TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data

Shreshth Tuli
Imperial College London
London, UK
s.tuli20@imperial.ac.uk

Giuliano Casale
Imperial College London
London, UK
g.casale@imperial.ac.uk

Nicholas R. Jennings
Loughborough University
London, UK
n.r.jennings@lboro.ac.uk

Abstract

Efficient anomaly detection and diagnosis in multivariate time-series data is of great importance for modern industrial applications. However, building a system that is able to quickly and accurately pinpoint anomalous observations is a challenging problem. This is due to the lack of anomaly labels, high data volatility and the demands of ultra-low inference times in modern applications. Despite the recent developments of deep learning approaches for anomaly detection, only a few of them can address all of these challenges. In this paper, we propose TranAD, a deep transformer network based anomaly detection and diagnosis model which uses attention-based sequence encoders to swiftly perform inference with the knowledge of the broader temporal trends in the data. TranAD uses focus score-based self-conditioning to enable robust multi-modal feature extraction and adversarial training to gain stability. Additionally, model-agnostic meta learning (MAML) allows us to train the model using limited data. Extensive empirical studies on six publicly available datasets demonstrate that TranAD can outperform state-of-the-art baseline methods in detection and diagnosis performance with data and time-efficient training. Specifically, TranAD increases F1 scores by up to 17%, reducing training times by up to 99% compared to the baselines.

1 Introduction

Modern IT operations generate enormous amounts of high dimensional sensor data used for continuous monitoring and proper functioning of large-scale datasets. Traditionally, data mining experts have studied and highlighted data that do not follow usual trends to report faults. Such reports have been crucial for system management models for reactive fault tolerance and robust database design [47]. However, with the advent of big-data analytics and deep learning, this problem has become of interest to data mining researchers and to aid experts in handling increasing amounts of data. One particular use case is in artificial intelligence for Industry-4.0 databases, with a specific focus on service reliability [?] that has automated fault detection, recovery and management of modern systems. Detecting data-faults, or any type of behavior not conforming to the expected trends, is an active research discipline referred to as anomaly detection in multivariate time series [11]. Many data-driven industries, including ones related to distributed computing, Internet of Things (IoT), robotics and urban resource management [4, 46] are now adopting machine learning based unsupervised methods for anomaly detection.

Challenges. The problem of anomaly detection is becoming increasingly challenging in large-scale databases due to the increasing data modality [18, 28, 54]. In particular, the increasing number of sensors and devices in contemporary IoT platforms with increasing data volatility creates the requirement for significant amounts of data for accurate inference. However, due to the rising federated learning paradigm with geographically distant clusters, synchronizing databases across devices is expensive, causing limited data availability for training [48, 57]. Further, next-generation applications need ultra-fast inference speeds for quick recovery and optimal Quality of Service (QoS) [6, 49, 50]. Time-series databases are generated using several engineering artifacts (servers, robots, *etc.*) that interact with the environment, humans or other systems. As a result, the data often displays both stochastic and temporal trends [45]. It thus becomes crucial to distinguish outliers due to stochasticity and only pinpoint observations that do not adhere to the observed temporal trends. Moreover, the lack of labeled data and anomaly diversity makes the problem challenging as we cannot use supervised learning models, which have shown to be effective in other areas of data mining [12]. Finally, it is not only important to detect anomalies but also the root causes, *i.e.*, the specific data sources leading to abnormal behavior [23]. This complicates the problem further as we need to perform multi-class prediction (whether there is an anomaly and from which source if so) [60].

Existing solutions. The above discussed challenges have led to the development of a myriad of unsupervised learning solutions for automated anomaly detection. Researchers have developed reconstruction-based methods that predominantly aim to encapsulate the temporal trends and predict the time-series data in an unsupervised fashion, then use the deviation of the prediction with the ground-truth data as anomaly scores. Based on various extreme value analysis methods, such approaches classify timestamps with high anomaly scores as abnormal [4, 10, 14, 20, 28, 29, 45, 60, 62]. The way prior works generate a predicted time-series from a given one varies from one work to another. Traditional approaches, like SAND [10], use clustering and statistical analysis to detect anomalies. Contemporary methods like openGauss [30] and LSTM-NDT [20] use a Long-Short-Term-Memory (LSTM) based neural networks to forecast the data with an input time-series and a non-parametric dynamic thresholding approach for detecting anomalies from prediction errors. However, recurrent models like LSTMs are known to be slow and computationally expensive [4]. Recent state-of-the-art methods, like MTAD-GAT [62] and GDN [14], use deep neural networks with a time-series window as an input for more accurate predictions. However, as the inputs become more data-intensive, small constant size window inputs limit the detection performance of such models due to the restricted local context information given to the model [4]. There is a need for a model that is fast and can capture high-level trends with minimal overheads.

New insights. As noted above, recurrent models based on prior methods are not only slow and computationally expensive, but are

also unable to model long-term trends effectively [4, 14, 62]. This is because, at each timestamp, a recurrent model needs to first perform inference for all previous timestamps before proceeding further. Recent developments of the transformer models allow single-shot inference with the complete input series using position encoding [51]. Using transformers allows much faster detection compared to recurrent methods by parallelizing inference on GPUs [19]. However, transformers also provide the benefit of being able to encode large sequences with accuracy and training/inference times nearly agnostic to the sequence length [51]. Thus, we use transformers to grow the temporal context information sent to an anomaly detector without significantly increasing the computational overheads.

Our contributions. This work uses various tools, including Transformer neural networks and model-agnostic meta learning, as building blocks. However, each of these different technologies cannot be directly used and need necessary adaptations to create a generalizable model for anomaly detection. Specifically, we propose a transformer-based anomaly detection model (TranAD), that uses self-conditioning and an adversarial training process. Its architecture makes it fast for training and testing while maintaining stability with large input sequences. Simple transformer-based encoder-decoder networks tend to miss anomalies if the deviation is too small, *i.e.*, it is relatively close to normal data. One of our contributions is to show that this can be alleviated by an adversarial training procedure that can amplify reconstruction errors. Further, using self-conditioning for robust multi-modal feature extraction can help gain training stability and allow generalization [32]. This, with model-agnostic meta learning (MAML) helps keep optimum detection performance even with limited data [15], as we show later in the validation that methods with simple transformers underperform by over 11% compared to TranAD. We perform extensive empirical experiments on publicly available datasets to compare and analyze TranAD against the state-of-the-art methods. Our experiments show that TranAD is able to outperform baselines by increasing prediction scores by up to 17% while reducing training time overheads by up to 99%.

The rest of the paper is organized as follows. Section 2 overviews related work. Section 3 outlines the working of the TranAD model for multivariate anomaly detection and diagnosis. A performance evaluation of the proposed method is shown in Section 4. Section 5 presents additional analysis. Finally, Section 6 concludes.

2 Related Work

Time series anomaly detection is a long-studied problem in the VLDB community. The prior literature works on two types of time-series data: univariate and multivariate. For the former, various methods analyze and detect anomalies in time-series data with a single data source [34], while for the latter multiple time-series together [14, 45, 62].

Classical methods. Such methods for anomaly detection typically model the time-series distribution using various classical techniques like k-Mean clustering, Support Vector Machines (SVMs) or regression models [10, 28, 43, 52]. Other methods use wavelet theory or various signal transformation methods like Hilbert transform [25]. Other classes of methods use Principal Component Analysis (PCA), process regression or hidden Markov chains to model

time-series data [41]. The GraphAn technique [9] converts the time-series inputs to graphs and uses graph distance metrics to detect outliers. Another technique, namely isolation forest, uses an ensemble of several isolation trees that recursively partition the feature space for outlier detection [5, 31]. Finally, classical methods use variants of Auto-Regressive Integrated Moving Average (ARIMA) to model and detect anomalous behaviour [56]. However, auto-regression based approaches are rarely used for anomaly detection in high-order multivariate time series due to their inability to efficiently capture volatile time-series [1]. Other methods like SAND [10], CPOD [47] and Elle [28] utilize clustering and database read-write history to detect outliers.

Time-series discord discovery is another recently proposed method for fault prediction [16, 37, 58, 59]. Time series discords refer to the most unusual time series subsequences, *i.e.*, subsequences that are maximally different from all other subsequences in the same time series. A sub-class of methods uses matrix profiling or its variants for anomaly and motif discovery by detecting time series discords [16, 35, 63]. Many advances have been proposed to make matrix profiling techniques data and time efficient [21]. Other efforts aim to make matrix profiling applicable to diverse domains [64]. However, matrix profiling has many more uses than just anomaly detection and is considered to be slower than pure discord discovery algorithms [37]. A recent approach, MERLIN [37], uses a parameter-free version of time series discord discovery by iteratively comparing subsequences of varying length with their immediate neighbors. MERLIN is considered to be the state-of-the-art discord discovery approach with low overheads; hence, is regarded as one of the baselines in our experiments.

Deep Learning based methods. Most contemporary state-of-the-art techniques employ some form of deep neural networks. The LSTM-NDT [20] method relies on an LSTM based deep neural network model that uses the input sequence as training data and, for each input timestamp, forecasts data for the next timestamp. LSTMs are auto-regressive neural networks that learn order dependence in sequential data, where the prediction at each timestamp uses feedback from the output of the previous timestamp. This work also proposes a non-parametric dynamic error thresholding (NDT) strategy to set a threshold for anomaly labeling using moving averages of the error sequence. However, being a recurrent model, such models are slow to train in many cases with long input sequences. Further, LSTMs are often inefficient in modeling long temporal patterns, especially when the data is noisy [62].

The DAGMM [65] method uses a deep autoencoding Gaussian mixture model for dimension reduction in the feature space and recurrent networks for temporal modeling. This work predicts an output using a mixture of Gaussians, where the parameters of each Gaussian are given by a deep neural model. The autoencoder compresses an input datapoint into a latent space, that is then used by a recurrent estimation network to predict the next datapoint. The decoupled training of both networks allows the model to be more robust; however, it still is slow and unable to explicitly utilize inter-modal correlations [14]. The Omnianomaly [45] uses a stochastic recurrent neural network (similar to an LSTM-Variational Autoencoder [39]) and a planar normalizing flow to generate reconstruction probabilities. It also proposes an adjusted Peak Over

Threshold (POT) method for automated anomaly threshold selection that outperforms the previously used NDT approach. This work led to a significant performance leap compared to the prior art, but at the expense of high training times.

The Multi-Scale Convectional Recursive Encoder-Decoder (MS-CRED) [60] converts an input sequence window into a normalized two-dimensional image and then passes it through a ConvLSTM layer. This method is able to capture more complex inter-modal correlations and temporal information, however is unable to generalize to settings with insufficient training data. The MAD-GAN [29] uses an LSTM based GAN model to model the time-series distribution using generators. This work uses not only the prediction error, but also the discriminator loss in the anomaly scores. MTAD-GAT [62] uses a graph-attention network to model both feature and temporal correlations and pass it through a lightweight Gated-Recurrent-Unit (GRU) network that aids detection without severe overheads. Traditionally, attention operations perform input compression using convex combination where the weights are determined using neural networks. GRU is a simplified version of LSTM with a smaller parameter set and can be trained in limited data settings. The CAE-M [61] uses a convolutional autoencoding memory network, similar to MSCRED. It passes the time-series through a CNN with the output being processed by bidirectional LSTMs to capture long-term temporal trends. Such recurrent neural network-based models have been shown to have high computation costs and low scalability for high dimensional datasets [4].

More recent works such as USAD [4], GDN [14] and open-Gauss [30] do not use resource-hungry recurrent models, but only attention-based network architectures to improve training speeds. The USAD method uses an autoencoder with two decoders with an adversarial game-style training framework. This is one of the first works that focus on low overheads by using a simple autoencoder and can achieve a several-fold reduction in training times compared to the prior art. The Graph Deviation Network (GDN) approach learns a graph of relationships between data modes and uses attention-based forecasting and deviation scoring to output anomaly scores. The openGauss approach uses a tree-based LSTM that has lower memory and computational footprint and allows capturing temporal trends even with noisy data. However, due to the small window as an input and the use of simple or no recurrent models, the latest models are unable to capture long-term dependencies effectively.

The recently proposed HitAnomaly [19] method uses vanilla transformers as encoder-decoder networks, but is only applicable to natural-language log data and not appropriate for generic continuous time-series data as inputs. In our experiments, we compare TranAD against the state-of-the-art methods MERLIN, LSTM-NDT, DAGMM, OmniAnomaly, MSCRED, MAD-GAN, USAD, MTAD-GAT, CAE-M and GDN. These methods have shown superiority in anomaly detection and diagnosis, but complement one another in terms of performance across different time-series datasets. Out of these, only USAD aims to reduce training times, but does this to a limited extent. Just like reconstruction based prior work [4, 29, 45, 60, 61], we develop a TranAD model that learns broad level trends using training data to find anomalies in test data. We specifically improve anomaly detection and diagnosis performance with also reducing the training times in this work.

3 Methodology

3.1 Problem Formulation

We consider a multivariate time-series, which is a timestamped sequence of observations/datapoints of size T

$$\mathcal{T} = \{x_1, \ldots, x_T\},$$

where each datapoint x_t is collected at a specific timestamp t and $x_t \in \mathbb{R}^m$, $\forall t$. Here, the univariate setting is a particular case where $m = 1$. We now define the two problems of anomaly detection and diagnosis.

Anomaly Detection: Given a training input time-series $\mathcal{T}$, for any unseen test time-series $\hat{\mathcal{T}}$ of length $\hat{T}$ and same modality as the training series, we need to predict $\mathcal{Y} = \{y_1, \ldots, y_{\hat{T}}\}$, where we use $y_t \in \{0, 1\}$ to denote whether the datapoint at the t-th timestamp of the test set is anomalous (1 denotes an anomalous datapoint).

Anomaly Diagnosis: Given the above training and test time-series, we need to predict $\mathcal{Y} = \{y_1, \ldots, y_{\hat{T}}\}$, where $y_t \in \{0, 1\}^m$ to denote which of the modes of the datapoint at the t-th timestamp are anomalous.

3.2 Data Preprocessing

To make our model more robust, we normalize the data and convert it to time-series windows, both for training and testing. We normalize the time-series as:

$$x_t \leftarrow \frac{x_t - \min(\mathcal{T})}{\max(\mathcal{T}) - \min(\mathcal{T}) + \epsilon'}, \tag{1}$$

where $\min(\mathcal{T})$ and $\max(\mathcal{T})$ are the mode wise minimum and maximum vectors in the training time-series. ϵ' is a small constant vector to prevent zero-division. Knowing the ranges a-priori, we normalize the data to get it in the range $[0, 1)$.

To model the dependence of a data point x_t at a timestamp t, we consider a local contextual window of length K as

$$W_t = \{x_{t-K+1}, \ldots, x_t\}.$$

We use replication padding for $t < K$ and convert an input time series $\mathcal{T}$ to a sequence of sliding windows $\mathcal{W} = \{W_1, \ldots, W_T\}$. Replication padding, for each $t < K$, appends to the window W_t a constant vector $\{x_t, \ldots, x_t\}$ of length $K - t$ to maintain the window length of K for each t. Instead of using $\mathcal{T}$ as training input, we use $\mathcal{W}$ for model training and $\hat{\mathcal{W}}$ (corresponding to $\hat{\mathcal{T}}$) as the test series. This is a common practice in prior work [4, 45] as it allows us to give a datapoint with its local context instead of a standalone vector, and hence is used in our model. We also consider the time slice until the current timestamp t of a series $\mathcal{T}$ and denote it as C_t.

Now, instead of directly predicting the anomaly label y_t for each input window W_t, we shall first predict an anomaly score s_t for this window. Using anomaly scores for the past input windows, we calculate a threshold value D, above which we label the input window as anomalous, thus $y_t = \mathbb{1}(s_t \geq D)$. To calculate the anomaly score s_t, we reconstruct the input window as O_t and use the deviation between W_t and O_t. For the sake of simplicity and without loss of generality, we shall use W, C, O and s for the rest of the discussion.
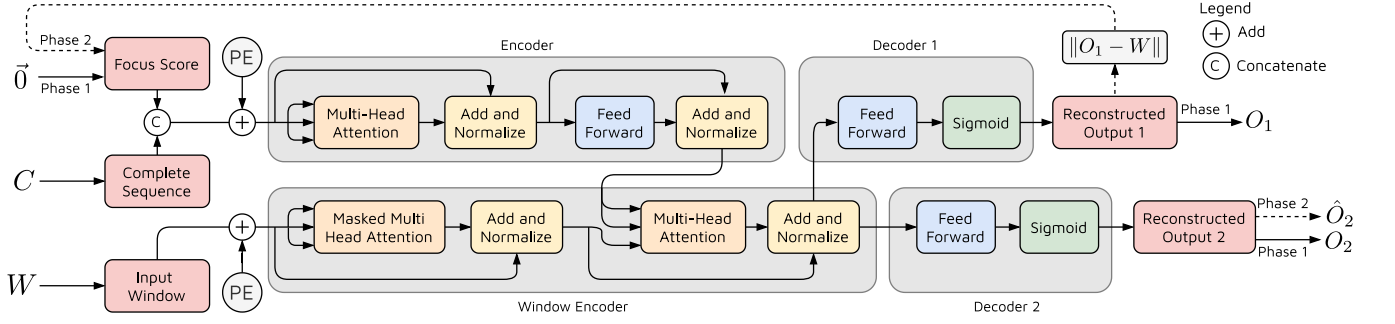
Figure 1: The TranAD Model.

3.3 Transformer Model

Transformers are popular deep learning models that have been used in various natural language and vision processing tasks [51]. However, we use insightful refactoring of the transformer architecture for the task of anomaly detection in time-series data. Just like other encoder-decoder models, in a transformer, an input sequence undergoes several attention-based transformations. Figure 1 shows the architecture of the neural network used in TranAD. The encoder encodes the complete sequence until the current timestamp C with a focus score (more details later). The window encoder uses this to create an encoded representation of the input window W, which is then passed to two decoders to create its reconstruction.

We now provide details on the working of TranAD. A multivariate sequence like W or C is transformed first into a matrix form with modality m. We define scaled-dot product attention [51] of three matrices Q (query), K (key) and V (value):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{m}}\right)V. \quad (2)$$

Here, the softmax forms the convex combination weights for the values in V, allowing us to compress the matrix V into a smaller representative embedding for simplified inference in the downstream neural network operations. Unlike traditional attention operation, the scaled-dot product attention scales the weights by a $\sqrt{m}$ term to reduce the variance of the weights, facilitating stable training [51]. For input matrices Q, K and V, we apply Multi-Head Self Attention [51] by first passing it through h (number of heads) feed-forward layers to get Q_i, K_i and V_i for $i \in \{1, \ldots, h\}$, and then applying scaled-dot product attention as

$$\text{MultiHeadAtt}(Q, K, V) = \text{Concat}(H_1, \ldots, H_h),$$
$$\text{where } H_i = \text{Attention}(Q_i, K_i, V_i). \quad (3)$$

Multi-Head Attention allows the model to jointly attend to information from different representation sub-spaces at different positions. In addition, we use position encoding of the input matrices as defined in [51].

As GAN models have been shown to perform well in characteristic tasks of whether an input is anomalous or not, we leverage a time-efficient GAN style adversarial training method. Our model consists of two transformer encoders and two decoders (Figure 1). We consider the model inference in two phases. We first take the W and C pair as an input and a focus score F (initially a zero matrix of the dimension of W, more details in the next subsection). We

broadcast F to match the dimension of W, with appropriate zero-padding and concatenate the two. We then apply position encoding and obtain the input for the first encoder, say I_1. The first encoder performs the following operations

$$I_1^1 = \text{LayerNorm}(I_1 + \text{MultiHeadAtt}(I_1, I_1, I_1)),$$
$$I_1^2 = \text{LayerNorm}(I_1^1 + \text{FeedForward}(I_1^1)). \quad (4)$$

Here, $\text{MultiHeadAtt}(I_1, I_1, I_1)$ denotes the multi-head self attention operation for the input matrix I_1 and $+$ denotes matrix addition. The above operations generate attention weights using the input time-series windows and the complete sequence to capture temporal trends within the input sequences. These operations enable the model to infer over multiple batches of the time-series windows in parallel as the neural network, at each timestamp, does not depend on the output of a previous timestamp, significantly improving the training time of the proposed method. For the window encoder, we apply position encoding to the input window W to get I_2. We modify the self-attention in the window encoder to mask the data at subsequent positions. This is done to prevent the decoder from looking at the datapoints for future timestamp values at the time of training as all data W and C is given at once to allow parallel training. The window encoder performs the following operations

$$I_2^1 = \text{Mask}(\text{MultiHeadAtt}(I_2, I_2, I_2)),$$
$$I_2^2 = \text{LayerNorm}(I_2 + I_2^1), \quad (5)$$
$$I_2^3 = \text{LayerNorm}(I_2^2 + \text{MultiHeadAtt}(I_1^2, I_1^2, I_2^2)).$$

The encoding of the complete sequence I_1^2 is used as value and keys by the window encoder for the attention operation using the encoded input window as the query matrix. The motivation behind the operations in (5) is similar to the one for (4); however, here we apply masking of the window input to hide the window sequences for future timestamps in the same input batch. As the complete input sequence up to the t-th timestamp is given to the model as an input; it allows the model to encapsulate and leverage a larger context compared to a bounded, limited one as in prior art [4, 45, 62]. Finally, we use two identical decoders which perform the operation

$$O_i = \text{Sigmoid}(\text{FeedForward}(I_2^3)), \quad (6)$$

where $i \in \{1, 2\}$ for the first and second decoder respectively. The Sigmoid activation is used to generate an output in the range $[0, 1]$, to match the normalized input window. Thus, the TranAD model takes the inputs C and W to generate two outputs O_1 and O_2.

Algorithm 1 The TranAD training algorithm

Require:
    Encoder E, Decoders D_1 and D_2
    Dataset used for training $\mathcal{W}$
    Evolutionary hyperparameter ϵ
    Iteration limit N
 1: Initialize weights E, D_1, D_2
 2: $n \leftarrow 0$
 3: **do**
 4:    **for**$(t = 1$ to $T)$
 5:      $O_1, O_2 \leftarrow D_1(E(W_t, \vec{0})), D_2(E(W_t, \vec{0}))$
 6:      $\hat{O}_2 \leftarrow D_2(E(W_t, \|O_1 - W_t\|_2))$
 7:      $L_1 = \epsilon^{-n}\|O_1 - W_t\|_2 + (1 - \epsilon^{-n})\|\hat{O}_2 - W_t\|_2$
 8:      $L_2 = \epsilon^{-n}\|O_2 - W_t\|_2 - (1 - \epsilon^{-n})\|\hat{O}_2 - W_t\|_2$
 9:      Update weights of E, D_1, D_2 using L_1, L_2
10:     $n \leftarrow n + 1$
11:    Meta-Learn weights E, D_1, D_2 using a random batch
12: **while** $n < N$

3.4 Offline Two-Phase Adversarial Training

We now describe the adversarial training process and the two-phase inference approach in the TranAD model, summarized in Algorithm 1.

Phase 1 - Input Reconstruction. The Transformer model enables us to predict the reconstruction of each input time-series window. It does this by acting as an encoder-decoder network at each timestamp. However, traditional encoder-decoder models often are unable to capture short-term trends and tend to miss anomalies if the deviations are too small [29]. To tackle this challenge, we develop an auto-regressive inference style that predicts the reconstructed window in two-phases. In the first phase, the model aims to generate an approximate reconstruction of the input window. The deviation from this inference, referred to as the *focus score* mentioned previously, facilitates the attention network inside the Transformer Encoder to extract temporal trends, focusing on the sub-sequences where the deviations are high. Thus, the output of the second phase is conditioned on the deviations generated from the first phase. Thus, in the first stage, the encoders convert the input window $W \in \mathbb{R}^{K \times m}$ (with focus score $F = [0]_{K \times m}$) to a compressed latent representation I_2^3 using context-based attention as in a common transformer model. This compressed representation is then converted to generate outputs O_1 and O_2 via Eq. (6).

Phase 2 - Focused Input Reconstruction. In the second phase, we use the reconstruction loss for the first decoder as a focus score. Having the focus matrix for the second phase $F = L_1$, we rerun model inference to obtain the output of the second decoder as $\hat{O}_2$.

The focus score generated in the first phase indicates the deviations of the reconstructed output from the given input. This acts as a prior to modify the attention weights in the second phase and gives higher neural network activation to specific input sub-sequences to extract short-term temporal trends. We refer to this approach as "self-conditioning" in the rest of the paper. This two-phase auto-regressive inference style has a three-fold benefit. First, it amplifies the deviations, as the reconstruction error acts as an activation in the attention part of the Encoder in Figure 1, to generate an anomaly

score, simplifying the fault-labeling task (discussed in Section 3.5). Second, it prevents false positives by capturing short-term temporal trends in the Window Encoder in Figure 1. Third, the adversarial style training is known to improve generalizability and make the model robust to diverse input sequences [4].

Evolving Training Objective. The above-described model is bound to suffer from similar challenges as in other adversarial training frameworks. One of the critical challenges is maintaining training stability. To tackle this, we design an adversarial training procedure that uses outputs from two separate decoders (Decoders 1 and 2 in Figure 1). Initially, both decoders aim to independently reconstruct the input time-series window. As in [45] and [39], we define the reconstruction loss for each decoder using the L2-norm using the outputs of the first phase:

$$L_1 = \|O_1 - W\|_2, \\ L_2 = \|O_2 - W\|_2. \tag{7}$$

We now introduce the adversarial loss that uses outputs of the second phase. Here, the second decoder aims to distinguish between the input window and the candidate reconstruction generated by the first decoder in phase 1 (using the focus scores) by maximizing the difference $\|\hat{O}_2 - W\|_2$. On the other hand, the first decoder aims to fool the second decoder by aiming to create a degenerate focus score (a zero vector) by perfectly reconstructing the input (*i.e.*, $O_1 = W$). This pushes the decoder 2, in this phase, to generate the same output as O_2 which it aims to match the input in phase 1. This means the training objective is

$$\min_{\text{Decoder1}} \max_{\text{Decoder2}} \|\hat{O}_2 - W\|_2. \tag{8}$$

Thus, the objective of the first decoder is to minimize the reconstruction error of this self-conditioned output, whereas the objective of the second one is to maximize the same. We realize this by using the loss as:

$$L_1 = +\|\hat{O}_2 - W\|_2, \\ L_2 = -\|\hat{O}_2 - W\|_2. \tag{9}$$

Now that we have loss functions for both phases, we need to determine the cumulative loss for each decoder. We thus use an evolutionary loss function that combines the reconstruction and adversarial loss functions from the two phases as

$$L_1 = \epsilon^{-n}\|O_1 - W\|_2 + (1 - \epsilon^{-n})\|\hat{O}_2 - W\|_2, \\ L_2 = \epsilon^{-n}\|O_2 - W\|_2 - (1 - \epsilon^{-n})\|\hat{O}_2 - W\|_2, \tag{10}$$

where n is the training epoch and ϵ is a training parameter close to one (lines 7-8 in Alg. 1). Initially, the weight given to the reconstruction loss is high. This is to ensure stable training when the outputs of the decoders are poor reconstructions of the input window. With poor reconstructions, the focus scores used in the second phase would be unreliable; and hence, cannot be utilized as a prior to indicating reconstructions that are far from the input sequence. Thus, the adversarial loss is given a low weight in the initial part of the process to avoid destabilizing model training. As reconstructions become closer to the input windows, and focus scores become more precise, the weight to the adversarial loss is increased. As loss curves in the neural network training process typically follow exponential function, we use weights of the form

---

Algorithm 2 The TranAD testing algorithm

Require:

    Trained Encoder E, Decoders D_1 and D_2

    Test Dataset $\hat{\mathcal{W}}$

1: **for**$(t = 1$ to $\hat{T})$

2:     $O_1, O_2 \leftarrow D_1(E(\hat{W}_t, \vec{0})), D_2(E(\hat{W}_t, \vec{0}))$

3:     $\hat{O}_2 \leftarrow D_1(E(\hat{W}_t, \|O_1 - W\|_2)), D_2(E(\hat{W}_t, \|O_1 - W\|_2))$

4:     $s = \frac{1}{2}\|O_1 - \hat{W}\|_2 + \frac{1}{2}\|\hat{O}_2 - \hat{W}\|_2$

5:     $y_i = \mathbb{1}(s_i \geq \text{POT}(s_i))$

6:     $y = \bigvee_i y_i$

Figure 2: Visualization of anomaly prediction.

ϵ^{-n} in the training process with a small positive constant parameter ϵ.

As the training process does not assume that the data is available sequentially (as in an online process), the complete time-series can be split into (W, C) pairs and the model can be trained using input batches. Masked multi-head attention allows us to run this in parallel across several batches and speed up the training process.

Meta Learning. Finally, our training loop uses model-agnostic meta learning (MAML), a few-shot learning model for fast adaptation of neural networks [15]. This helps our TranAD model learn temporal trends in the input training time-series with limited data. In each training epoch, a gradient update for neural network weights (without loss in generality assume θ) can be simply written as

$$\theta' \leftarrow \theta - \alpha\nabla_\theta L(f(\theta)), \tag{11}$$

where α, $f(\cdot)$ and $L(\cdot)$ are learning rate, abstract representation of the neural network and loss function respectively. Now, at the end of each epoch we perform meta-learning step as

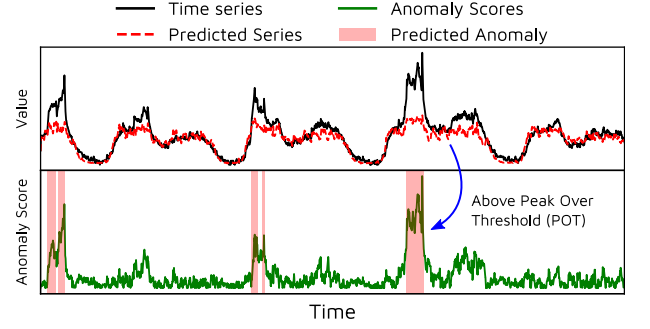$$\theta \leftarrow \theta - \beta\nabla_\theta L(f(\theta')). \tag{12}$$

The meta-optimization is performed with a meta step-size β, over the model weights θ where the objective is evaluated using the updated weights θ'. Prior work has shown that this allows models to be trained quickly with limited data [15]. We encapsulate this in a single line in Algorithm 1 (line 11).

3.5 Online Inference, Anomaly Detection and Diagnosis

We now describe the inference procedure using the trained transformer model (summarized in Algorithm 2). For an unseen data $(\hat{W}, \hat{C})$, the anomaly score is defined as

$$s = \frac{1}{2}\|O_1 - \hat{W}\|_2 + \frac{1}{2}\|\hat{O}_2 - \hat{W}\|_2. \tag{13}$$

The inference at test time runs again in two phases and hence we get a single pair of reconstruction $(O_1, \hat{O}_2)$ (lines 2 and 3 in Alg. 2). At test time, we only consider the data until the current timestamp and hence this operation runs sequentially in an online fashion. Once we have the anomaly scores for a timestamp for each dimension s_i, we label the timestamp anomalous if this score is greater than a threshold. As is common in prior work [9, 20, 45], for fair comparison, we use the Peak Over Threshold (POT) [44] method to choose the threshold automatically and dynamically. In essence, this is a statistical method that uses "extreme value theory" to fit the data distribution with a Generalized Pareto Distribution
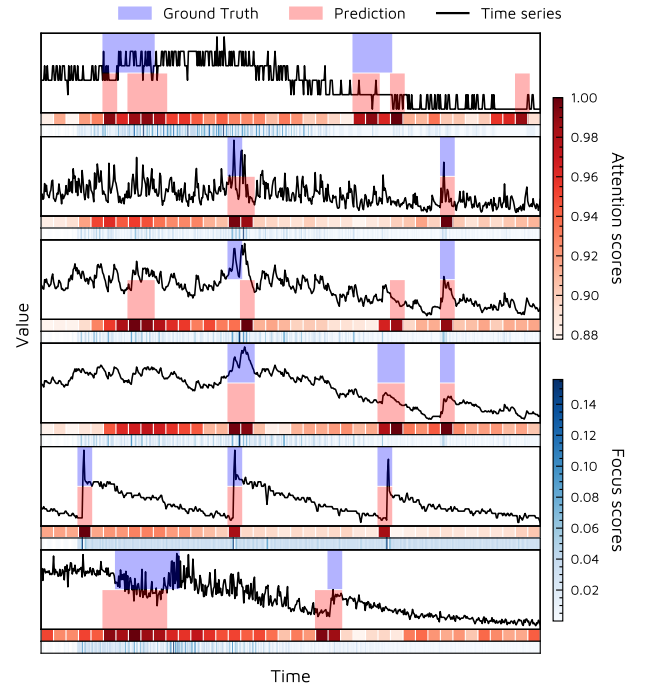
Figure 3: Visualization of focus and attention scores.

and identify appropriate value at risk to dynamically determine threshold values. We also tested with another popular EVT method, namely annual maximum (AM) [7]; however, we have observed 7.2% higher F1 scores on an average for TranAD with POT than AM. Anomaly diagnosis label for each dimension i (y_i) and detection (y) results is defined as

$$y_i = \mathbb{1}(s_i \geq \text{POT}(s_i)),$$
$$y = \bigvee_i y_i. \tag{14}$$

Thus, we label the current timestamp anomalous if any of the m dimensions is anomalous (lines 5-6 in Alg. 2). Figure 2 illustrates this process for a sample time-series.

Impact of Attention and Focus Scores. Figure 3 visualizes the attention and focus scores for the TranAD model trained on the SMD dataset (details in Section 4.1). We show the time-series, the average attention weights for each window (averaged over multiple heads) and focus scores for the first six dimensions of the dataset. It is apparent that the focus scores are highly correlated with the

peaks and noise in the data. There is also a high correlation of focus scores across dimensions. For timestamps with sudden changes in the time-series, focus scores are higher. Further, the model gives higher attention weights to the specific dimensions of the time-series where the deviations are higher. This allows the model to specifically detect anomalies in each dimension individually, with the contextual trend of the complete sequence as a prior.

4 Experiments

We compare TranAD with state-of-the-art models for mutlivariate time-series anomaly detection, including MERLIN [37], LSTM-NDT [20] (with autoencoder implementation from openGauss [30]), DAGMM [65], OmniAnomaly [45], MSCRED [60], MAD-GAN [29], USAD [4], MTAD-GAT [62], CAE-M [61] and GDN [14] (with graph embedding implementation from GraphAn [9]) . For more details refer Section 2.[1] We also tested the Isolation Forest method, but due to its low F1 scores, do not include the corresponding results in our discussion. Other classical methods have been omitted as deep-learning based approaches have already been shown to outperform them in prior work [4, 14, 62]. We use hyperparameters of the baseline models as presented in their respective papers. We train all models using PyTorch-1.7.1 [40] library[2].

We use the AdamW [27] optimizer to train our model with an initial learning rate of 0.01 (meta learning rate 0.02) and step-scheduler with step size of 0.5 [42]. We use the following hyperparameter values.

- Window size = 10.
- Number of layers in transformer encoders = 1
- Number of layers in feed-forward unit of encoders = 2
- Hidden units in encoder layers = 64
- Dropout in encoders = 0.1

The effect of window size on anomaly detection performance is analyzed in Section 5. We choose hyperparameters other than the window size using grid search. For POT parameters, coefficient = 10^{-4} for all data sets, low quantile is 0.07 for SMAP, 0.01 for MSL, and 0.001 for others. These were selected as per the implementation of the OmniAnomaly baseline [45]. The only dataset-specific hyperparameter is the number of heads in multi-head attention, which was kept to be the same as the dimension size of the dataset. Other assignments for this hyperparameter give similar broad-level trends.

To train TranAD, we divide the training time-series into 80% training data and 20% validation data. To avoid model over-fitting, we use early-stopping criteria to train TranAD, *i.e.*, we stop the training process once the validation accuracy starts to decrease.

4.1 Datasets

We use seven publicly available datasets in our experiments. We summarize their characteristics in Table 1. The values in parenthesis are the number of sequences in the dataset repository and we report

---

Table 1: Dataset Statistics

Dataset	Train	Test	Dimensions	Anomalies (%)
NAB	4033	4033	1 (6)	0.92
UCR	1600	5900	1 (4)	1.88
MBA	100000	100000	2 (8)	0.14
SMAP	135183	427617	25 (55)	13.13
MSL	58317	73729	55 (3)	10.72
SWaT	496800	449919	51 (1)	11.98
WADI	1048571	172801	123 (1)	5.99
SMD	708405	708420	38 (4)	4.16
MSDS	146430	146430	10 (1)	5.37

average scores across all sequences in a dataset. For instance, the SMAP dataset has 55 traces with 25 dimensions each.

(1) *Numenta Anomaly Benchmark (NAB)*: is a dataset of multiple real-world data traces, including readings from temperature sensors, CPU utilization of cloud machines, service request latencies and taxi demands in New York city [2]. However, this dataset is known to have sequences with incorrect anomaly labels [55] such as the nyc-taxi trace [37], which we exclude in our experiments.

(2) *HexagonML (UCR) dataset*: is a dataset of multiple univariate time series (included just for completeness) that was used in KDD 2021 cup [13, 26]. We include only the datasets obtained from natural sources (the InternalBleeding and ECG datasets) and ignore the synthetic sequences.

(3) *MIT-BIH Supraventricular Arrhythmia Database (MBA)*: is a collection of electrocardiogram recordings from four patients, containing multiple instances of two different kinds of anomalies (either supraventricular contractions or premature heartbeats) [17, 36]. This is a popular large-scale dataset in the data management community [8, 10].

(4) *Soil Moisture Active Passive (SMAP) dataset*: is a dataset of soil samples and telemetry information using the Mars rover by NASA [20].

(5) *Mars Science Laboratory (MSL) dataset*: is a dataset similar to SMAP but corresponds to the sensor and actuator data for the Mars rover itself [20]. However, this dataset is known to have many trivial sequences [55]; hence, we consider only the three non-trivial ones (A4, C2 and T1) pointed out by [37].

(6) *Secure Water Treatment (SWaT) dataset*: This dataset is collected from a real-world water treatment plant with 7 days of normal and 4 days of abnormal operation [33]. This dataset consists of sensor values (water level, flow rate, etc.) and actuator operations (valves and pumps).

(7) *Water Distribution (WADI) dataset*: This is an extension of the SWaT system but had more than twice the number of sensors and actuators than the SWaT model [3]. The dataset is also collected for a longer duration of 14 and 2 days of normal and attack scenarios.

(8) *Server Machine Dataset (SMD)*: This is a five-week long dataset of resource utilizations of 28 machines from a compute cluster [45]. Similar to MSL, we use the non-trivial sequences in this dataset, specifically the traces named machine-1-1, 2-1, 3-2 and 3-7.

(9) *Multi-Source Distributed System (MSDS) Dataset*: This is a recent high-quality multi-source data composed of distributed traces, application logs, and metrics from a complex distributed system [38]. This dataset is specifically built for AI operations, including automated anomaly detection, root cause analysis, and remediation.

We eschew comparisons on the Yahoo [53] dataset that has been claimed to suffer from mislabeling and run-to-failure bias [55].

4.2 Evaluation Metrics

4.2.1 Anomaly Detection We use precision, recall, area under the receiver operating characteristic curve (ROC/AUC) and F1 score to evaluate the detection performance of all models [14, 62]. We also measure the AUC and F1 scores by training all models with 20% of the training data (again using the 80:20 split for validation dataset and the rest as the test set), and call these AUC* and F1* respectively, to measure the performance of the models with limited data. We train on the five sets of 20% training data and report average results for statistical significance.

4.2.2 Anomaly Diagnosis We use commonly used metrics to measure the diagnosis performance of all models [62]. HitRate@P% is the measure of how many ground truth dimensions have been included in the top candidates predicted by the model [45]. $P\%$ is the percentage of the ground truth dimensions for each timestamp, which we use to consider the top predicted candidates. For instance, if at timestamp t, if 2 dimensions are labeled anomalous in the ground truth, HitRate@100% would consider top 2 dimensions and HitRate@150% would consider 3 dimensions (100 and 150 are chosen based on prior work [62]). We also measure the Normalized Discounted Cumulative Gain (NDCG) [24]. NDCG@P% considers the same number of top predicted candidates as HitRate@P%.

4.3 Results

Anomaly Detection. Tables 2 and 3 provide the precision, recall, AUC, F1, AUC* and F1* scores for TranAD and baseline models for all datasets. On average, the F1 score of the TranAD model is 0.8802 and F1* is 0.8012. TranAD outperforms the baselines (in terms of F1 score) for all datasets except MSL when we consider the complete dataset for model training. TranAD also outperforms baselines for all datasets except the WADI dataset with 20% of the dataset used for training (F1* score). For MSL, the GDN model has the highest F1 score (0.9591) and for the WADI dataset, OmniAnomaly has the highest F1* score (0.1017). Similarly, TranAD outperforms baselines in terms of AUC scores for all datasets except MSDS, where GDN has the highest AUC (0.9105). All models perform relatively poorly on WADI due to its large-scale in terms of sequence lengths and data modality. Specifically, TranAD achieves improvement of up to 17.06% in F1 score, 14.64% in F1* score, 11.69% in AUC and 11.06% in AUC* scores over the state-of-the-art baseline models.

The MERLIN baseline is a parameter free approach that does not require any training data; hence, we report F1* and AUC* as F1 and AUC scores, respectively. MERLIN performs relatively well only on the univariate datasets, *i.e.* NAB and UCR, and is unable to scale effectively to multivariate data in our traces. The baseline method LSTM-NDT has a good performance on MSL and SMD, but
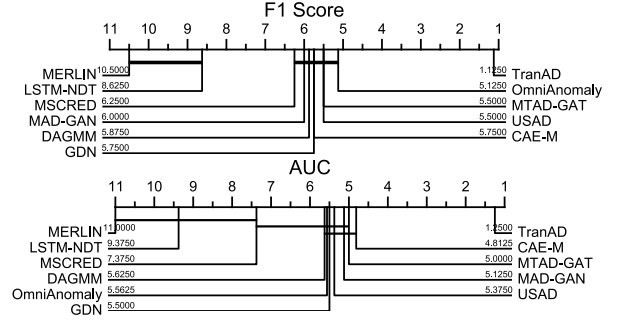
Figure 4: Critical difference diagrams for F1 and AUC scores using the Wilkoxon pairwised signed rank test (with α = 0.05) on all datasets. Rightmost methods are ranked higher.

performs poorly on other datasets. This is due to its sensitivity to different scenarios and poor efficiency of the NDT thresholding method [62]. The POT technique used in TranAD and other models like OmniAnomaly helps set more accurate threshold values by also considering the localized peak values in the data sequence. DAGMM model performs very well for short datasets like UCR, NAB, MBA and SMAP, but its scores drop significantly for other datasets with longer sequences. This is because it does not map the temporal information explicitly as it does not use sequence windows but only a single GRU model. The window encoder in TranAD, with the encoding of the complete sequence as a self-condition, allows it to perform better even with long high-dimensional sequences. The OmniAnomaly, CAE-M and MSCRED models use sequential observations as input, allowing these methods to retain the temporal information. Such methods perform reconstruction regardless of anomalous data, which prevents them from detecting anomalies close to the normal trends [4]. TranAD tackles this by using adversarial training to amplify errors. Hence, in datasets like SMD, where anomalous data is not very far from normal data, it can detect even mild anomalies.

Recent models such as USAD, MTAD-GAT and GDN use attention mechanisms to focus on specific modes of the data. Moreover, these models try to capture the long-term trends by adjusting the weights of their neural network and only use a local window as an input for reconstruction. GDN has slightly higher scores for MSL and MSDS datasets than TranAD due to the scalable graph-based inference over the inter-dimensional data correlations [14]. TranAD does this using self-attention and performs better than GDN overall across all datasets. The limitation of seeing only a local contextual window prevents methods such as USAD and MTAD-GAT from classifying long-term anomalies (like in SMD or WADI). However, self-conditioning on an embedding of the complete trace with position encoding aids temporal attention, thanks to the transformer architecture in TranAD. This allows TranAD to capture long-term trends more effectively. Further, due to the meta-learning, TranAD also outperforms baselines with limited training data except for OmniAnomaly on the WADI dataset, indicating its high efficacy even with limited data. OmniAnomaly performs best among all methods on the WADI dataset due to high noise in this dataset and dedicated stochasticity modeling in OmniAnomaly [45]. TranAD is slightly behind this method in terms of F1* and AUC*; however,

Table 2: Performance comparison of TranAD with baseline methods on the complete dataset. P: Precision, R: Recall, AUC: Area under the ROC curve, F1: F1 score with complete training data. The best F1 and AUC scores are highlighted in bold.

Method	NAB				UCR				MBA			
	P	R	AUC	F1	P	R	AUC	F1	P	R	AUC	F1
MERLIN	0.8013	0.7262	0.8414	0.7619	0.7542	0.8018	0.8984	0.7542	0.9846	0.4913	0.7828	0.6555
LSTM-NDT	0.6400	0.6667	0.8322	0.6531	0.5231	0.8294	0.9781	0.5231	0.9207	0.9718	0.9780	0.9456
DAGMM	0.7622	0.7292	0.8572	0.7453	0.5337	0.9718	0.9916	0.5337	0.9475	0.9900	0.9858	0.9683
OmniAnomaly	0.8421	0.6667	0.8330	0.7442	0.8346	0.9999	0.9981	0.8346	0.8561	1.0000	0.9570	0.9225
MSCRED	0.8522	0.6700	0.8401	0.7502	0.5441	0.9718	0.9920	0.5441	0.9272	1.0000	0.9799	0.9623
MAD-GAN	0.8666	0.7012	0.8478	0.7752	0.8538	0.9891	0.9984	0.8538	0.9396	1.0000	0.9836	0.9689
USAD	0.8421	0.6667	0.8330	0.7442	0.8952	1.0000	0.9989	0.8952	0.8953	0.9989	0.9701	0.9443
MTAD-GAT	0.8421	0.7272	0.8221	0.7804	0.7812	0.9972	0.9978	0.7812	0.9018	1.0000	0.9721	0.9484
CAE-M	0.7918	0.8019	0.8019	0.7968	0.6981	1.0000	0.9957	0.6981	0.8442	0.9997	0.9661	0.9154
GDN	0.8129	0.7872	0.8542	0.7998	0.6894	0.9988	0.9959	0.6894	0.8832	0.9892	0.9528	0.9332
TranAD	0.8889	0.9892	**0.9541**	**0.9364**	0.9407	1.0000	**0.9994**	**0.9407**	0.9569	1.0000	**0.9885**	**0.9780**

Method	SMAP				MSL				SWaT			
	P	R	AUC	F1	P	R	AUC	F1	P	R	AUC	F1
MERLIN	0.1577	0.9999	0.7426	0.2725	0.2613	0.4645	0.6281	0.3345	0.6560	0.2547	0.6175	0.3669
LSTM-NDT	0.8523	0.7326	0.8602	0.7879	0.6288	1.0000	0.9532	0.7721	0.7778	0.5109	0.7140	0.6167
DAGMM	0.8069	0.9891	0.9885	0.8888	0.7363	1.0000	0.9716	0.8482	0.9933	0.6879	0.8436	0.8128
OmniAnomaly	0.8130	0.9419	0.9889	0.8728	0.7848	0.9924	0.9782	0.8765	0.9782	0.6957	0.8467	0.8131
MSCRED	0.8175	0.9216	0.9821	0.8664	0.8912	0.9862	0.9807	0.9363	0.9992	0.6770	0.8433	0.8072
MAD-GAN	0.8157	0.9216	0.9891	0.8654	0.8516	0.9930	0.9862	0.9169	0.9593	0.6957	0.8463	0.8065
USAD	0.7480	0.9627	0.9890	0.8419	0.7949	0.9912	0.9795	0.8822	0.9977	0.6879	0.8460	0.8143
MTAD-GAT	0.7991	0.9991	0.9844	0.8880	0.7917	0.9824	0.9899	0.8768	0.9718	0.6957	0.8464	0.8109
CAE-M	0.8193	0.9567	0.9901	0.8827	0.7751	1.0000	0.9903	0.8733	0.9697	0.6957	0.8464	0.8101
GDN	0.7480	0.9891	0.9864	0.8518	0.9308	0.9892	0.9814	**0.9591**	0.9697	0.6957	0.8462	0.8101
TranAD	0.8043	0.9999	**0.9921**	0.8915	0.9038	0.9999	**0.9916**	0.9494	0.9760	0.6997	**0.8491**	**0.8151**

Method	WADI				SMD				MSDS			
	P	R	AUC	F1	P	R	AUC	F1	P	R	AUC	F1
MERLIN	0.0636	0.7669	0.5912	0.1174	0.2871	0.5804	0.7158	0.3842	0.7254	0.3110	0.5022	0.4353
LSTM-NDT	0.0138	0.7823	0.6721	0.0271	0.9736	0.8440	0.9671	0.9042	0.9999	0.8012	0.8013	0.8896
DAGMM	0.0760	0.9981	0.8563	0.1412	0.9103	0.9914	0.9954	0.9491	0.9891	0.8026	0.9013	0.8861
OmniAnomaly	0.3158	0.6541	0.8198	0.4260	0.8881	0.9985	0.9946	0.9401	1.0000	0.7964	0.8982	0.8867
MSCRED	0.2513	0.7319	0.8412	0.3741	0.7276	0.9974	0.9921	0.8414	1.0000	0.7983	0.8943	0.8878
MAD-GAN	0.2233	0.9124	0.8026	0.3588	0.9991	0.8440	0.9933	0.9150	0.9982	0.6107	0.8054	0.7578
USAD	0.1873	0.8296	0.8723	0.3056	0.9060	0.9974	0.9933	0.9495	0.9912	0.7959	0.8979	0.8829
MTAD-GAT	0.2818	0.8012	0.8821	0.4169	0.8210	0.9215	0.9921	0.8683	0.9919	0.7964	0.8982	0.8835
CAE-M	0.2782	0.7918	0.8728	0.4117	0.9082	0.9671	0.9783	0.9367	0.9908	0.8439	0.9013	0.9115
GDN	0.2912	0.7931	0.8777	0.4260	0.7170	0.9974	0.9924	0.8342	0.9989	0.8026	**0.9105**	0.8900
TranAD	0.3529	0.8296	**0.8968**	**0.4951**	0.9262	0.9974	**0.9974**	**0.9605**	0.9999	0.8626	0.9013	**0.9262**

outperforms it when compared across all datasets and also when given the complete WADI dataset.

We perform critical difference analysis to assess the significance of the differences among the performance of the models. Figure 4 depicts the critical difference diagrams for the F1 and AUC scores based on the Wilcoxon pair-wised signed-rank test (with $\alpha = 0.05$) after rejecting the null hypothesis using the Friedman test on all datasets [22]. TranAD achieves the best rank across all models with a significant statistical difference.

Anomaly Diagnosis. The anomaly diagnosis results in Table 4 where H and N correspond to HitRate and NDCG (with complete data used for model testing). We only present results on the multivariate SMD and MSDS datasets for the sake of brevity (TranAD yields better scores for others as well). We also ignore models that do not explicitly output anomaly class outputs for each dimension individually. Multi-head attention in TranAD allows it to attend to multiple modes simultaneously, making it more suitable for more inter-correlated anomalies. This is observed and explained by datasets like MSDS (distributed systems) where anomalous behavior in one mode can lead to a chain of events causing anomalies in other modes (see Figure 5). TranAD is able to leverage the complete trace information with the local window to aid in pinpointing

Table 3: Performance comparison of TranAD with baseline methods with 20% of the training dataset. AUC*: AUC with 20% training data, F1*: F1 score with 20% training data. The best F1* and AUC* scores are highlighted in bold.

Method	NAB		UCR		MBA	
	AUC*	F1*	AUC*	F1*	AUC*	F1*
MERLIN	0.8029	0.7619	0.8984	0.7773	0.7828	0.6555
LSTM-NDT	0.8013	0.6212	0.8913	0.5198	0.9617	0.9282
DAGMM	0.7827	0.6125	0.9812	0.5718	0.9671	0.9396
OmniAnomaly	0.8129	0.6713	0.9728	0.7918	0.9407	0.9217
MSCRED	0.8299	0.7013	0.9637	0.4929	0.9499	0.9108
MAD-GAN	0.8194	0.7109	0.9959	0.8216	0.9550	0.9192
USAD	0.7267	0.6781	0.9967	0.8538	0.9697	0.9425
MTAD-GAT	0.6956	0.7013	0.9974	0.8671	0.9688	0.9425
CAE-M	0.7312	0.7126	0.9926	0.7525	0.9616	0.9002
GDN	0.8299	0.7013	0.9937	0.8029	0.9671	0.9316
TranAD	**0.9217**	**0.8421**	**0.9989**	**0.9399**	**0.9718**	**0.9617**

Method	SMAP		MSL		SWaT	
	AUC*	F1*	AUC*	F1*	AUC*	F1*
MERLIN	0.7426	0.2725	0.6281	0.3345	0.6175	0.3669
LSTM-NDT	0.7007	0.5418	0.9520	0.7608	0.6690	0.4145
DAGMM	0.9881	0.8369	0.9606	0.8010	0.8421	0.8001
OmniAnomaly	0.9879	0.8131	0.9703	0.8424	0.8319	0.7433
MSCRED	0.9811	0.8050	0.9797	0.8232	0.8385	0.7922
MAD-GAN	0.9877	0.8468	0.9649	0.8190	0.8456	0.8012
USAD	0.9883	0.8379	0.9649	0.8190	0.8438	0.8087
MTAD-GAT	0.9814	0.8225	0.9782	0.8024	**0.8459**	0.8079
CAE-M	**0.9892**	0.8312	0.9836	0.7303	0.8458	0.7841
GDN	0.9887	0.8411	0.9414	0.8959	0.8390	0.8072
TranAD	0.9885	**0.8889**	**0.9857**	**0.9172**	0.8438	**0.8094**

| Method	WADI		SMD		MSDS	
	AUC*	F1*	AUC*	F1*	AUC*	F1*
MERLIN	0.5912	0.1174	0.7158	0.3842	0.5022	0.4353
LSTM-NDT	0.6637	0.0000	0.9563	0.6754	0.7813	0.7912
DAGMM	0.6497	0.0630	0.9845	0.8986	0.7763	0.8389
OmniAnomaly	**0.7913**	**0.1017**	0.9859	0.9352	0.5613	0.8389
MSCRED	0.6029	0.0413	0.9768	0.8004	0.7716	0.8283
MAD-GAN	0.5383	0.0937	0.8635	0.9318	0.5002	0.7390
USAD	0.7011	0.0733	0.9854	0.9213	0.7613	0.8389
MTAD-GAT	0.6267	0.0520	0.9798	0.6661	0.6122	0.8248
CAE-M	0.6109	0.0781	0.9569	0.9318	0.6001	0.8389
GDN	0.6121	0.0412	0.9811	0.7107	0.6819	0.8389
TranAD	0.7688	0.0649	**0.9869**	**0.9478**	**0.8113**	**0.8391**

anomalous behavior to specific modes. The table demonstrates that TranAD is able to detect $46.3\% - 75.3\%$ root causes for anomalies.

Compared to the baseline methods, TranAD is able to improve diagnosis score by up to 6% for SMD and 30% for MSDS. The average improvement in diagnosis scores is 4.25%.

5 Analyses

5.1 Ablation Analysis

To study the relative importance of each component of the model, we exclude every major one and observe how it affects the performance in terms of the F1 scores for each dataset. First, we consider the TranAD model without the transformer-based encoder-decoder architecture but instead with a feed-forward network. Second, we consider the model without the self-conditioning, *i.e.*, we fix the
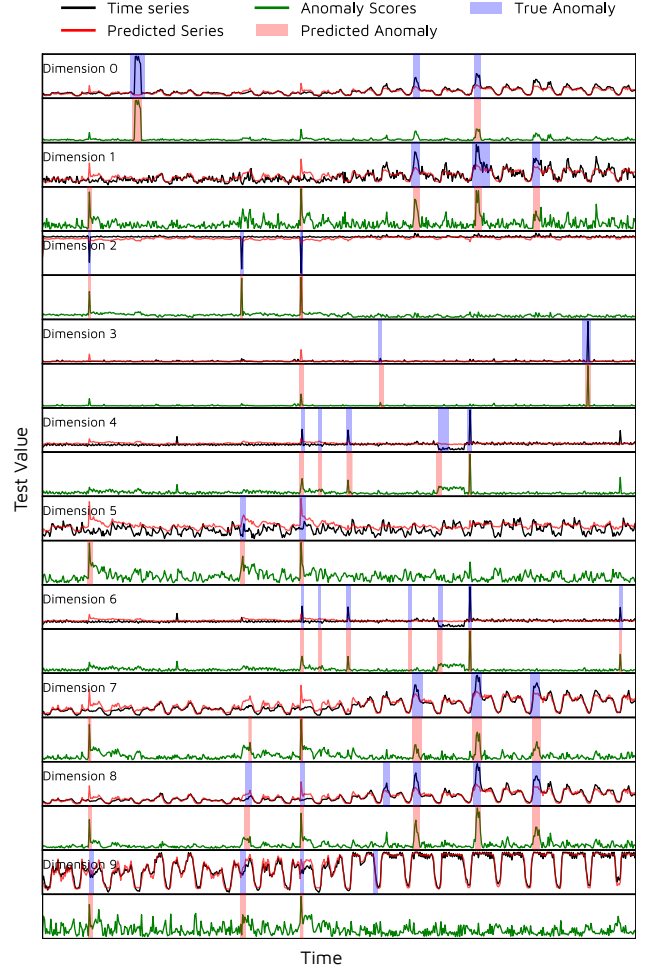
Figure 5: Predicted and Ground Truth labels for the MSDS test set using the TranAD model.

focus score $F = \vec{0}$ in phase 2. Third, we study the model without the adversarial loss, *i.e.*, a single-phase inference and only the reconstruction loss for model training. Finally, we consider the model without meta-learning. The results are summarized in Table 6 and provide the following findings:

- Replacing the transformer-based encoder-decoder has the highest performance drop of nearly 11% in terms of the F1 score. This drop is more pronounced for the WADI dataset (56%), demonstrating the need for the attention-based transformer for large-scale datasets.
- When we remove the self-conditioning, the average drop in F1 scores is 6%, which shows that the focus score aids prediction performance.
- Removing the two-phase adversarial training mainly affects SMD and WADI datasets as these traces have a large proportion of mild anomalies and the adversarial loss helps amplify the anomaly scores. The average drop in F1 score, in this case, is 5%.
- Not having the meta-learning in the model has little effect to the F1 scores ($\approx 1\%$); however, it leads to a nearly 12% drop in F1* scores.

Table 4: Diagnosis Performance.

Method	SMD				MSDS			
	H@100%	H@150%	N@100%	N@150%	H@100%	H@150%	N@100%	N@150%
MERLIN	0.5907	0.6177	0.4150	0.4912	0.3816	0.5626	0.3010	0.3947
LSTM-NDT	0.3808	0.5225	0.3603	0.4451	0.1504	0.2959	0.1124	0.1993
DAGMM	0.4927	0.6091	0.5169	0.5845	0.2617	0.4333	0.3153	0.4154
OmniAnomaly	0.4567	0.5652	0.4545	0.5125	0.2839	0.4365	0.3338	0.4231
MSCRED	0.4272	0.5180	0.4609	0.5164	0.2322	0.3469	0.2297	0.2962
MAD-GAN	0.4630	0.5785	0.4681	0.5522	0.3856	0.5589	0.4277	0.5292
USAD	0.4925	0.6055	0.5179	0.5781	0.3095	0.4769	0.3534	0.4515
MTAD-GAT	0.3493	0.4777	0.3759	0.4530	**0.5812**	0.5885	0.5926	0.6522
CAE-M	0.4707	0.5878	**0.5474**	**0.6178**	0.2530	0.4171	0.2047	0.3010
GDN	0.3143	0.4386	0.2980	0.3724	0.2276	0.3382	0.2921	0.3570
TranAD	**0.4981**	**0.6401**	0.4941	**0.6178**	0.4630	**0.7533**	**0.5981**	**0.6963**

Table 5: Comparison of training times in seconds per epoch.

Method	NAB	UCR	MBA	SMAP	MSL	SWaT	WADI	SMD	MSDS
MERLIN	3.28	4.09	20.19	6.89	5.12	10.12	132.69	72.32	42.22
LSTM-NDT	10.64	8.71	27.80	27.62	26.24	26.43	297.12	373.14	361.12
DAGMM	25.38	20.78	74.62	19.05	16.41	18.51	178.17	204.36	187.54
OmniAnomaly	38.27	27.96	109.86	27.05	21.31	28.39	212.99	276.97	277.10
MSCRED	258.86	262.45	592.13	16.13	33.47	183.67	1349.05	237.66	109.63
MAD-GAN	39.80	25.71	160.29	29.49	26.27	27.79	293.60	314.82	285.25
USAD	31.21	21.10	120.86	23.63	21.22	22.72	242.86	250.97	232.82
MTAD-GAT	145.00	97.12	233.08	1015.03	1287.42	103.92	9812.13	6564.11	1304.09
CAE-M	22.48	19.42	67.44	187.35	575.96	41.25	5525.62	3102.12	552.83
GDN	83.84	58.78	159.01	62.33	96.71	59.40	4063.05	809.94	585.34
TranAD	**1.25**	**0.84**	**4.08**	**3.55**	**5.27**	**0.87**	**115.91**	**43.56**	**17.15**
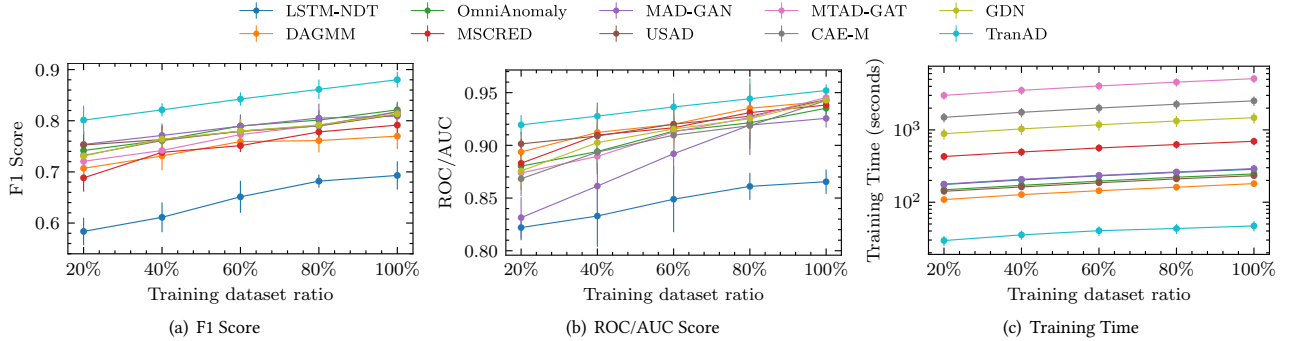
Figure 6: F1 score, ROC/AUC score and training times with dataset size.

5.2 Overhead Analysis

Table 5 shows the average training times for all models on every dataset in seconds per epoch. For comparison, we report the training time for MERLIN as the time it takes to discover discords in the test data. The training time of TranAD is $75\% - 99\%$ lower than those of the baseline methods. This clearly shows the advantage of having a transformer model with positional encoding to push the complete sequence as an input instead of sequentially inferring over local windows.

5.3 Sensitivity Analysis

Sensitivity to the training set size. Figure 6 shows the variation of the F1 and AUC scores of all models averaged for all datasets, and the training time with the ratio of the training data used for model training, ranging from 20% to 100%. We do not report sensitivity results on MERLIN as it does not use training data. Other deep learning based reconstruction models are given the same randomly sampled subsequence of 20% to 100% size as that of the training data. We report 90% confidence bounds in Figure 6. Clearly, as dataset size increases, the prediction performance improves and the training
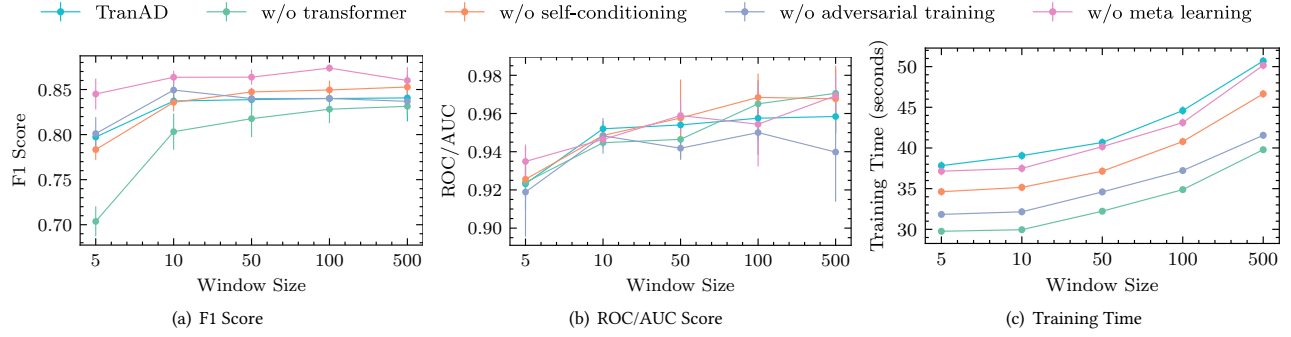
Figure 7: F1 score, ROC/AUC score and training times with window size.

Table 6: Ablation Study - F1 and F1* scores for TranAD and its ablated versions.

Method	NAB		UCR		MBA	
	F1	F1*	F1	F1*	F1	F1*
TranAD	**0.9364**	**0.8421**	**0.9694**	**0.9399**	**0.9780**	**0.9617**
w/o transformer	0.8850	0.8019	0.8466	0.5495	0.9749	0.9584
w/o self-condition	0.8887	0.8102	0.9191	0.9028	0.9770	**0.9617**
w/o adversarial training	0.9012	0.8102	0.9634	0.9289	0.9752	0.9592
w/o MAML	0.9068	0.8210	0.9689	0.9304	0.9756	**0.9617**

Method	SMAP		MSL		SWaT	
	F1	F1*	F1	F1*	F1	F1*
TranAD	**0.8915**	**0.8889**	**0.9494**	**0.9172**	**0.8151**	**0.8094**
w/o transformer	0.8643	0.8147	0.9137	0.9037	0.8143	0.6360
w/o self-condition	0.8894	0.8153	0.9175	0.8913	0.7953	0.8094
w/o adversarial training	0.8906	0.8476	0.9455	0.9172	0.8028	0.7832
w/o MAML	**0.8915**	0.8899	0.9466	0.6732	0.8143	0.8079

Method	WADI		SMD		MSDS	
	F1	F1*	F1	F1*	F1	F1*
TranAD	**0.4951**	**0.0649**	**0.9605**	**0.9478**	**0.9262**	**0.8391**
w/o transformer	0.2181	0.0037	0.9071	0.9032	0.8867	0.8389
w/o self-condition	0.3620	0.0631	0.9502	0.8847	0.8748	0.8214
w/o adversarial training	0.3820	0.0621	0.9177	0.8667	0.9181	0.8389
w/o MAML	0.4815	0.0553	0.9433	0.8164	0.8870	0.8389

time increases. We observe that for every ratio, the TranAD model has a higher F1 score and a lower training time.

Sensitivity to the window size. We also show the performance of the TranAD model and its variants with different window sizes in Figure 7. The window size has an impact both on the anomaly detection scores and training times. TranAD can detect anomalies faster when we have smaller windows since the inference time is lower for smaller inputs. If the window is too small, it does not represent the local contextual information well. However, if the window is too large, short anomalies may be hidden in a large number of datapoints in such a window (see the drop in F1 score for some models). A window size of 10 gives a reasonable balance between the F1 score and training times and hence is used in our experiments.

6 Conclusions

We present a transformer based anomaly detection model (TranAD) that can detect and diagnose anomalies for multivariate time-series data. The transformer based encoder-decoder allows quick model

training and high detection performance for a variety of datasets considered in this work. TranAD leverages self-conditioning and adversarial training to amplify errors and gain training stability. Moreover, meta-learning allows it to be able to identify data trends even with limited data. Specifically, TranAD achieves an improvement of 17% and 11% for F1 score on complete and limited training data, respectively. It is also able to correctly identify root causes for up to 75% of the detected anomalies, higher than the state-of-the-art models. It is able to achieve this with up to 99% lower training times compared to the baseline methods. This makes TranAD an ideal choice for modern industrial systems where accurate and quick anomaly predictions are required.

For the future, we propose to extend the method with other transformer models like bidirectional neural networks to allow model generalization to diverse temporal trends in data. We also wish to explore the direction of applying cost-benefit analysis for each model component based on the deployment setting to avoid expensive computation.

Software Availability

The code and relevant training scripts are made publicly available on GitHub under BSD-3 licence at `https://github.com/imperial-qore/TranAD`.

References

[1] Hossein Abbasimehr, Mostafa Shabani, and Mohsen Yousefi. 2020. An optimized model using LSTM network for demand forecasting. *Computers & industrial engineering* 143 (2020), 106435.
[2] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. 2017. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* 262 (2017), 134–147.
[3] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. 2017. WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*. 25–28.
[4] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. 2020. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3395–3404.

[5] Tharindu R Bandaragoda, Kai Ming Ting, David Albrecht, Fei Tony Liu, and Jonathan R Wells. 2014. Efficient anomaly detection by isolation using nearest neighbour ensemble. In *2014 IEEE International Conference on Data Mining Workshop*. IEEE, 698–705.

[6] Julian Bellendorf and Zoltán Ádám Mann. 2020. Classification of optimization problems in fog computing. *Future Generation Computer Systems* 107 (2020), 158–176.

[7] Nejc Bezak, Mitja Brilly, and Mojca Šraj. 2014. Comparison between the peaks-over-threshold method and the annual maximum method for flood frequency analysis. *Hydrological Sciences Journal* 59, 5 (2014), 959–977.

[8] Paul Boniol, Michele Linardi, Federico Roncallo, and Themis Palpanas. 2020. Automated anomaly detection in large sequences. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1834–1837.

[9] Paul Boniol, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. 2020. Graphan: Graph-based subsequence anomaly detection. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2941–2944.

[10] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J. Franklin. 2021. SAND: Streaming Subsequence Anomaly Detection. *Proc. VLDB Endow.* 14, 10 (2021), 1717–1729.

[11] Saikiran Bulusu, Bhavya Kailkhura, Bo Li, Pramod K Varshney, and Dawn Song. 2020. Anomalous example detection in deep learning: A survey. *IEEE Access* 8 (2020), 132330–132347.

[12] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).

[13] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. 2018. The UCR Time Series Classification Archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[14] Ailin Deng and Bryan Hooi. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4027–4035.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. PMLR, 1126–1135.

[16] Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall, Amirali Darvishzadeh, and Eamonn Keogh. 2018. Matrix profile XII: MPDIST: a novel time series distance measure to allow data mining in more challenging scenarios. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 965–970.

[17] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation* 101, 23 (2000), e215–e220.

[18] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A Survey of the State-of-the-Art. *Knowledge-Based Systems* 212 (2021), 106622.

[19] Shaohan Huang, Yi Liu, Carol Fung, Rong He, Yining Zhao, Hailong Yang, and Zhongzhi Luan. 2020. HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log. *IEEE Transactions on Network and Service Management* 17, 4 (2020), 2064–2076.

[20] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 387–395.

[21] Shima Imani, Frank Madrid, Wei Ding, Scott Crouter, and Eamonn Keogh. 2018. Matrix profile xiii: Time series snippets: a new primitive for time series data mining. In *2018 IEEE international conference on big knowledge (ICBK)*. IEEE, 382–389.

[22] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* 33, 4 (2019), 917–963.

[23] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul. 2020. Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series. *Proceedings of the VLDB Endowment* (2020).

[24] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[25] Stratis Kanarachos, Jino Mathew, Alexander Chroneos, and M Fitzpatrick. 2015. Anomaly detection in time series data using a combination of wavelets, neural networks and Hilbert transform. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*. IEEE, 1–6.

[26] Eamonn Keogh, Dutta Roy Taposh, U Naik, and A Agrawal. 2021. Multi-dataset Time-Series Anomaly Detection Competition. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://compete.hexagon-ml.com/practice/competition/39/.

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Kyle Kingsbury and Peter Alvaro. 2020. Elle: inferring isolation anomalies from experimental observations. *Proceedings of the VLDB Endowment* 14, 3 (2020), 268–280.

[29] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. 2019. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*. Springer, 703–716.

[30] Guoliang Li, Xuanhe Zhou, Ji Sun, Xiang Yu, Yue Han, Lianyuan Jin, Wenbo Li, Tianqing Wang, and Shifu Li. 2021. opengauss: An autonomous database system. *Proceedings of the VLDB Endowment* 14, 12 (2021), 3028–3042.

[31] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[32] Steven Liu, Tongzhou Wang, David Bau, Jun-Yan Zhu, and Antonio Torralba. 2020. Diverse image generation via self-conditioned GANs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14286–14295.

[33] Aditya P Mathur and Nils Ole Tippenhauer. 2016. SWaT: a water treatment testbed for research and training on ICS security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*. IEEE, 31–36.

[34] Gideon Mbiydzenyuy. 2020. Univariate Time Series Anomaly Labelling Algorithm. In *International Conference on Machine Learning, Optimization, and Data Science*. Springer, 586–599.

[35] Steena Monteiro, Forrest Iandola, and Daniel Wong. 2016. STOMP: Statistical Techniques for Optimizing and Modeling Performance of blocked sparse matrix vector multiplication. In *2016 28th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 93–100.

[36] George B Moody and Roger G Mark. 2001. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine* 20, 3 (2001), 45–50.

[37] Takaaki Nakamura, Makoto Imamura, Ryan Mercer, and Eamonn Keogh. 2020. MERLIN: Parameter-Free Discovery of Arbitrary Length Anomalies in Massive Time Series Archives. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1190–1195.

[38] Sasho Nedelkoski, Jasmin Bogatinovski, Ajay Kumar Mandapati, Soeren Becker, Jorge Cardoso, and Odej Kao. 2020. Multi-source Distributed System Data for AI-Powered Analytics. In *European Conference on Service-Oriented and Cloud Computing*. Springer, 161–176.

[39] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. 2018. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1544–1551.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems* 32 (2019), 8026–8037.

[41] Animesh Patcha and Jung-Min Park. 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks* 51, 12 (2007), 3448–3470.

[42] Noorhan Saleh and Maggie Mashaly. 2019. A Dynamic Simulation Environment for Container-based Cloud Data Centers using ContainerCloudSim. In *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*. IEEE, 332–336.

[43] Osman Salem, Alexey Guerassimov, Ahmed Mehaoua, Anthony Marcus, and Borko Furht. 2014. Anomaly detection in medical wireless sensor networks using SVM and linear regression models. *International Journal of E-Health and Medical Communications (IJEHMC)* 5, 1 (2014), 20–45.

[44] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. 2017. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1067–1075.

[45] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2828–2837.

[46] Srikanth Thudumu, Philip Branch, Jiong Jin, and Jugdutt Jack Singh. 2020. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data* 7, 1 (2020), 1–30.

[47] Luan Tran, Min Y Mun, and Cyrus Shahabi. 2020. Real-time distance-based outlier detection in data streams. *Proceedings of the VLDB Endowment* 14, 2 (2020), 141–153.

[48] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. 2022. PreGAN: Preemptive Migration Prediction Network for Proactive Fault-Tolerant Edge Computing. In *IEEE Conference on Computer Communications (INFOCOM)*. IEEE.

[49] Shreshth Tuli, Shivananda Poojara, Satish Narayana Srirama, Giuliano Casale, and Nick Jennings. 2021. COSCO: Container Orchestration using Co-Simulation and Gradient Based Optimization for Fog Computing Environments. *IEEE Transactions on Parallel and Distributed Systems* (2021).

[50] Shreshth Tuli, Shikhar Tuli, Ruchi Verma, and Rakesh Tuli. 2020. Modelling for prediction of the spread and severity of COVID-19 and its association with socioeconomic factors and virus types. *Biomedical Research and Clinical Reviews* 1 (2020). Issue 3.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.

[52] Yiyang Wang, Neda Masoud, and Anahita Khojandi. 2020. Real-time sensor anomaly detection and recovery in connected automated vehicle sensors. *IEEE Transactions on Intelligent Transportation Systems* 22, 3 (2020), 1411–1421.

[53] Y Webscope. [n.d.]. S5-A Labeled Anomaly Detection Dataset, Version 1.0. `https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70`. Accessed: 2021-08-31.

[54] Krzysztof Witkowski. 2017. Internet of things, big data, industry 4.0–innovative solutions in logistics and supply chains management. *Procedia engineering* 182 (2017), 763–769.

[55] Renjie Wu and Eamonn J Keogh. 2020. Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress. *arXiv preprint arXiv:2009.13807* (2020).

[56] Asrul H Yaacob, Ian KT Tan, Su Fong Chien, and Hon Khi Tan. 2010. Arima based network anomaly detection. In *2010 Second International Conference on Communication Software and Networks*. IEEE, 205–209.

[57] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

[58] Dragomir Yankov, Eamonn Keogh, and Umaa Rebbapragada. 2008. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. *Knowledge and Information Systems* 17, 2 (2008), 241–262.

[59] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th international conference on data mining (ICDM)*. Ieee, 1317–1322.

[60] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. 2019. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1409–1416.

[61] Yuxin Zhang, Yiqiang Chen, Jindong Wang, and Zhiwen Pan. 2021. Unsupervised Deep Anomaly Detection for Multi-Sensor Time-Series Signals. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[62] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2020. Multivariate time-series anomaly detection via graph attention network. *International Conference on Data Mining* (2020).

[63] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh. 2018. Matrix profile XI: SCRIMP++: time series motif discovery at interactive speeds. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 837–846.

[64] Zachary Zimmerman, Nader Shakibay Senobari, Gareth Funning, Evangelos Papalexakis, Samet Oymak, Philip Brisk, and Eamonn Keogh. 2019. Matrix profile XVIII: time series mining in the face of fast moving streams using a learned approximate matrix profile. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 936–945.

[65] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*.