in the time series for future value prediction. We use point-wise prediction errors as the anomaly scores to detect anomalies.

**HTM** (Prediction-based). Hierarchial Temporal Memory (HTM) [2] has shown better performance over many statistical analysis models in the Numenta Anomaly Benchmark. It encodes the current input to a hidden state and predicts the next hidden state. Prediction errors are computed as the differences between the predicted state and the true state, which are then used as the anomaly scores for anomaly detection.

**LSTM** (Prediction-based). The neural network used in our experiments consists of two LSTM layers with 80 units each, and a subsequent dense layer with one unit which predicts the value at the next time step (similar to the one used by Hundman et al. [6]). Point-wise prediction errors are used for anomaly detection.

**AutoEncoder** (Reconstruction-based). Our approach can be viewed as a special instance of "adversarial autoencoders" [44], $\mathcal{E} \circ \mathcal{G} : X \to X$. Thus, we compare our method with standard autoencoders with dense layers or LSTM layers [5]. The dense autoencoder consists of three dense layers with 60, 20 and 60 units respectively. The LSTM autoencoder contains two LSTM layers, each with 60 units. Again, a point-wise reconstruction error is used to detect anomalies.

**MAD-GAN** (Reconstruction-based). This method [7] uses a vanilla GAN along with an optimal instance searching strategy in latent space to support multivariate time series reconstruction. We use MAD-GAN to compute the anomaly scores at every time step and then apply the same anomaly detection method introduced in Sec. V-D to find anomalies.

**Microsoft Azure Anomaly Detector** (Commercial tool). Microsoft uses Spectral Residual Convolutional Neural Networks (SR-CNN) in which the models are applied serially [8]. The SR model is responsible for saliency detection, and the CNN is responsible for learning a discriminating threshold. The output of the model is a sequence of binary labels that is attributed to each timestamp.

**Amazon DeepAR** (Commercial tool). DeepAR is a probabilistic forecasting model with autoregressive recurrent networks [9]. We use this model in a similar manner to LSTM in that it is a prediction-based approach. Anomaly scores are presented as the regression errors which are computed as the distance between the median of the predicted value and true value.

### *C. Benchmarking Results*

**TadGAN outperformed all the baseline methods by having the highest averaged F1 score (0.7) across all the datasets.** Table IV ranks all the methods based on their averaged F1 scores (the last column) across the eleven datasets. The second (LSTM, 0.623) and the third (Arima, 0.599) best

| | NASA | | Yahoo S5 | | | | NAB | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Baseline** | MSL | SMAP | A1 | A2 | A3 | A4 | Art | AdEx | AWS | Traf | Tweets | Mean±SD |
| TadGAN | **0.623** | **0.704** | **0.8** | 0.867 | 0.685 | 0.6 | **0.8** | **0.8** | 0.644 | 0.486 | **0.609** | **0.700±0.123** |
| (P) LSTM | 0.46 | 0.69 | 0.744 | **0.98** | 0.772 | 0.645 | 0.375 | 0.538 | 0.474 | **0.634** | 0.543 | 0.623±0.163 |
| (P) Arima | 0.492 | 0.42 | 0.726 | 0.836 | **0.815** | **0.703** | 0.353 | 0.583 | 0.518 | 0.571 | 0.567 | 0.599±0.148 |
| (C) DeepAR | 0.583 | 0.453 | 0.532 | 0.929 | 0.467 | 0.454 | 0.545 | 0.615 | 0.39 | 0.6 | 0.542 | 0.555±0.130 |
| (R) LSTM AE | 0.507 | 0.672 | 0.608 | 0.871 | 0.248 | 0.163 | 0.545 | 0.571 | **0.764** | 0.552 | 0.542 | 0.549±0.193 |
| (P) HTM | 0.412 | 0.557 | 0.588 | 0.662 | 0.325 | 0.287 | 0.455 | 0.519 | 0.571 | 0.474 | 0.526 | 0.489±0.108 |
| (R) Dense AE | 0.507 | 0.7 | 0.472 | 0.294 | 0.074 | 0.09 | 0.444 | 0.267 | 0.64 | 0.333 | 0.057 | 0.353±0.212 |
| (R) MAD-GAN | 0.111 | 0.128 | 0.37 | 0.439 | 0.589 | 0.464 | 0.324 | 0.297 | 0.273 | 0.412 | 0.444 | 0.35±0.137 |
| (C) MS Azure | 0.218 | 0.118 | 0.352 | 0.612 | 0.257 | 0.204 | 0.125 | 0.066 | 0.173 | 0.166 | 0.118 | 0.219±0.145 |

TABLE IV
F1-SCORES OF BASELINE MODELS USING WINDOW-BASED RULES. COLOR ENCODES THE PERFORMANCE OF THE F1 SCORE. ONE IS EVENLY DIVIDED INTO 10 BINS, WITH EACH BIN ASSOCIATED WITH ONE COLOR. FROM DARK RED TO DARK BLUE, F1 SCORE INCREASES FROM 0 TO 1.

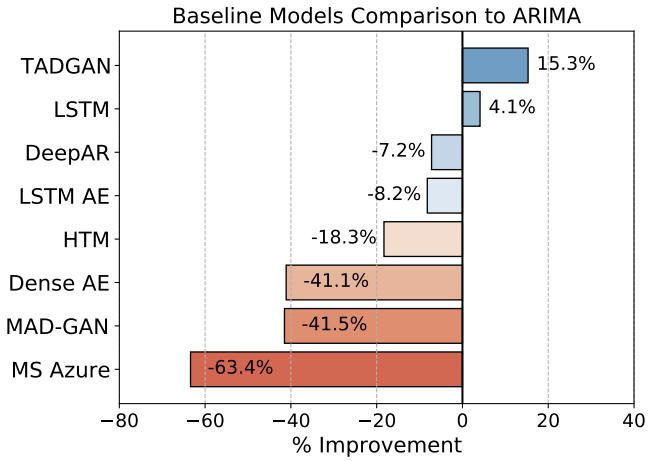


Fig. 3. Comparing average F1-Scores of baseline models across all datasets to ARIMA. The x-axis represents the percentage of improvement over the ARIMA score by each one of the baseline models.

are both prediction-based methods and TadGAN outperformed them by 12.46% and 16.86%, respectively, compared to the averaged F1 score.

**Baseline models in comparison to Arima.** Figure 3 depicts the performance of all baseline models with respect to Arima. It shows how much improvement in F1-Score is gained by each model. The F1-Score presented is the average across the eleven datasets. TadGAN achieves the best overall improvement with an over 15% improvement in score, followed by LSTM with a little over 4% improvement. It's worth noting that all the remaining models struggle to beat Arima.

**Synthetic data v.s. real-world datasets.** Although TadGAN outperforms all baselines on average, we note that it ranks below Arima when detecting anomalies within synthetic dataset with point anomalies. Specifically, TadGAN achieved an average of 0.717 while Arima scored an average of 0.784. However, TadGAN still produces competitive results in both scenarios.

**How well do AutoEncoders perform?** To view the superiority of GAN, we compare it to other reconstruction-based method such as LSTM AE, and Dense AE. One striking result is that the autoencoder alone does not perform well on point anomalies. We observe this as LSTM, AE, and Dense AE obtained an average F1 Score on A3 and A4 of 0.205 and 0.082 respectively, while TadGAN and MAD-GAN achieved a higher score of 0.643 and 0.527 respectively. One potential reason could be that AutoEncoders are optimizing L2 function and strictly attempt to fit the data, resulting in that anomalies get fitted as well. However, adversarial learning does not have this type of issue.

**TadGAN v.s. MadGAN.** Overall, TadGAN (0.7) outperformed Mad-GAN (0.219) significantly. This fully demonstrates the usage of forward cycle-consistency loss (Eq. 5) which prevents the contradiction between two *Generators* $\mathcal{E}$ and $\mathcal{G}$ and paves the most direct way to the optimal $z_i$ that corresponds to the testing sample $x_i$. Mad-GAN uses only vanilla GAN and does not include any regularization mechanisms to guarantee the mapping route $x_i \rightarrow z_i \rightarrow \hat{x}_i$. Their approach to finding the optimal $z_i$ is that they first sample a random $z$ from the latent space and then optimize it with the gradient descent algorithm by optimizing the anomaly detection loss.

### D. Ablation Study

We evaluated multiple variations of TadGAN, using different anomaly score computation methods for each (Sec. V-C). The results are summarized in Table V. Here we report some noteworthy insights.

**Using *Critic* alone is unstable**, because it has the lowest average F1 score (0.29) and the highest standard deviation (0.237). While only using *Critic* can achieve a good performance in some datasets, such as SMAP and Art, its performance may also be unexpectedly bad, such as in A2, A3, A4, AdEx, and Traf. No clear shared characteristics are identified among these five datasets (see Table III). For example, some datasets contain only collective anomalies (Traf, AdEx), while other datasets, like A3 and A4, have point anomalies as the majority types. One explanation could be that *Critic*'s behavior

| | NASA | | Yahoo S5 | | | | NAB | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Variation** | MSL | SMAP | A1 | A2 | A3 | A4 | Art | AdEx | AWS | Traf | Tweets | Mean+SD |
| Critic | 0.393 | 0.672 | 0.285 | 0.118 | 0.008 | 0.024 | 0.625 | 0 | 0.35 | 0.167 | 0.548 | 0.290±0.237 |
| Point | 0.585 | 0.588 | 0.674 | 0.758 | 0.628 | **0.6** | 0.588 | 0.611 | 0.551 | 0.383 | 0.571 | 0.594±0.086 |
| Area | 0.525 | 0.655 | 0.681 | 0.82 | 0.567 | 0.523 | 0.625 | 0.645 | 0.59 | 0.435 | 0.559 | 0.602±0.096 |
| DTW | 0.514 | 0.581 | 0.697 | 0.794 | 0.613 | 0.547 | 0.714 | 0.69 | 0.633 | 0.455 | 0.559 | 0.618±0.095 |
| Critic×Point | 0.619 | 0.675 | 0.703 | 0.75 | **0.685** | 0.536 | 0.588 | 0.579 | 0.576 | 0.4 | 0.59 | 0.609±0.091 |
| Critic+Point | 0.529 | 0.653 | **0.8** | 0.78 | 0.571 | 0.44 | 0.625 | 0.595 | **0.644** | 0.439 | 0.592 | 0.606±0.111 |
| Critic×Area | 0.578 | **0.704** | 0.719 | **0.867** | 0.587 | 0.46 | **0.8** | 0.6 | 0.6 | 0.4 | 0.571 | 0.625±0.131 |
| Critic+Area | 0.493 | 0.692 | 0.789 | 0.847 | 0.483 | 0.367 | 0.75 | 0.75 | 0.607 | 0.474 | 0.6 | 0.623±0.148 |
| Critic×DTW | **0.623** | 0.68 | 0.667 | 0.82 | 0.631 | 0.497 | 0.667 | 0.667 | 0.61 | 0.455 | 0.605 | **0.629±0.091** |
| Critic+DTW | 0.462 | 0.658 | 0.735 | 0.857 | 0.523 | 0.388 | 0.667 | **0.8** | 0.632 | **0.486** | **0.609** | 0.620±0.139 |
| Mean | 0.532 | 0.655 | 0.675 | 0.741 | 0.529 | 0.438 | 0.664 | 0.593 | 0.579 | 0.409 | 0.580 | |
| SD | 0.068 | 0.039 | 0.137 | 0.211 | 0.182 | 0.154 | 0.067 | 0.209 | 0.081 | 0.087 | 0.02 | |

TABLE V
F1-SCORES OF ALL THE VARIATIONS OF OUR MODEL.

is unpredictable when confronted with anomalies ($x \nsim \mathbb{P}_X$), because it is only taught to distinguish real time segments ($x \sim \mathbb{P}_X$) from generated ones.

**DTW outperforms the other two reconstruction error types slightly.** Among all variations, Critic×DTW has the best score (0.629). Further, its standard deviation is smaller than most of the other variations except for Point, indicating that this combination is more stable than others. Therefore, this combination should be the safe choice when encountering new datasets without labels.

**Combining *Critic* outputs and reconstruction errors does improve performance in most cases.** In all datasets except A4, combinations achieve the best performance. Let us take the MSL dataset as an example. We observe that when using DTW alone, the F1 score is $0.514$. Combining this with the *Critic score*, we obtain a score of $0.623$, despite the fact that the F1 score when using *Critic* alone is $0.393$. In addition, we find that after combining the Critic scores, the averaged F1 score improves for each of the individual reconstruction error computation methods. However, one interesting pattern is that for dataset A4, which consists mostly of point anomalies, using only point-wise errors achieve the best performance.

**Multiplication is a better option than convex combination.** Multiplication consistently leads to a higher averaged F1 score than convex combination does when using the same reconstruction error type (e.g., *Critic*×Point v.s. *Critic*+Point). Multiplication also has consistently smaller standard deviations. Thus, multiplication is the recommended way to combine reconstruction scores and *Critic* scores. This can be explained by the fact that multiplication can better amplify high anomaly scores.

### *E. Limitations and Discussion*

Here we compare our approach to one well-known GAN-based anomaly detection method [7]. However, there are many other GAN architectures tailored for time series reconstruction, such as Time-Series GAN [14]. Due to our modular design, any reconstruction-based algorithm of time series can employ our anomaly scoring method for time series anomaly detection. In the future, we plan to investigate various strategies for time series reconstruction and compare their performances to the current state-of-the-art. Moreover, it is worth understanding how better signal reconstruction affects the performance of anomaly detection. In fact, it is expected that better reconstruction might overfit to anomalies. Therefore, further experiments are required to understand the relationship between reconstruction and detecting anomalies.

## VII. CONCLUSION

In this paper, we presented a novel framework, TadGAN , that allows for time series reconstruction and effective anomaly detection, showing how GANs can be effectively used for anomaly detection in time series data. We explored point-wise and window-based methods to compute reconstruction errors. We further proposed two different ways to combine reconstruction errors and *Critic* outputs to obtain anomaly scores at every time step. We have also tested several anomaly-scoring techniques and reported the best-suited one in this work. Our experimental results showed that (1) TadGAN outperformed all the baseline methods by having the highest averaged F1 score across all the datasets, and showed superior performance over baseline methods in 6 out of 11 datasets; (2) window-based reconstruction errors outperformed the point-wise method; and (3) the combination of both reconstruction errors and critic outputs offers more robust anomaly scores, which help to reduce the number of false positives as well as increase the number of true positives. Finally, our code is open source and is available as a tool for benchmarking time series datasets for anomaly detection.


## VIII. ACKNOWLEDGEMENT

The authors are grateful to SES S.A. of Betzdorf, Luxembourg, for their financial and non financial support in this

work. Dr. Cuesta-Infante is funded by the Spanish Government research fundings RTI2018-098743-B-I00 (MICINN/FEDER) and Y2018/EMT-5062 (Comunidad de Madrid). Alnegheimish is supported by King Abdulaziz City for Science and Technology (KACST).

## REFERENCES


[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[2] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[3] D. Zheng, F. Li, and T. Zhao, "Self-adaptive statistical process control for anomaly detection in time series," *Expert Systems with Applications*, vol. 57, pp. 324–336, 2016.

[4] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[5] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," in *Anomaly Detection Workshop at 33rd ICML*, 2016.

[6] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding," in *Proc. of the 24th ACM SIGKDD*, 2018.

[7] 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*. Springer, 2019, pp. 703–716.

[8] 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 *Proc. of the 25th ACM SIGKDD*, 2019, pp. 3009–3017.

[9] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, 2019.

[10] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, pp. 949–961, 2017.

[11] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2015.

[12] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in *IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, 2017, pp. 140–145.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Proc. of Advances in neural information processing systems*, 2014, pp. 2672–2680.

[14] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Proc. of Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019, pp. 5509–5519.

[15] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[16] J. Qiu, Q. Du, and C. Qian, "Kpi-tsad: A time-series anomaly detector for kpi monitoring in cloud applications," *Symmetry*, vol. 11, no. 11, p. 1350, 2019.

[17] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ecg morphology and heartbeat interval features," *IEEE transactions on biomedical engineering*, vol. 51, no. 7, pp. 1196–1206, 2004.

[18] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[19] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLOS ONE*, vol. 11, no. 4, pp. 1–31, 04 2016.

[20] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: A survey," *International Journal of Information Management*, vol. 45, pp. 289–307, 2019.

[21] J.-A. Martínez-Heras and A. Donati, "Enhanced Telemetry Monitoring with Novelty Detection," *AI Magazine*, vol. 35, no. 4, p. 37, 2014.

[22] D. Decoste, "Automated Learning and Monitoring of Limit Functions," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 1997.

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proc. of the ACM SIGMOD*, 2000, pp. 93–104.

[24] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2002, pp. 15–27.

[25] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003.

[26] E. H. Pena, M. V. de Assis, and M. L. Proença, "Anomaly detection using forecasting methods arima and hwds," in *International Conference of the Chilean Computer Science Society (SCCC)*, 2013, pp. 63–66.

[27] J. M. Torres, P. G. Nieto, L. Alejano, and A. Reyes, "Detection of outliers in gas emissions from urban areas using functional data analysis," *Journal of hazardous materials*, vol. 186, no. 1, pp. 144–149, 2011.

[28] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of pca for traffic anomaly detection," in *Proc. of the 2007 ACM SIGMETRICS*, 2007, pp. 109–120.

[29] X. Dai and Z. Gao, "From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2226–2238, 2013.

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

[31] B. Zhou, S. Liu, B. Hooi, X. Cheng, and J. Ye, "BeatGAN: Anomalous Rhythm Detection using Adversarially Generated Time Series," in *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence, (IJCAI)*, 2019, pp. 4433–4439.

[32] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *IEEE Int. Conf. on Computer Vision (ICCV)*, oct 2017, pp. 2242–2251.

[33] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Proc. of Advances in neural information processing systems*, 2016, pp. 613–621.

[34] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks," *Medical Image Analysis*, vol. 54, pp. 30 – 44, 2019.

[35] L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft, "Anomaly detection with generative adversarial networks," 2018. [Online]. Available: https://openreview.net/forum?id=S1EfylZ0Z

[36] 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*. Springer, 2017, pp. 146–157.

[37] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially Learned Anomaly Detection," in *IEEE ICDM*, nov 2018, pp. 727–736.

[38] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial Feature Learning," in *IEEE ICLR*, 2017.

[39] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially Learned Inference," in *IEEE ICLR*, 2017.

[40] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. of the 34th ICML*, 2017, pp. 214–223.

[41] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," in *Proc. of the 31st Int. Conf. on Neural Information Processing Systems*, 2017, pp. 5769–5779.

[42] D. J. Bemdt and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series," in *AAAI Workshop on Knowledge Discovery in Databases*, Seattle, Washington, 1994.

[43] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark," in *Proc. of IEEE ICMLA*, 2015, pp. 38–44.

[44] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, "Adversarial autoencoders," in *Proc. of ICLR, Workshop Track*, 2016.