Anomaly Detection in Time Series via Contrastive Representation Learning

Lalit Karthikeyan M A
3rd Year Undergraduate
Indian Institute of Technology Kharagpur
Kharagpur, India
Email: karthikeyanlalit05@kgpian.iitkgp.ac.in

Daksh Jogchand
3rd Year Undergraduate
Indian Institute of Technology Kharagpur
Kharagpur, India
Email: daksh.jogchand@kgpian.iitkgp.ac.in

Pratyush Chatterjee
3rd Year Undergraduate
Indian Institute of Technology Kharagpur
Kharagpur, India
Email: pratyushchatterjee@kgpian.iitkgp.ac.in

Nyasa
4th Year Undergraduate
Indian Institute of Technology Kharagpur
Kharagpur, India
Email: nyasajha@kgpian.iitkgp.ac.in

Abstract—**Time series anomaly detection is a fundamental task with critical applications in domains such as industrial monitoring, healthcare, and finance. Traditional supervised methods rely heavily on annotated data, which is often scarce and costly to obtain for anomalies. Recent advances in self-supervised and unsupervised learning have shown promise in learning rich representations from unlabeled time series data. This report focuses on reproducing and evaluating two notable models: TS2Vec, which learns universal time series representations using hierarchical contrastive learning, and DCdetector, which applies a dual attention-based contrastive framework to distinguish anomalous from normal patterns without reconstruction loss. We compare their architectures, training procedures, and performance across benchmark datasets to understand their effectiveness and practical applicability for unsupervised anomaly detection tasks.**

1. Introduction

Time series data is increasingly generated by modern systems in sectors ranging from manufacturing and finance to healthcare and infrastructure. Detecting anomalies in such data is crucial for early fault diagnosis, fraud detection, and system health monitoring. However, real-world deployment of anomaly detection systems is hindered by the scarcity of labeled anomaly data and the unpredictable nature of novel anomalies. These limitations motivate the need for unsupervised representation learning methods that can identify deviations from normal patterns without requiring labeled supervision.

In this report, we reproduce and compare two recent approaches that leverage contrastive learning to address this challenge: **TS2Vec** and **DCdetector**. TS2Vec introduces a hierarchical contrastive learning framework that captures contextual representations of time series at different temporal resolutions. It learns timestamp-level embeddings that can be aggregated for downstream tasks such as classification, forecasting, and anomaly detection. Its robustness to varying sequence lengths, scales, and missing data makes it a strong candidate for universal representation learning.

In contrast, DCdetector is specifically designed for anomaly detection and adopts a dual-branch contrastive learning framework enhanced with temporal and feature attention mechanisms. By learning to differentiate between consistent (normal) and inconsistent (anomalous) representations across branches, it highlights discrepancies without relying on reconstruction error. The model also incorporates multi-scale patching and a channel-independent attention mechanism to effectively process multivariate time series.

This combined study aims to evaluate the effectiveness of both models in unsupervised anomaly detection scenarios. Through detailed reproduction, analysis, and comparison, we assess their capabilities in terms of architecture, scalability, and performance across multiple real-world time series datasets.

2. Literature Review

Time series anomaly detection has gained significant attention due to its wide applicability in domains like manufacturing, healthcare, and finance [1]. The task involves identifying unusual or abnormal patterns in temporal data that may indicate system faults, fraud, or rare events. Broadly, anomaly detection techniques can be categorized based on the availability of labeled data, leading to supervised and unsupervised learning approaches [2].

2.1. Limitations of Supervised Anomaly Detection

Supervised models generally perform well by learning detailed distinctions between normal and anomalous data. However, in industrial settings where anomalies are rare and labeled data is limited, such models face challenges.

Annotating rare failure types is costly and often infeasible, especially when intentionally inducing faults is impractical [3]. Moreover, many potential anomaly types are unknown beforehand, making supervised classification inadequate.

2.2. Advantages of Unsupervised Approaches

To address these limitations, unsupervised models are trained only on normal data and aim to detect deviations during testing. This paradigm is known as one-class classification or out-of-distribution detection [4]. It avoids reliance on labeled anomalies and is more practical in real-world industrial applications.

2.3. Categories of Unsupervised Methods

Unsupervised anomaly detection methods are typically divided into:

- **Reconstruction-based methods**: These aim to rebuild the input data and flag large reconstruction errors as anomalies. A common solution is to use convolutional autoencoders trained on normal images to reconstruct inputs and compute pixel-wise differences—using metrics like L2-distance or SSIM—to detect anomalies. The underlying assumption is that autoencoders trained solely on normal data fail to accurately reconstruct anomalous regions, leading to noticeable reconstruction errors [5]. Variants using VAEs [6] and GANs [7] follow a similar approach, with some leveraging reconstruction probability or likelihood scores. However, such models often generate blurred reconstructions, especially at edges and textures, resulting in frequent false positives.
- **Representation-based methods**: Representation-based methods aim to learn feature embeddings where anomalies can be clearly distinguished from normal instances [8]. These approaches, including contrastive learning techniques, use deep neural networks to extract compact feature vectors that describe the entire image. The anomaly score is typically computed as the distance between the test image's embedding and a reference embedding derived from normal training data. The core idea is to minimize intra-class variance among normal samples, making anomalies stand out in the feature space [9].

2.4. TS2Vec: Universal Representation of Time Series

TS2Vec [10] proposes a unified framework for learning robust, universal representations of time series at arbitrary semantic levels. Unlike prior methods focused on coarse-grained, instance-level embeddings, TS2Vec leverages a hierarchical contrastive learning strategy to generate contextualized representations at each timestamp. This makes it particularly effective for tasks that require fine-grained understanding, such as anomaly detection and forecasting.

The model performs contrastive learning across multiple temporal resolutions by constructing augmented context views. For any arbitrary sub-sequence, TS2Vec can derive a representation via simple aggregation—typically max-pooling—over the corresponding timestamp-level embeddings. This design enables the framework to adapt to various semantic granularities, capturing both global trends and local variations.

One of TS2Vec's key innovations lies in its hierarchical contrasting mechanism, which operates along both the instance-wise and temporal dimensions. Furthermore, the paper introduces the concept of contextual consistency in positive pair selection, which avoids assumptions like transformation- or cropping-invariance that are ill-suited for time series data. This makes TS2Vec more robust to the challenges posed by diverse distributions, varying scales, and even missing values.

2.4.1. Why did we choose this paper?. The TS2Vec paper was selected due to its innovative approach to learning universal, fine-grained representations of time series data without the need for labeled samples. Its hierarchical contrastive learning framework effectively addresses limitations in traditional instance-level models, making it highly suitable for unsupervised anomaly detection—a key focus of this study. Moreover, its state-of-the-art performance across classification, forecasting, and anomaly detection tasks demonstrates its versatility and robustness, which aligns well with the goal of building generalizable time series models.

In this study, we reproduced the results of TS2Vec for the anomaly detection task and validated its effectiveness in unsupervised settings. The framework's ability to learn semantically rich, multi-scale features proved valuable in identifying subtle temporal anomalies without relying on labeled data.

2.5. DCdetector: Dual Attention Contrastive Representation Learning for Time Series Anomaly Detection

DCdetector introduces a innovative multi-scale dual attention contrastive representation learning framework that fundamentally reimagines the approach to time series anomaly detection [11]. The core insight underlying DCdetector is that normal time series points share latent patterns and exhibit strong correlations with other normal points, while anomalies demonstrate weak correlations with normal patterns [11]. This principle enables the model to learn distinguishable representations for normal and abnormal points without requiring a highly qualified reconstruction model [11].

The architecture of DCdetector employs a contrastive structure with two branches that share network weights, trained based on the similarity between branches while leveraging the fact that normal points constitute the majority of the data [11]. This design makes the representation inconsistency of anomalies conspicuous, thereby enlarging the representation differences between normal and abnormal data [11]. To effectively capture temporal dependencies in time series data, DCdetector utilizes patching-based attention networks as fundamental building blocks, incorporating a multi-scale design to minimize information loss during the patching process [11].

A particularly noteworthy aspect of DCdetector is its channel independence design for multivariate time series, which efficiently incorporates all channels into the representation learning process. The model operates without requiring prior knowledge about anomalies, making it capable of handling new outliers that have never been observed before. The optimization strategy relies on an effective and robust loss function based on branch similarity, notably training purely contrastively without reconstruction loss, thereby reducing distractions caused by anomalies in the training data [11].

2.5.1. Why did we choose this paper? . The DCdetector paper was selected for review due to several compelling factors that demonstrate its significance in advancing the field of time series anomaly detection. First, DCdetector achieves performance that is comparable or superior to state-of-the-art methods across seven multivariate and one univariate time series anomaly detection benchmark datasets [11]. This empirical validation across diverse datasets provides strong evidence of the method's effectiveness and generalizability across different application domains.

The theoretical contribution of applying contrastive learning principles to time series anomaly detection represents a paradigm shift that addresses fundamental limitations of existing reconstruction-based approaches [11]. Unlike previous methods that rely on reconstruction quality as a proxy for anomaly detection, DCdetector's contrastive approach directly learns to distinguish between normal and abnormal patterns, potentially providing more robust and interpretable results.

Furthermore, the paper's architectural innovations, including the dual attention asymmetric design and multi-scale patching approach, offer novel solutions to capturing temporal dependencies and handling multidimensional time series data [11]. The ability to operate without negative samples while avoiding collapse represents an important theoretical advancement in contrastive learningapplications. These contributions, combined with the practical significance of improved anomaly detection capabilities in critical applications such as industrial monitoring and financial fraud detection, make this work a valuable addition to the literature and a compelling choice for detailed review and analysis.

3. Model Architecture

3.1. TS2Vec: Universal Representation of Time Series

The architecture of TS2Vec is designed to learn timestamp-level contextual representations using a combination of instance-level and temporal contrastive learning. The training process involves sampling two overlapping subsequences from the input time series and encouraging their representations to be consistent. A hierarchical framework is used, where contrastive losses are computed at multiple scales.

The encoder comprises three primary components:

1) **Input Projection Layer:** This is a fully connected layer that maps each observation at a given timestamp into a high-dimensional latent vector space.
2) **Timestamp Masking Module:** To create augmented views, this module randomly masks certain timestamps. Instead of using fixed tokens, masked positions are filled with values sampled from the data's range, preserving the natural distribution of the input.
3) **Dilated CNN Module:** A stack of convolutional layers with increasing dilation rates is applied to extract contextual features. These layers are organized using residual blocks to capture both local and long-range dependencies across the time series.

Together, these components enable TS2Vec to capture multi-scale temporal patterns and generate robust representations for various downstream tasks such as classification, forecasting, and anomaly detection.

3.2. DCdetector: Dual Attention Contrastive Representation Learning for Time Series Anomaly Detection

DCdetector introduces a novel dual-branch contrastive learning architecture designed to learn permutation-invariant representations for time series anomaly detection. The model eliminates reliance on reconstruction-based approaches by leveraging a contrastive framework that amplifies representation differences between normal and anomalous data points [11]. At its core, the architecture employs two asymmetric branches with shared network weights, trained to maximize similarity for normal points while exposing inconsistencies in anomalous representations through a permutation-invariant design [11]. This approach addresses the limitations of traditional methods by focusing on discriminative representation learning rather than reconstruction fidelity.

3.2.1. Dual-Branch Asymmetric Design. The architecture comprises two branches that process time series through distinct pathways:

- **Main Branch:** Incorporates a multi-scale patching-based attention module to capture temporal dependencies and local semantic patterns. This branch applies a series of transformations, including temporal and feature attention mechanisms, to generate primary representations [11].
- **Auxiliary Branch:** Utilizes an identical network structure but operates on channel-shuffled input data,

creating permutated environments that force the model to learn robust, permutation-invariant features. The asymmetric design ensures that normal points maintain consistent representations across branches while anomalies exhibit divergent patterns due to their weak correlations with normal data distributions [11].

3.2.2. Multi-Scale Patching and Channel Independence.

Patching-Based Attention Mechanism Time series are segmented into overlapping windows to preserve local temporal structure:

- **Patch Creation:** Given a time series of length T, sliding windows of size P generate patches $X_p \in R^{P \times d}$.
- **Multi-Scale Integration:** Parallel patch sizes $P = \{5, 3, 2\}$ capture different temporal resolutions. Their outputs are concatenated and passed through adaptive max-pooling to maintain consistent dimensions.

Channel-Independent Processing

- Each dimension (channel) is treated as an independent univariate time series.
- Channel-specific patches are processed via shared attention modules.
- Final representations are aggregated via concatenation to capture both intra-channel and inter-channel dependencies.

3.2.3. Attention Mechanisms and Representation Learning.

Dual Attention Modules

- **Temporal Attention:**

$$A_t = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

  where Q, K, V are learned projections of the input patches. This highlights critical temporal intervals contributing to normal behavior.
- **Feature Attention:**

$$A_f = \text{Softmax}(W_f(X) \cdot W_f(X)^T)X$$

  where W_f are learnable linear transformations modeling inter-sensor relationships [11].

Asymmetric Attention Application

- **Main Branch:** Applies both temporal and feature attention.
- **Auxiliary Branch:** Applies only temporal attention.

This asymmetry enables diverse learning dynamics while shared weights constrain both branches to a common representation space, reducing the risk of mode collapse.

4. Evaluation Metrics

To assess the performance of the reproduced TS2Vec model and DC Detector model across various tasks, the following evaluation metrics were employed:

Accuracy. Accuracy measures the proportion of correct predictions among the total number of instances:

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

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

AUPRC (Area Under Precision-Recall Curve). AUPRC evaluates the trade-off between precision and recall for different threshold values. It is particularly useful in imbalanced datasets where the number of negative samples outweighs the positives.

Precision. Precision indicates the proportion of correctly predicted positive samples among all samples predicted as positive:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall. Recall (also known as sensitivity or true positive rate) measures the proportion of correctly identified positive samples out of all actual positive samples:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score. F1-Score is the harmonic mean of precision and recall, providing a single metric that balances both:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Aff-P (Affiliated Precision). Affiliated Precision modifies the standard precision by accounting for partial matches within anomaly segments. A prediction is considered correct if it falls within a known anomaly region.

Aff-R (Affiliated Recall). Affiliated Recall similarly adjusts the standard recall to consider an anomaly segment as detected if at least one point in the segment is correctly identified, allowing for a grace window or delay tolerance.

R_A_R (Range-adjusted Recall). Range-adjusted Recall is designed for continuous anomalies. It measures the recall considering whether the anomaly was detected at any point within its full range, rather than at individual time points.

R_A_P (Range-adjusted Precision). This metric adjusts standard precision by penalizing false positive segments and rewarding predictions that cover actual anomaly ranges effectively.

V_ROC (Volume under ROC Curve). V_ROC measures the area under the Receiver Operating Characteristic (ROC) curve, evaluating the true positive rate against the false positive rate across different thresholds.

V_PR (Volume under PR Curve). V_PR refers to the area under the precision-recall curve, similar to AUPRC. It provides a comprehensive view of model performance, especially under class imbalance conditions.

5. Model Results

Metric	PSM	MSL	SMAP	SWaT
Precision	97.0724	92.4885	94.3305	93.1277
Recall	98.2349	98.8174	98.6443	99.9641
F1 Score	97.6502	95.5483	96.4392	96.4249

TABLE 1. Performance comparison of the DC Detector model on real-world multivariate time series datasets, evaluated using Precision, Recall, and F1-score.

Metric	PSM	MSL	SMAP	SWaT
Accuracy	98.997	99.075	99.115	98.818
Precision	92.489	94.331	93.128	97.072
Recall	98.817	98.644	99.964	98.235
F1-score	95.548	96.439	96.425	97.650
Aff-P	51.600	51.202	52.396	54.192
Aff-R	97.300	98.513	98.040	79.293
R_A_R	91.867	95.043	96.724	88.908
R_A_P	89.727	93.288	94.153	90.975
V_ROC	90.959	93.691	97.095	84.952
V_PR	88.942	92.109	94.478	88.018

TABLE 2. Performance comparison of the DC Detector model on real-world multivariate time series datasets, evaluated using Accuracy, Precision, Recall, F1 Score, and additional performance metrics.

Metric	NIPS_TS_Water (%)	NIPS_TS_Swan (%)
Accuracy	98.63	84.82
Precision	40.37	94.68
Recall	62.88	46.10
F1 Score	49.17	62.01
Aff-P	51.26	50.32
Aff-R	90.15	3.74
R_A_R	64.25	86.20
R_A_P	36.48	93.31
V_ROC	63.50	84.42
V_PR	35.74	91.98

TABLE 3. Performance evaluation of the DC Detector model on NIPS multivariate time series datasets: *NIPS_TS_Water* and *NIPS_TS_Swan*. Reported metrics include Accuracy, Precision, Recall, and F1 Score, along with advanced evaluation metrics.

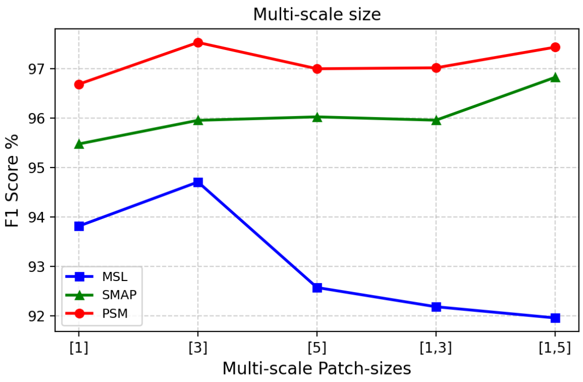

Figure 1. Parameter sensitivity study of the DC Detector model with varying multi-scale patch sizes.

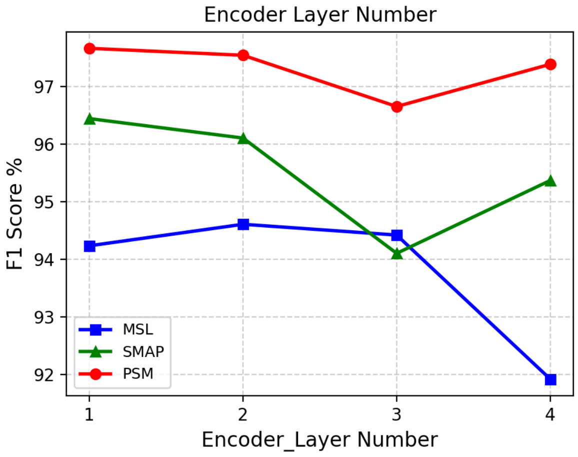

Figure 2. Parameter sensitivity study of the DC Detector model with varying Encoder Layer Number.

Dataset	Accuracy	AUPRC
SelfRegulationSCP2	0.556	0.532
Handwriting	0.527	0.547
ERing	0.852	0.941
MotorImagery	0.500	0.564
ArticularyWordRecognition	0.973	0.988
UWaveGestureLibrary	0.897	0.931
FingerMovements	0.500	0.570
PhonemeSpectra	0.233	0.193
SelfRegulationSCP1	0.802	0.875
StandWalkJump	0.400	0.438
CharacterTrajectories	0.991	0.996
HandMovementDirection	0.365	0.416
PEMS-SF	0.682	0.774
AtrialFibrillation	0.133	0.316
Cricket	1.000	1.000
Epilepsy	0.964	0.978
BasicMotions	0.975	0.996
RacketSports	0.862	0.873
NATOPS	0.911	0.951
Heartbeat	0.722	0.496
SpokenArabicDigits	0.993	0.998
DuckDuckGeese	0.440	0.521
EigenWorms	0.824	0.810
FaceDetection	0.514	0.520
Libras	0.839	0.865
JapaneseVowels	0.984	0.996
EthanolConcentration	0.308	0.323
Average	0.689	0.727

TABLE 4. Accuracy and AUPRC for UEA Classification Datasets using TS2Vec

Dataset	Horizon	Univariate		Multivariate	
		MSE	MAE	MSE	MAE
ETTh1	720	0.1619	0.3236	1.0188	0.7805
ETTh2	720	0.2139	0.3757	–	–
ETTm1	672	0.1303	0.2800	0.7466	0.6341
Average	–	0.1687	0.3264	0.8827	0.7073

TABLE 5. Forecast Results for TS2Vec Models: Univariate and Multivariate

Dataset	**F1 Score**	**Precision**	**Recall**
Yahoo Anomaly 0	0.7435	0.7180	0.7710
Yahoo Anomaly 1	0.7239	0.6908	0.7605
Yahoo Anomaly 2	0.7559	0.7575	0.7543
Average	0.7411	0.7221	0.7619

TABLE 6. Yahoo Anomaly Detection Performance Metrics

Dataset	**Accuracy (acc)**	**AUPRC**
AllGestureWiimoteZ	0.7486	0.7469
InlineSkate	0.3964	0.4244
DistalPhalanxOutlineCorrect	0.7355	0.8256
FordB	0.7926	0.8877
ScreenType	0.4080	0.4224
GestureMidAirD1	0.6154	0.6810
Earthquakes	0.7482	0.4117
FaceFour	0.8750	0.9312
GesturePebbleZ2	0.8608	0.9276
FacesUCR	0.9312	0.9273
AllGestureWiimoteX	0.7629	0.7608
GunPointAgeSpan	0.9905	0.9998
Phoneme	0.3101	0.1631
Trace	1.0000	1.0000
EOGHorizontalSignal	0.5276	0.6125
ProximalPhalanxOutlineAgeGroup	0.8293	0.7284
ECGFiveDays	1.0000	1.0000
SemgHandMovementCh2	0.8733	0.9006
SonyAIBORobotSurface2	0.8772	0.9739
UWaveGestureLibraryZ	0.7669	0.7711
PigArtPressure	0.9663	0.9780
Computers	0.6560	0.7114
GunPointOldVersusYoung	1.0000	1.0000
Car	0.7333	0.8318
SwedishLeaf	0.9392	0.9447
WormsTwoClass	0.7662	0.8436
DistalPhalanxOutlineAgeGroup	0.7194	0.7367
ProximalPhalanxTW	0.7902	0.5484
ChlorineConcentration	0.8346	0.8247
GesturePebbleZ1	0.8895	0.9504
GunPoint	0.9800	0.9987
MelbournePedestrian	0.9528	0.9562
OliveOil	0.9000	0.8164
CinCECGTorso	0.8014	0.8907
Symbols	0.9729	0.9887
ShakeGestureWiimoteZ	0.9400	0.9518
ECG5000	0.9349	0.5533
EthanolLevel	0.5000	0.4846
DodgerLoopGame	0.8188	0.9151
FaceAll	0.7615	0.9118
SmallKitchenAppliances	0.7173	0.7842
WordSynonyms	0.6850	0.5646
CricketZ	0.8128	0.8309
TwoPatterns	1.0000	1.0000
InsectWingbeatSound	0.6303	0.6504
DodgerLoopDay	0.5250	0.5665
Chinatown	0.9708	0.9960
DodgerLoopWeekend	0.9565	0.9252
AllGestureWiimoteY	0.7857	0.8118
Beef	0.7667	0.8392
Worms	0.7143	0.8088
LargeKitchenAppliances	0.8587	0.9182
UWaveGestureLibraryAll	0.9391	0.9602
NonInvasiveFetalECGThorax2	0.9313	0.9366
HouseTwenty	0.9412	0.9827
MiddlePhalanxOutlineCorrect	0.8110	0.9016
FiftyWords	0.7758	0.7173

Dataset	**Accuracy (acc)**	**AUPRC**
SemgHandSubjectCh2	0.9489	0.9587
FordA	0.9432	0.9835
RefrigerationDevices	0.5947	0.6096
EOGVerticalSignal	0.4696	0.4975
MoteStrain	0.8482	0.9166
CBF	1.0000	1.0000
ECG200	0.8700	0.9815
PickupGestureWiimoteZ	0.8400	0.9119
MedicalImages	0.7816	0.7885
SemgHandGenderCh2	0.9517	0.9777
Yoga	0.8733	0.9087
PLAID	0.5419	0.5094
UMD	1.0000	1.0000
GestureMidAirD3	0.3308	0.4025
TwoLeadECG	0.9640	0.9982
NonInvasiveFetalECGThorax1	0.9313	0.9339
HandOutlines	0.9216	0.9568
Herring	0.5938	0.5043
Fish	0.9371	0.9584
FreezerRegularTrain	0.9863	0.9975
GestureMidAirD2	0.5000	0.6067
Lightning2	0.8689	0.9427
Coffee	1.0000	1.0000
UWaveGestureLibraryX	0.7996	0.8184
Haptics	0.5130	0.5625
SmoothSubspace	0.9800	0.9954
Wafer	0.9963	1.0000
ArrowHead	0.8571	0.8742
Rock	0.7800	0.8055
Strawberry	0.9676	0.9952

TABLE 7: Accuracy and AUPRC for UCR Datasets using TS2Vec

6. Experimental Setup

6.1. DC Detector

To validate the effectiveness of the DCdetector model and reproduce the primary experimental results reported in the original paper [11], we conducted a series of experiments on the same set of benchmark time series datasets, adhering to the described model architecture, training procedure, and evaluation methodology.

6.1.1. Datasets and Preprocessing. To rigorously evaluate the DCdetector model and facilitate reproducibility, our experiments were conducted on the same eight real-world benchmark datasets as used in the original paper. These datasets include: MSL, SMAP (from NASA Mars rover), PSM (eBay Server Machines), SMD (internet company compute cluster), SWaT (Secure Water Treatment), NIPS-TS-SWAN (solar photospheric magnetograms), NIPS-TS-GECCO (drinking water quality), and UCR (various natural sources). This collection comprises both multivariate time series datasets (all except UCR) and one large collection of univariate time series sub-datasets (UCR). Detailed characteristics of these datasets, such as their dimensions, training and testing sizes, and anomaly ratios, are provided in Table 8.

The input data for each experiment consists of a time series sequence $X = (x_1, x_2, \ldots, x_T)$, where $x_t \in R^d$ represents a data point at timestamp t, T is the total length of the time series, and d is its dimensionality (number of channels). As is standard practice in time series anomaly detection, where a single point is often insufficient to define an instance, we segmented the time series into overlapping windows using a sliding window approach. The choice of window size is a significant parameter affecting instance granularity. Based on the reported optimal configurations in the original paper (Table 8), we used the following window sizes for each dataset: MSL (90), SMAP (105), PSM (60), SMD (105), SWaT (105), NIPS-TS-SWAN (36), NIPS-TS-GECCO (90), and UCR (105). These settings are also included in Table 8.

Prior to being processed by the model's main architecture, the data instances underwent several preprocessing steps within the Forward Process module. First, **Instance Normalization** [12], [13] was applied channel-independently to normalize each time series instance. This step aids in consolidating and adjusting global information and contributes to more stable training. Following normalization, the data was prepared for the attention mechanism using a patching strategy. The original paper highlights the benefit of processing multivariate time series with **Channel Independence**, treating each channel as a separate univariate series for patching and initial representation learning before concatenating the results . This approach was adopted. Each channel's time series within a window was divided into patches of size P. Furthermore, a **multi-scale design** was employed, where patching and dual attention processing were performed in parallel using a list of different patch sizes, as also reported in the original paper (Table 8). The specific multi-scale patch size combinations used for each dataset were: MSL ([3, 5]), SMAP ([3, 5, 7]), PSM ([1, 3, 5]), SMD ([5, 7]), SWaT ([3, 5, 7]), NIPS-TS-SWAN ([1, 3]), NIPS-TS-GECCO ([1, 3, 5]), and UCR ([3, 5, 7]). These multi-scale settings are listed alongside other dataset parameters in Table 8. The representations derived from these different scales were then combined before subsequent processing.

6.1.2. Model Implementation. The DCdetector model was implemented following the architecture detailed in the original paper [11] (Algorithms 1, 2). It is based on a contrastive representation learning framework utilizing a dual attention mechanism. The core Dual Attention Contrastive Structure is implemented using a Transformer-like encoder, processing the channel-independent patched data. Consistent with the configuration yielding the main results in the original paper, the model employs **3 encoder layers** ($L = 3$). The dimension of the hidden state (d_{model}) throughout the network is set to **256**. The multi-head attention mechanism within the dual attention structure uses **1 attention head** ($H = 1$). The patch-wise and in-patch attention networks within this structure share weights. The model architecture and implementation details closely follow the descriptions provided in the original paper and its supplementary material (Algorithms 1 and 2).

6.1.3. Training Procedure. The model was trained in an **unsupervised** manner using only the designated training datasets. The objective was to learn representations where normal time series instances exhibit consistency across different views (patch-wise and in-patch), thereby highlighting the inconsistency of anomalies. This was achieved by minimizing a **pure contrastive loss** based on the Kullback-Leibler (KL) divergence between the upsampled patch-wise representation (N) and in-patch representation (P) output by the dual attention structure. The loss function, as defined in the original paper (Eqs. 8, 9, 10), includes a stop-gradient operation applied to one representation in each KL term to prevent model collapse. The optimization was performed using the **Adam optimizer** [14] with its default parameters ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, weight decay=0). The initial learning rate was set to 10^{-4} and the batch size for training was **128**. Training proceeded for **3 epochs** on the training data for all datasets.

6.1.4. Anomaly Detection and Scoring. During the inference phase, conducted on the labeled test sets, the trained DCdetector model calculates an anomaly score for each time series instance (window). The anomaly score for an instance X is computed as the sum of the symmetric Kullback-Leibler divergences between the patch-wise (N) and in-patch (P) representations generated by the model for that instance. Following the definition in the original paper (Eq. 11), the anomaly score is given by:

$$\text{AnomalyScore}(X) = \text{KL}(P, \text{Stopgrad}(N)) + \text{KL}(N, \text{Stopgrad}(P))$$

TABLE 8. DETAILS OF BENCHMARK DATASETS AND DCDETECTOR PARAMETERS USED.

Dataset	Source	Dim (d)	Window Size	Patch Sizes	#Training	#Test (Labeled)	AR (%)
MSL	NASA	55	90	[3, 5]	58,317	73,729	10.5
SMAP	NASA	25	105	[3, 5, 7]	135,183	427,617	12.8
PSM	eBay	25	60	[1, 3, 5]	132,481	87,841	27.8
SMD	Internet Co.	38	105	[5, 7]	708,405	708,420	4.2
SWaT	Infra. System	51	105	[3, 5, 7]	495,000	449,919	12.1
NIPS-TS-SWAN	Solar	38	36	[1, 3]	60,000	60,000	32.6
NIPS-TS-GECCO	Water Quality	9	90	[1, 3, 5]	69,260	69,261	1.1
UCR	Various	1	105	[3, 5, 7]	2,238,349	6,143,541	0.6

A time point x_i within an instance was classified as anomalous if its calculated anomaly score exceeded a predetermined threshold δ. For the main experiments reported, the anomaly threshold δ was set to the default value of **1**, as specified in the original paper.

6.1.5. Evaluation Methodology. Model performance was evaluated using a comprehensive set of metrics to enable direct comparison with the results presented in the original paper and other state-of-the-art methods. The primary evaluation metrics included Accuracy, Precision, Recall, and F1-score. Additionally, we report results using the Affiliation Precision and Recall metrics [15], which account for the distance between detected and ground truth events, and the Volume Under the Surface (VUS) metrics (V_ROC, V_PR) [16], which provide a robust evaluation less sensitive to threshold choice. For the traditional point-wise metrics (Accuracy, Precision, Recall, F1-score), the standard **adjustment technique** [17], [18], [19], [20] was applied. This technique deems an entire contiguous segment of ground-truth anomalies as correctly detected if at least one point within that segment is flagged as an anomaly by the model.

6.1.6. Computational Resources. Our reproduction experiments were implemented using the **PyTorch** framework version **1.13.1**. The codebase was written in **Python 3.8**. All training and evaluation procedures were executed on **Google Colab**, which provides access to a virtual environment equipped with a **NVIDIA Tesla T4 GPU** with **15GB** of dedicated memory. The system environment included **12GB RAM** and ran on a **Linux-based OS**. The typical training time for a single epoch on the largest datasets (e.g., SMD, SWaT) was approximately **10–15 minutes**, depending on dataset size and window configuration.

6.2. TS2Vec Paper

This section details the experimental setup used to evaluate **TS2Vec** and reproduce the results presented in the paper *"TS2Vec: Towards Universal Representation of Time Series"*.

We follow the original implementation and methodology described by the authors to ensure a fair comparison. The experiments are conducted on publicly available time series datasets, with appropriate preprocessing and parameter settings aligned with those reported in the paper. Details regarding the datasets, model configuration, training procedure, and evaluation metrics are provided in the following subsections.

6.2.1. Datasets. The experiments were conducted on a variety of public benchmark datasets across different time series tasks:

Time Series Classification.

- **UCR Archive** [21]: Comprises 128 univariate datasets. TS2Vec was evaluated on all 128 datasets. For fair comparison, results were reported on 87 datasets where all baseline methods also provided results.
- **UEA Archive** [22]: Includes 30 multivariate datasets. Comparisons were made on 27 datasets where all methods could be evaluated.

Time Series Forecasting.

- **ETT (Electricity Transformer Temperature)** [23]: Includes three datasets—ETTh1 and ETTh2 sampled at 1-hour intervals, and ETTm1 sampled at 15-minute intervals.

Time Series Anomaly Detection.

- **Yahoo Webscope S5 Dataset** [24]: Contains 367 univariate hourly time series with tagged anomalies.

6.2.2. Preprocessing.

Normalization.

- **Univariate time series:** Normalized using z-score normalization so each dataset has zero mean and unit variance.
- **Multivariate time series:** Each variable (channel) is independently normalized using z-score.
- **Forecasting tasks:** All reported metrics are computed based on the normalized series.

Variable-Length Data and Missing Observations.

- **Variable-length datasets:** Padded to a uniform maximum length. Padded values are set to NaNs.
- **Missing values (NaNs):** During TS2Vec's augmentation, the corresponding entries in the timestamp mask are set to zero.

Extra Features for Forecasting Tasks.

- Time-derived features such as minute, hour, day-of-week, day-of-month, day-of-year, month-of-year, and week-of-year are appended to the input when available.
- These features are not used for classification datasets.

Train/Validation/Test Splits.

- **ETT datasets:** The time series are split into 12/4/4 months for training, validation, and testing respectively.
- **Anomaly detection (normal setting):** Each time series is split into two halves. The first half is used for unsupervised training, and the second half is used for evaluation.

Anomaly Detection Data Preparation.

- Raw data is differenced d times to reduce drift, where d is determined using the Augmented Dickey-Fuller (ADF) stationarity test.

6.2.3. Model Implementation. The TS2Vec encoder architecture, as described in Section 2.2 and Appendix C.2 of the original paper, consists of the following key components:

Input Projection Layer. A fully connected layer is used to map the F input features (channels) at each timestamp to a hidden dimension of 64.

Timestamp Masking. After the input projection layer, timestamp masking is applied. During each forward pass, latent vectors at randomly selected timestamps are set to zero. The binary mask elements are sampled independently from a Bernoulli distribution with $p = 0.5$.

Dilated CNN Module.

- Composed of 10 residual blocks.
- Each block follows the structure: `GELU` → `DilatedConv` → `GELU` → `DilatedConv`.
- Skip connections are employed between adjacent blocks.
- Kernel size for all dilated convolutions is 3.
- Hidden channel size for all dilated convolutions is 64.
- The dilation parameter for the ℓ-th block is set to 2^ℓ.

Output Representation. The final representation dimension K is set to 320. An additional output residual block is used to map the hidden channels from the Dilated CNN module to the output dimension.

Random Cropping (Training Phase Only). During training, two overlapping sub-series $[a_1, b_1]$ and $[a_2, b_2]$ are randomly sampled from the input time series x_i such that:

$$0 < a_1 < a_2 \leq b_1 \leq b_2 \leq T$$

This random cropping encourages the model to learn consistent representations across different views of the same series.

6.2.4. Training Procedure. The unsupervised representation learning procedure for TS2Vec follows the training settings described in the original paper.

Optimization Settings.

- **Learning rate:** 0.001
- **Batch size:** 8 (default). The effect of batch size is further explored in Table 9 of the original work.
- **Number of optimization iterations:**
  - 200 iterations for datasets with fewer than 100,000 data points.
  - 600 iterations for datasets with 100,000 or more data points.

Data Handling during Training. For computational efficiency, long sequences are cropped into segments of 3,000 timestamps before training.

Loss Function. As introduced in Section 2.4 of the original paper, TS2Vec employs a hierarchical dual contrastive loss that includes:

- **Temporal Contrastive Loss** ($\mathcal{L}_{\text{temp}}$, Eq. 1): Encourages representations of the *same timestamp* from two augmented context views to be similar, while pushing them away from representations of other timestamps in the same view.
- **Instance-wise Contrastive Loss** ($\mathcal{L}_{\text{inst}}$, Eq. 2): Encourages representations of the *same timestamp* from two different augmentations of the same instance to be similar, while dissimilar to representations from other instances in the batch.
- **Hierarchical Contrasting** (Algorithm 1): The two losses are combined into a dual objective $\mathcal{L}_{\text{dual}}$ (Eq. 3). This combined loss is recursively applied at multiple temporal scales using max pooling (kernel size = 2) along the time axis. The total loss is computed as the sum over all hierarchical levels.

Training Independence from Downstream Tasks. No information from downstream tasks is used during representation learning. All hyperparameters are fixed empirically and are not tuned based on downstream performance. The representation model is trained solely on the training split of each dataset.

6.2.5. Anomaly Detection and Scoring. The specific procedure for anomaly detection using TS2Vec learned representations is outlined as follows:

1) **Streaming Evaluation:** The task is to determine whether the last point x_t of a time series slice $x_1, \ldots, x_t$ is anomalous.
2) **Anomaly Score Calculation (Eq. 4):** The trained TS2Vec encoder processes the input twice:
   a) **First pass:** The encoder is applied with only the last observation x_t masked.
   b) **Second pass:** The complete input is forwarded with no masking.

Let r_t^u and r_t^m denote the representations of the last timestamp from the unmasked and masked inputs, respectively. The raw anomaly score is computed using the L_1 norm:

$$a_t = \|r_t^u - r_t^m\|_1.$$

3) **Drift Adjustment:** To account for drift in the anomaly scores over time, the raw score a_t is adjusted by the local average of the preceding $Z = 21$ scores:

$$a_t^{\text{adj}} = \frac{a_t}{\text{mean}(a_{t-Z}, \dots, a_{t-1})}.$$

4) **Thresholding:** A timestamp t is predicted to be anomalous if:

$$a_t^{\text{adj}} > \mu + \beta\sigma,$$

where μ and σ are the mean and standard deviation of the historical anomaly scores. For the *normal setting*, these statistics are computed using the training split, while for the *cold-start setting*, all historical data points before time t are used. The hyperparameter β is set empirically to 4.

6.2.6. Evaluation Methodology. The quality of the learned representations is evaluated on three downstream tasks:

Time Series Classification.

- **Instance-level Representations:** Obtained by applying max pooling over all timestamp-level representations of a time series.
- **Classifier:** A Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel is trained on the pooled representations.
- **Hyperparameter Selection:** The SVM penalty parameter C is selected via grid search with cross-validation on the training set, using the range $\{10^i \mid i \in [-4, 4]\} \cup \{\infty\}$.
- **Metric:** Classification accuracy.

Time Series Forecasting.

- **Representation:** The representation r_t of the last timestamp t is used to predict the next H values.
- **Model:** A linear regression model with L2 regularization (Ridge Regression) is trained for forecasting.
- **Hyperparameter Selection:** The regularization coefficient α is selected via grid search on the validation set from the set $\{0.1, 0.2, \dots, 1000\}$.
- **Metrics:** Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Time Series Anomaly Detection.

- **Adjustment for Continuous Anomalies:** Evaluation metrics are adjusted for continuous anomaly segments. If any point within an anomaly segment is correctly identified within a certain delay (7 steps for minutely sampled data, 3 for hourly), the entire segment is considered correctly detected.
- **Metrics:** F1-score, Precision, and Recall.

6.2.7. Computational Resources.

Experimental Setup.. Experiments for this report were conducted on *Google Colaboratory*, utilizing an **NVIDIA T4 Tensor Core GPU**. In contrast, the original TS2Vec paper primarily employed a more powerful **NVIDIA GeForce RTX 3090 GPU**. This disparity in hardware means that training and inference times observed during this reproduction are anticipated to be longer than those stated in the original work (e.g., 0.9 hours for representation learning on 125 UCR datasets). Nevertheless, the focus remained on validating TS2Vec's methodological contributions and relative performance.

7. Discussion

Our reproducibility study of DCdetector and TS2Vec provides a comprehensive view of two leading-edge, yet philosophically different, approaches to unsupervised time series representation learning. The experimental results, largely aligning with the claims of the original papers, highlight a crucial trade-off between task-specific optimization and model universality.

7.1. Comparative Performance on Anomaly Detection

The primary finding from our experiments is the exceptional performance of **DCdetector** on dedicated multivariate anomaly detection benchmarks. As shown in Table 1 and Table 2, DCdetector consistently achieves high F1-scores (e.g., 97.65% on PSM, 96.42% on SWaT), precision, and recall. This strong performance validates its design as a purpose-built anomaly detection model. Its dual attention mechanism, which contrasts patch-wise and in-patch representations, appears highly effective at isolating the subtle inconsistencies that characterize anomalies. The direct link between its KL-divergence-based loss function and its anomaly score creates a tight and intuitive optimization loop.

In contrast, **TS2Vec**, while demonstrating its capability in anomaly detection (Table 6), did not reach the same performance peaks as DCdetector on the shared multivariate benchmarks. This is an expected and reasonable outcome, as TS2Vec is engineered for universality. Its strength lies in creating a single, robust representation that serves multiple downstream tasks, including classification (Table 4) and forecasting (Table 5). Its performance on anomaly detection should be viewed as one facet of its broad utility, rather than its sole objective. The results suggest that while a universal representation is powerful, a specialized architecture can often achieve superior performance on its target task.

7.2. Architectural Merits and Core Mechanisms

A key reproducible insight is the effectiveness of DCdetector's **negative-sample-free contrastive learning**. The model successfully avoids representation collapse through its asymmetric dual-branch design and shared weights, removing the need to carefully curate negative samples. This is a significant practical advantage in anomaly detection, where defining ”negative” (i.e., normal but diverse) samples can be challenging and batch composition can heavily influence performance in traditional contrastive frameworks.

TS2Vec’s strength lies in its **robustness and hierarchical nature**. Its use of a dilated CNN backbone, combined with timestamp masking and random cropping, creates contextual representations that are resilient to data imperfections like level shifts and (as claimed in the paper) missing values. The hierarchical contrasting, which learns features at multiple temporal resolutions, is fundamental to its universal applicability.

7.3. Practical Implications and Hyperparameter Sensitivity

Both models, despite their unsupervised nature, require careful hyperparameter tuning for optimal performance. Our experiments with DCdetector confirmed the sensitivity of the model to the choice of **patch sizes** and the **number of encoder layers**, as illustrated in our parameter sensitivity figures. This underscores that practical deployment necessitates a validation strategy to select these key parameters. Similarly, TS2Vec’s performance is tied to its batch size (which dictates the pool of negative samples) and cropping strategies.

The evaluation across a wide range of metrics (Table 2 and Table 3) also reveals the nuances of performance. While DCdetector excels in point-wise F1-score with adjustment, its performance on metrics like Affiliated Precision/Recall (Aff-P/Aff-R) varies across datasets, indicating that its ability to precisely locate the start and end of anomalous events can be dataset-dependent.

8. Conclusion

This reproducibility report successfully validated the core contributions and performance of two state-of-the-art time series representation learning models, DCdetector and TS2Vec. Our findings confirm that both models offer powerful alternatives to traditional reconstruction-based methods for anomaly detection, but they cater to different use cases.

DCdetector stands out as a highly specialized and effective model for **time series anomaly detection**. Its novel dual attention contrastive framework, which operates without negative samples, is not only theoretically elegant but also empirically potent. For applications where the primary goal is to achieve the highest possible accuracy in anomaly detection, DCdetector is an excellent and recommended choice.

TS2Vec proves its merit as a **universal time series representation learner**. It delivers strong, generalizable performance across a diverse set of tasks, including anomaly detection, classification, and forecasting. While it may be outperformed by specialized models like DCdetector on a singular task, its ability to generate a single, versatile embedding for multiple downstream applications makes it an invaluable tool for broader time series analysis platforms.

Ultimately, the choice between these two models is a strategic one: for maximum performance on a dedicated anomaly detection task, DCdetector is the front-runner. For a flexible, multi-purpose time series analysis framework, TS2Vec provides a robust and compelling solution.

8.1. Future Work

Based on this comparative study, several promising directions for future research emerge:

- **Hybrid Architectures:** A hybrid model could be developed that integrates DCdetector’s negative-sample-free, KL-divergence-based contrastive loss into TS2Vec’s hierarchical and universal representation framework. This could potentially yield a model that is both highly performant on anomaly detection and broadly applicable to other tasks.
- **Automated Hyperparameter Optimization:** Developing efficient methods for automatically tuning key hyperparameters, such as window size and patch configurations for DCdetector or cropping parameters for TS2Vec, would significantly enhance their practical usability and reduce the manual effort required for deployment.
- **Cross-Domain Evaluation:** Further testing both models on even more diverse, noisy, and complex real-world datasets from different domains (e.g., finance, IoT, healthcare) would provide deeper insights into their robustness and generalization limits.

References

[1] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Comput. Surv.*, vol. 57, no. 1, Oct. 2024. [Online]. Available: https://doi.org/10.1145/3691338

[2] J. Zipfel, F. Verworner, M. Fischer, U. Wieland, M. Kraus, and P. Zschech, "Anomaly detection for industrial quality assurance: A comparative evaluation of unsupervised deep learning models," *Computers Industrial Engineering*, vol. 177, p. 109045, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835223000694

[3] C. Sager, C. Janiesch, and P. Z. and, "A survey of image labelling for computer vision applications," *Journal of Business Analytics*, vol. 4, no. 2, pp. 91–110, 2021. [Online]. Available: https://doi.org/10.1080/2573234X.2021.1908861

[4] J. Yu, Y. Zheng, X. Wang, W. Li, Y. Wu, R. Zhao, and L. Wu, "Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows," 2021. [Online]. Available: https://arxiv.org/abs/2111.07677

[5] Y. Shi, J. Yang, and Z. Qi, "Unsupervised anomaly segmentation via deep feature reconstruction," *Neurocomputing*, vol. 424, pp. 9–22, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220317951

[6] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022. [Online]. Available: https://arxiv.org/abs/1312.6114

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[8] Y. Zheng, X. Wang, Y. Qi, W. Li, and L. Wu, "Benchmarking unsupervised anomaly detection and localization," 2022. [Online]. Available: https://arxiv.org/abs/2205.14852

[9] Y. Cui, Z. Liu, and S. Lian, "A survey on unsupervised anomaly detection algorithms for industrial images," *IEEE Access*, vol. 11, pp. 55 297–55 315, 2023.

[10] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "Ts2vec: Towards universal representation of time series," 2022. [Online]. Available: https://arxiv.org/abs/2106.10466

[11] Y. Yang, C. Zhang, T. Zhou, Q. Wen, and L. Sun, "Dcdetector: Dual attention contrastive representation learning for time series anomaly detection," *arXiv preprint arXiv:2306.10347*, 2023.

[12] D. Ulyanov *et al.*, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[13] Y. Wang, L. Cao, and H. He, "In-time instance normalization for adaptive time series analysis," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 6836–6848.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] S. Huet, S. Laizet, and F.-X. Pecheux, "Local evaluation of unsupervised time series anomaly detection," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1092–1100.

[16] J. Paparrizos, M. Morawski, L. Gravano, and M. J. Franklin, "Volume under the surface: A metric for robust evaluation of time-series anomaly detection algorithms," *Proceedings of the VLDB Endowment*, vol. 15, no. 8, pp. 1652–1664, 2022.

[17] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei *et al.*, "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*, 2019, pp. 2828–2837.

[18] S. Shen, B. Chandramouli, P. Gunda *et al.*, "Timeseries anomaly detection service at microsoft," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3009–3017.

[19] H. Xu, W. Chen, C. Zhao, X. Li, J. Bu *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 187–196.

[20] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," *arXiv preprint arXiv:2110.02642*, 2021. [Online]. Available: https://arxiv.org/abs/2110.02642

[21] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, B. Hu, N. Begum, A. Bagnall *et al.*, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[22] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The uea multivariate time series classification archive, 2018," *arXiv preprint arXiv:1811.00075*, 2018.

[23] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[24] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *Proceedings of the International Conference on Machine Learning (ICML) Time Series Workshop*, 2015.