Pattern Recognition 157 (2025) 110874

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

CARLA: Self-supervised contrastive representation learning for time series anomaly detection

Zahra Zamanzadeh Darban [a,*], Geoffrey I. Webb [a], Shirui Pan [b], Charu C. Aggarwal [c], Mahsa Salehi [a]

[a] Monash University, Melbourne, Victoria, Australia
[b] Griffith University, Gold Coast, Queensland, Australia
[c] IBM T. J. Watson Research Center, Yorktown Heights, NY, USA

ARTICLE INFO

Keywords:
Anomaly detection
Time series
Deep learning
Contrastive learning
Representation learning
Self-supervised learning

ABSTRACT

One main challenge in time series anomaly detection (TSAD) is the lack of labelled data in many real-life scenarios. Most of the existing anomaly detection methods focus on learning the normal behaviour of unlabelled time series in an unsupervised manner. The normal boundary is often defined tightly, resulting in slight deviations being classified as anomalies, consequently leading to a high false positive rate and a limited ability to generalise normal patterns. To address this, we introduce a novel end-to-end self-supervised ContrAstive Representation Learning approach for time series Anomaly detection (CARLA). While existing contrastive learning methods assume that augmented time series windows are positive samples and temporally distant windows are negative samples, we argue that these assumptions are limited as augmentation of time series can transform them to negative samples, and a temporally distant window can represent a positive sample. Existing approaches to contrastive learning for time series have directly copied methods developed for image analysis. We argue that these methods do not transfer well. Instead, our contrastive approach leverages existing generic knowledge about time series anomalies and injects various types of anomalies as negative samples. Therefore, CARLA not only learns normal behaviour but also learns deviations indicating anomalies. It creates similar representations for temporally close windows and distinct ones for anomalies. Additionally, it leverages the information about representations' neighbours through a self-supervised approach to classify windows based on their nearest/furthest neighbours to further enhance the performance of anomaly detection. In extensive tests on seven major real-world TSAD datasets, CARLA shows superior performance (F1 and AU-PR) over state-of-the-art self-supervised, semi-supervised, and unsupervised TSAD methods for univariate time series and multivariate time series. Our research highlights the immense potential of contrastive representation learning in advancing the TSAD field, thus paving the way for novel applications and in-depth exploration.

1. Introduction

In many modern applications, data analysis is required to identify and remove anomalies (a.k.a. outliers) to ensure system reliability. Several machine learning algorithms are well-suited for detecting these outliers [1]. In time series data, anomalies can result from different factors, including equipment failure, sensor malfunction, human error, and human intervention. Detecting anomalies in time series data has numerous real-world uses, including monitoring equipment for malfunctions, detecting unusual patterns in IoT sensor data, enhancing the reliability of computer programs and cloud systems, observing patients' health metrics, and pinpointing cyber threats. Time series anomaly detection (TSAD) has been the subject of decades of intensive research, with numerous approaches proposed to address the challenge of spotting rare and unexpected events in complex and noisy data. Statistical methods have been developed to monitor and identify abnormal behaviour [1]. Recent advancements in deep learning techniques have effectively tackled various anomaly detection problems [2]. Specifically, for time series with complex nonlinear temporal dynamics, deep learning methods have demonstrated remarkable performance [3].

Most of these models focus on learning the normal behaviour of data from an unlabelled dataset and, therefore, can potentially predict samples that deviate from the normal behaviour as anomalies. The lack of labelled data in real-world scenarios makes it difficult for models to learn the difference between normal and anomalous behaviours. The

* Corresponding author.
E-mail address: zahra.zamanzadeh@monash.edu (Z.Z. Darban).

https://doi.org/10.1016/j.patcog.2024.110874
Received 5 April 2024; Received in revised form 1 August 2024; Accepted 7 August 2024
Available online 10 August 2024

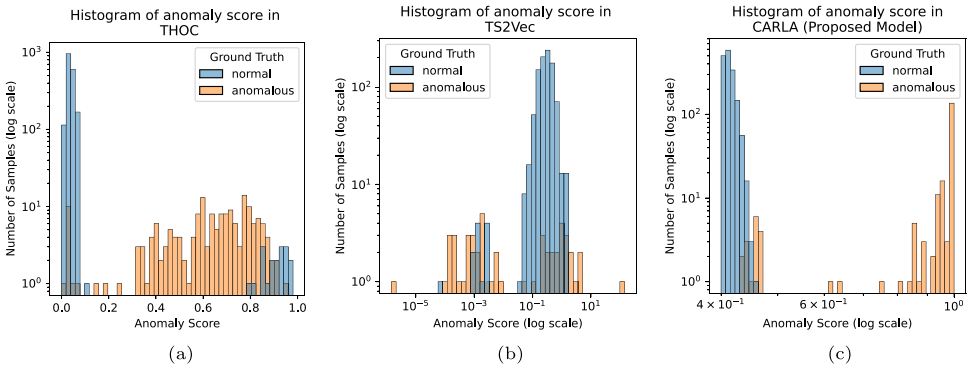

Fig. 1. Histograms of the distribution of anomaly scores produced by (a) THOC [4], (b) and TS2Vec [7] and (c) CARLA models using M-6 dataset of the MSL benchmark [8].

normal boundary is often tightly defined, which can result in slight deviations being classified as anomalies. Models that cannot discriminate between normal and anomaly classes may predict normal samples as anomalous, leading to high false positives. Fig. 1(a) shows an example of a histogram of anomaly scores of an existing unsupervised TSAD method called THOC [4] on a benchmark dataset (M-6 test set from the MSL dataset). Frequent false positives often occur when normal data are assigned high anomaly scores. In instances where the representation of a segment of the normal data closely resembles that of anomalous data, there can be an elevated anomaly score attributed to these normal samples, consequently leading to an increase in false positives. This issue can be seen on the right side of Fig. 1(a), where many normal samples are assigned high anomaly scores.

An alternative approach is to leverage self-supervised contrastive representation learning. Contrastive learning entails training a model to differentiate between pairs of similar and dissimilar samples in a dataset. Contrastive representation learning has been successful in image [5] and natural language processing [6], and its potential in TSAD has been explored in recent years. In the context of an anomaly detection task where the majority of the data is normal, contrastive loss functions can increase the distance between normal samples and their corresponding negative (anomalous) samples. Simultaneously, they decrease the distance between normal samples and their corresponding positive (normal) samples. By doing so, the distinction between normal and anomalous data is made clearer and more pronounced, which helps to identify a more accurate boundary for the normal samples. However, existing contrastive learning methods for TSAD mainly assume that augmented time series windows are positive samples and temporally distant windows are negative samples. We argue that these assumptions carry the risk that augmentation of time series can transform them to negative samples, and a temporally distant window can represent a positive sample, leading to ineffective anomaly detection performance. Fig. 1(b) shows anomaly scores of a contrastive learning TSAD method called TS2Vec [7]. As demonstrated, the anomaly scores of both normal and anomalous data are blended together, resulting in the inefficacy of anomaly detection. In fact, in situations where normal and anomalous data are intermingled within the representation space, achieving an enhanced anomaly detection rate – as measured by metrics such as the F1 score or the area under the precision–recall curve (AU-PR) – is accompanied by an increase in the false positive rate (FPR).

We propose a novel two-stage framework called CARLA, designed specifically to enhance time series anomaly detection. Our novel approach addresses the lack of labelled data through a contrastive approach, which leverages existing generic knowledge about different types of time series anomalies [9] in the first stage (pretext). We inject various types of anomalies, which facilitates the learning representations of normal behaviour. This is achieved by encouraging the learning of similar representations for windows that are temporally closed windows while ensuring dissimilar representations for windows and their corresponding injected anomalous windows.

Additionally, to ensure that the representation of existing real anomalous windows (for which we do not have labels) is different from normal representations, we employ a self-supervised approach to classify normal/anomalous representations of windows based on their nearest/furthest neighbours in the representation space in the second stage (self-supervised classification). By making the normal representations more discriminative, we enhance the anomaly detection performance in our proposed model. Specifically, the main contributions of this paper can be summarised as follows:

- We propose a novel contrastive representation learning model to detect anomalies in time series, which delivers top-tier outcomes across a range of real-world benchmark datasets, encompassing both univariate time series (UTS) and multivariate time series (MTS). Addressing the challenge of lack of labelled data, our model learns to effectively discriminate normal patterns from anomalous ones in the feature representation space (see Fig. 1(c)). CARLA's implementation is publicly available on Github: https://github.com/zamanzadeh/CARLA.
- We propose an effective contrastive method for TSAD to learn feature representations for a pretext task by leveraging existing generic knowledge about time series anomalies (see Fig. 5(a)).
- We propose a self-supervised classification method that leverages the representations learned in the pretext stage to classify time series windows. Our goal is to classify each sample by utilising its neighbours in the representation space learned during the pretext stage (see Fig. 5(b)).
- Our comprehensive analysis across seven real-world benchmark datasets reveals the superior performance of CARLA over a range of ten SOTA unsupervised, semi-supervised, and self-supervised contrastive learning models. CARLA's consistent balance between FPR and AU-PR throughout various MTS and UTS datasets underscores its precision and reliability. This balance ensures that CARLA provides reliable and precise alerts, which are crucial for many real-world applications.

2. Related work

In this section, we concentrate on three areas: deep learning methods in TSAD, unsupervised representation of time series, and the contrastive representation learning technique.

2.1. Time series anomaly detection

The detection of anomalies within time series data has been the subject of extensive research, using an array of techniques from statistical methods to classical machine learning and, more recently, deep learning models [3]. Established statistical techniques such as moving averages like the ARIMA model [10] have seen widespread application. Machine learning techniques, including clustering algorithms

and density-based approaches, alongside algorithms similar to decision trees [11] have also been leveraged.

Deep Learning methods in TSAD: In recent years, deep learning has proven highly effective due to its ability to autonomously extract features [2]. The focus within TSAD is largely on unsupervised [8], semi-supervised [12], and recently self-supervised [7] approaches, addressing the issue of scarce labelled data. Techniques such as OmniAnomaly [13] prove to be particularly useful in situations where anomaly labels are unavailable, while semi-supervised methods make efficient use of the labels that are available.

Deep learning techniques, encompassing autoencoders [14], variational autoencoders (VAEs) [12], RNNs [13], LSTM networks [8], GANs [15], and Transformers [16,17] have shown potential in TSAD, particularly for high-dimensional or non-linear data. The preference for deep models like LSTM-VAE [12], DITAN [18] and THOC [4] is because they excel at minimising forecasting errors while capturing time series data's temporal dependencies.

In summary, semi-supervised methods, such as LSTM-VAE [12] excel when labels are readily available. On the other hand, unsupervised methods, such as OmniAnomaly [13] and AnomalyTransformer [17], become more suitable when obtaining anomaly labels is a challenge. These unsupervised deep learning methods are preferred due to their capability to learn robust representations without requiring labelled data. The advent of self-supervised learning methods has further improved generalisation in unsupervised anomaly detection [19].

2.2. Unsupervised time series representation

The demonstration of substantial performance by unsupervised representation learning across a wide range of fields, such as computer vision [5,20], natural language processing [6], and speech recognition [21], has made it a desirable technique. In the realm of time series, methods like TKAE [22] and TST [23] have been suggested. While these methods have made significant contributions, some of them face challenges with scalability for exceptionally long time series or encounter difficulties in modelling complex time series. To overcome these limitations, methods such as TNC [24] and T-Loss [25] have been proposed. They leverage time-based negative sampling, triplet loss, and local smoothness of signals for the purpose of learning scalable MTS representations. Despite their merits, these methods often limit their universality by learning representations of particular semantic tiers, relying on significant assumptions regarding invariance during transformations. Recent studies, such as RoSAS [26], have outlier detection in data beyond time series by utilising triplet loss.

2.3. Contrastive representation learning

Contrastive representation learning creates an embedding space where similar samples are close and dissimilar ones are distant, applicable in domains like natural language processing, computer vision, and time series anomaly detection [27]. Traditional models like InfoNCE loss [28] and SimCLR [5] use positive–negative pairs to optimise this space, demonstrating strong performance and paving the way for advanced techniques. In TSAD, contrastive learning is crucial for recognising patterns. TS2Vec [7] uses this approach hierarchically for multi-level semantic representation, while DCdetector [29] introduces a dual attention asymmetric design with pure contrastive loss, enabling permutation invariant representations.

3. CARLA

Problem definition: Given a time series D which is partitioned into m overlapping time series windows $\{w_1, .., w_i, .., w_m\}$ with stride 1 where $w_i = \{x_1, \ldots, x_i, .., x_{WS}\}$, WS is time series window size, $x_i \in \mathbb{R}^{Dim}$ and Dim is the dimension of time series, the goal is to detect anomalies in time series windows.

CARLA (A Self-supervised **C**ontr**A**stive **R**epresentation **L**earning Approach for time series **A**nomaly detection) is built on several key components, each of which plays a critical role in achieving effective representation learning as illustrated in Fig. 2. CARLA consists of two main stages: the Pretext Stage and the Self-supervised Classification Stage.

Initially, in the Pretext Stage (Section 3.2), it employs anomaly injection to learn similar representations for temporally proximate windows and distinct representations for windows and their equivalent anomalous windows (Section 3.1). These injected anomalies include point anomalies, such as sudden spikes, and subsequence anomalies, like unexpected pattern shifts. This technique not only aids in training the model to recognise deviations from the norm but also strengthens its ability to generalise across various types of anomalies. At the end of the Pretext Stage, we establish a prior by finding the nearest and furthest neighbours for each window representation, setting the foundation for the next stage. The Self-supervised Classification Stage (Section 3.3) then classifies these window representations as normal or anomalous based on the proximity of their neighbours in the representation space (Section 3.4). This classification aims to group similar time series windows together while distinctly separating them from dissimilar ones. The effectiveness of this stage is pivotal in accurately categorising time series windows, reinforcing CARLA's capability to differentiate between normal and anomalous patterns. The comprehensive end-to-end pipeline of CARLA, including these stages, is illustrated in Fig. 2. The following sections present the ideas underlying our approach.

3.1. Anomaly injection

The technique of anomaly injection, a powerful data augmentation strategy for time series, facilitates the application of self-supervised learning in the absence of ground-truth labels. This technique has recently been applied in TSAD, notably in COUTA [30]. While the augmentation methods we employ are not designed to represent every conceivable anomaly type [17] – a goal that would be unattainable – they amalgamate various robust and generic heuristics to effectively identify prevalent out-of-distribution instances.

3.1.1. Anomaly injection steps

During the training phase, each window is manipulated by randomly choosing instances within a given window w_i. Two primary categories of anomaly injection models – point anomalies and subsequence anomalies – are adopted to inject anomalies to a time series window w_i. In a multivariate time series context, a random start time and a subset of dimension(s) d are selected for the injection of a point or subsequence anomalies. Note in the context of multivariate time series, anomalies are not always present across all dimensions, prompting us to randomly select a subset of dimensions for the induced anomalies ($d < \lceil Dim/10 \rceil$). The injected anomaly portion for each dimension varied from 1 data point to 90% of the window length.

This approach fosters the creation of a more diverse set of anomalies, enhancing our model's capability to detect anomalies that exist in multiple dimensions. It strengthens our model's effectiveness in discriminating between normal and anomalous representations in the Pretext Stage. It is important to underscore that the anomaly injection strategies employed in our model's evaluation were consistently applied across all benchmarks to ensure a fair and impartial comparison of the model's performance across various datasets. The step-by-step process for our anomaly injection approach is detailed in Algorithm 1, which encapsulates the methodology of point and subsequence anomaly injection, providing a clear understanding of our process and its implementation.

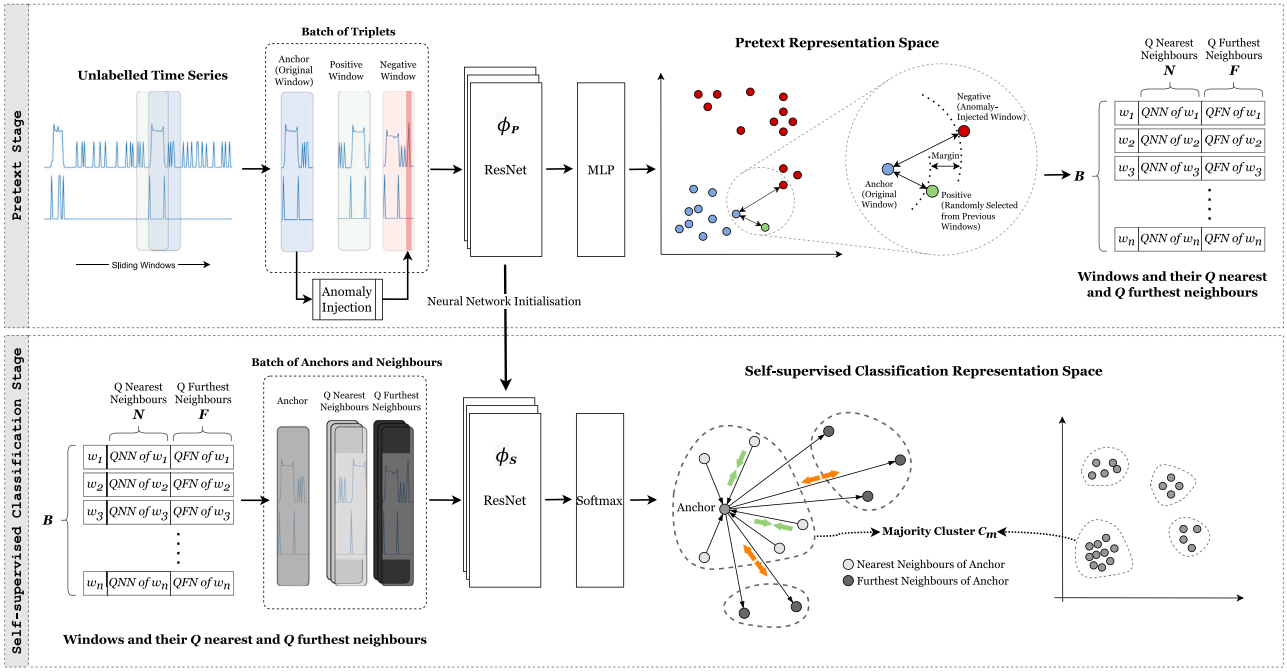

Fig. 2. The end-to-end pipeline of CARLA consists of two main stages: the Pretext Stage and the Self-supervised Classification Stage. In the Pretext Stage, anomaly injection techniques are used for self-supervised learning. The Self-supervised Classification Stage integrates the learned representations for a contrastive approach that maximises the similarity between anchors and their nearest neighbours while minimising the similarity between anchors and their furthest neighbours. The output is a trained model and the majority class, enabling inference for anomaly detection.

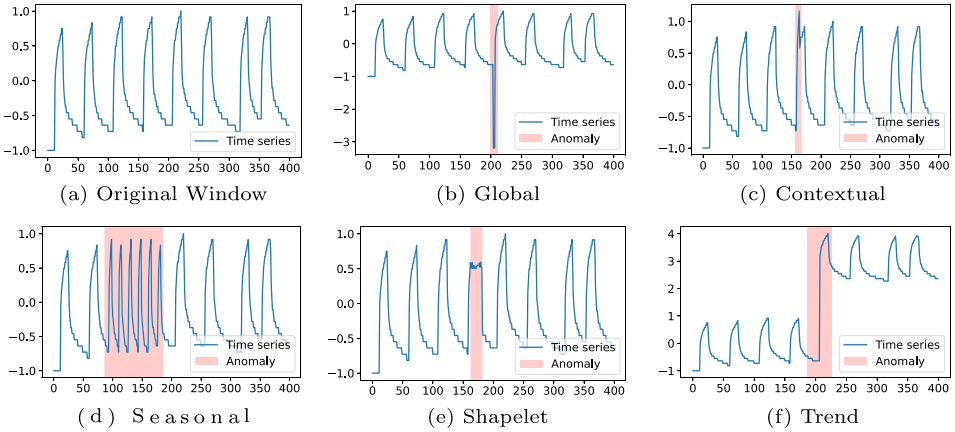

Fig. 3. Different types of synthetic anomaly injection used in CARLA. The figure presents the effect of synthetic anomaly injections into a randomly selected window of size 400 from the first dimension of E-2 in the MSL dataset [8]. The (a) represents the original time series window, while the remaining five demonstrate the same window but with different types of anomalies injected: (b) `Global`, (c) `Contextual`, (d) `Seasonal`, (e) `Shapelet`, and (f) `Trend`. Anomalous points or subsequences are accentuated in red.

3.1.2. Point anomalies

Our methodology incorporates the injection of single-point anomalies (a spike) at randomly chosen instances within a given window w_i. We employ two distinct types of point anomalies, namely, `Global` (Fig. 3(b)) and `Contextual` (Fig. 3(c)) [17].

3.1.3. Subsequence anomalies

The technique of subsequence anomaly injection, motivated by the successful implementation of the Outlier Exposure approach [31], enhances anomaly detection in time series data by generating contextual out-of-distribution examples. It involves the introduction of a subsequence anomaly within a window w_i, represented as $w(t|t \in [s, e])$, where s and e denotes the anomaly's start and end points. We explore three primary types of subsequence (a.k.a. pattern) anomalies: `Seasonal` (Fig. 3(d)), `Shapelet` (Fig. 3(e)), and `Trend` (Fig. 3(f)) [17].

3.2. Pretext stage

The Pretext Stage of CARLA consists of two parts: Part One is a contrastive representation learning using a ResNet architecture (which has shown effectiveness in times series classification task [32]) to learn representations for time series windows. Part Two is a post-processing that uses the learned representations from Part One to identify semantically meaningful nearest and furthest neighbours for each learned window representation. Algorithm 2 shows the steps.

3.2.1. Part one: Contrastive representation learning

In this part, we introduce a contrastive learning framework for learning a discriminative representation of features for time series windows. To extract features from the time series data, we utilise a multi-channel ResNet architecture, where each channel represents a

Algorithm 1 `InjectAnomaly`(w)

Input: Time series window w
Output: Anomaly injected time series window w'

$types \leftarrow$ {`Seasonal`, `Trend`, `Global`, `Contextual`, `Shapelet`}
$Dim \leftarrow$ random subset of dimensions of w (for UTS, $Dim = \{1\}$)
$w' \leftarrow$ copy of w
$s, e \leftarrow$ random start and end points from $(0, size(w)]$, where $e > s$
for each dimension d in Dim **do**
  $anomaly \leftarrow$ randomly choose anomaly type from $types$
  if $anomaly =$ `Global` **then**
    $\mu_w, \sigma_w \leftarrow$ mean and std of w in dimension d
    $g \leftarrow$ random number form $[3, 5]$ ▷ coefficient for `Global` anomaly
    $w'(s) =$ random value from $\{\mu_w + g.\sigma_w, \mu_w - g.\sigma_w\}$
  else if $anomaly =$ `Contextual` **then**
    $\mu_{se}, \sigma_{se} \leftarrow$ mean and std of subsequence $w(t|t \in [s, e])$ in dimension d
    $x \leftarrow$ random number form $[3, 5]$ ▷ coefficient for `Contextual` anomaly
    $w'(s) =$ random value from $\{\mu_{se} + x.\sigma_{se}, \mu_{se} - x.\sigma_{se}\}$
  else if $anomaly =$ `Seasonal` **then**
    $f \leftarrow$ random number from $\{\frac{1}{3}, \frac{1}{2}, 2, 3\}$ ▷ frequency coefficient for `Seasonal` anomaly
    $$w'(t) = \begin{cases} w(s + (\lfloor (t-s) \cdot f \rfloor \mod n)) & \text{if } s \le t < e \text{ and } f > 1 \\ w(s + \lfloor (t-s) \cdot f \rfloor) & \text{if } s \le t < e \text{ and } 0 < f < 1 \\ w(t) & \text{otherwise} \end{cases}$$
  else if $anomaly =$ `Trend` **then**
    $b \leftarrow$ random number form $[3, 5]$ ▷ coefficient for `Trend` anomaly
    $$w'(t) = \begin{cases} w(t) + b.\sigma_w & \text{if } s \le t \le e \\ w(t) & \text{if } t < s \text{ or } t > e \end{cases}$$
  else if $anomaly =$ `Shapelet` **then**
    $$w'(t) = \begin{cases} w(s) & \text{if } s \le t \le e \\ w(t) & \text{if } t < s \text{ or } t > e \end{cases}$$
  end if
end for
Return w'

Algorithm 2 `PretextCARLA`($\mathcal{D}$, Q, α)

Input: Sequential time series windows $\mathcal{D} = \{w_1, w_2, ..., w_m\}$, number of nearest/furthest neighbours Q, margin α.
Output: Trained model ϕ_p, all neighbours set $\mathcal{B}$, nearest neighbours set $\mathcal{N}$, furthest neighbours set $\mathcal{F}$.

$\mathcal{T} \leftarrow \emptyset$, $\mathcal{B} \leftarrow \emptyset$, $\mathcal{N} \leftarrow \emptyset$, $\mathcal{F} \leftarrow \emptyset$
for $i \leftarrow 1$ **to** $|D|$ **do**
  $a_i \leftarrow w_i$ ▷ anchor
  $p_i \leftarrow w_{i-r}$, where $r \sim \mathcal{U}(1, y)$ ▷ positive pair
  $n_i \leftarrow$ `InjectAnomaly`(w_i) ▷ negative pair (Alg. 1)
  append triplet (a_i, p_i, n_i) to $\mathcal{T}$ ▷ triplets batches
  add a_i and n_i to $\mathcal{B}$ ▷ neighbours set with size $2|D|$
end for
while Pretext loss $\mathcal{L}_{pretext}(\phi_p, \mathcal{T}, \alpha)$ decreases **do**
  Update ϕ_p with $\mathcal{L}_{pretext}$ ▷ i.e. Equation (1)
end while
for $j \leftarrow 1$ **to** $|\mathcal{B}|$ **do**
  $\mathcal{N}_j \leftarrow Q$ nearest neighbours of $w_j \in \mathcal{B}$ in $\phi_p(\mathcal{B})$ space
  $\mathcal{F}_j \leftarrow Q$ furthest neighbours of $w_j \in \mathcal{B}$ in $\phi_p(\mathcal{B})$ space
end for
Return ϕ_p, $\mathcal{B}$, $\mathcal{N}$, $\mathcal{F}$ ▷ inputs of the next stage

different time series dimension. Using different kernel sizes in ResNet allows us to capture features at various temporal scales, which is particularly important in analysing time series data and makes our model less sensitive to window size selection. We add an MLP layer as the final layer to produce a feature vector with lower dimensions.

To encourage the model to distinguish between different time series windows, we utilise a triplet loss function. Specifically, we create triplets of samples in the form of (a, p, n), where a is the anchor (i.e. the original window w_i), p is a positive sample (i.e. another random window w_{i-r}, selected from y previous windows, where $r \sim \mathcal{U}(1, y)$), and n is a negative sample (i.e. an anomaly injected version of the original window w_i). Assuming we have all the triplets of (a, p, n) in a set $\mathcal{T}$, the pretext triplet loss function is defined as follows:

$$\mathcal{L}_{Pretext}(\phi_p, \mathcal{T}, \alpha) = \frac{1}{|\mathcal{T}|} \sum_{(a,p,n) \in \mathcal{T}} \max(\|\phi_p(a) - \phi_p(p)\|_2^2 - \|\phi_p(a) - \phi_p(n)\|_2^2 + \alpha, 0) \quad (1)$$

Algorithm 3 `SelfSupevisedCARLA`(ϕ_p, $\mathcal{B}$, $\mathcal{N}$, $\mathcal{F}$, C, β)

Input: Initial trained neural network (ResNet) from the Pretext Stage ϕ_p, Dataset of time series windows including original windows and anomaly injected windows $\mathcal{B}$, Set of Q nearest neighbours for each window $\mathcal{N}$, Set of Q furthest neighbours for each window $\mathcal{F}$, Number of Classes C, Entropy loss weight β
Output: Trained model ϕ_s, Majority class C_m

$\phi_s \leftarrow$ initialise by ϕ_p ▷ ϕ_p from Algorithm 2 (`PretextCARLA`)
while $\mathcal{L}_{Self-supervised}(\phi_s, \mathcal{B}, \mathcal{N}, \mathcal{F}, C, \beta)$ decreases **do**
  Update ϕ_s with $\mathcal{L}_{Self-supervised}$ ▷ i.e. Equation (6)
end while
for $i \leftarrow 1$ **to** $|D|$ **do**
  $C^i = \arg\max(\phi_s(w_i)), w_i \in \mathcal{D}$ ▷ assign class label $C_j \in C$ to window i
end for
$C_m = \arg\max_{C_j \in C}(n(C_j))$ ▷ find majority class C_m
▷ $n(C_j)$ denotes the number of members in a class C_j
Return ϕ_s, C_m

Where $\phi_p(\cdot)$ is the learned feature representation neural network, $\|\cdot\|_2^2$ denotes the squared Euclidean distance, and α is a margin that controls the minimum distance between positive and negative samples. The objective of the pretext triplet loss function is to decrease the distance between the anchor and its corresponding positive sample while simultaneously increasing the distance between the anchor and negative samples. This approach encourages the model to learn a representation that can differentiate between normal and anomalous windows.

Our approach empowers the model to learn similar feature representations for temporally proximate windows. Since the majority of data is normal, the model captures temporal relationships of normal data through learning similar representations for normal windows that are temporally proximate. Furthermore, by introducing anomalies into the system, the model can learn a more effective decision boundary, resulting in a reduced FPR and enhanced precision in anomaly detection compared to current state-of-the-art models.

The output of Part One includes a trained neural network (ResNet) ϕ_p and a list of anchor and negative samples stored in $\mathcal{B}$.

3.2.2. Part two: Nearest and furthest neighbours

Part Two of our approach uses the feature representations learned in Part One to identify semantically meaningful nearest and furthest neighbours of each sample. To achieve this, we utilise $\mathcal{B}$ along with the indices of Q nearest and Q furthest neighbours of all samples in $\mathcal{B}$.

The primary goal of this part is to generate a prior that captures the *semantic similarity* and *semantic dissimilarity* between windows' representations using their neighbours, as our empirical analysis shows that in the majority of cases, these nearest neighbours belong to the same class (see Fig. 5(a)). Semantic similarity for a given window w_i defines as Q nearest neighbours of $\phi_p(w_i) \in \phi_p(\mathcal{B})$, where Q is the number of nearest neighbours. And, semantic dissimilarity for a given window w_i defined as Q furthest neighbours of $\phi_p(w_i) \in \phi_p(\mathcal{B})$. The output of Part Two is a set of all anchor and negative samples (anomaly injected) and the indices of their Q nearest neighbours $\mathcal{N}$ and Q furthest neighbours $\mathcal{F}$. Utilising $\mathcal{N}$ and $\mathcal{F}$ can enhance the performance of our classification method used in the Self-supervised Classification Stage. The culmination of the Pretext stage is a comprehensive set of windows alongside their nearest and furthest neighbours, setting the stage for their utilisation in the forthcoming Self-supervised Classification Stage.

3.3. Self-supervised classification stage

As we transition into the Self-supervised Classification Stage, we utilise the output of the Pretext stage as our foundational input. This stage includes initialising a new ResNet architecture with the learned feature representations from Part One of the Pretext Stage and then integrating the semantically meaningful nearest and furthest neighbours from Part Two as a prior ($\mathcal{N}$ and $\mathcal{F}$) into a learnable approach. The steps are detailed in Algorithm 3.

To encourage the model to produce both consistent and discriminative predictions, we employ a contrastive approach with a customised loss function. Specifically, the loss function maximises the similarity between each window representation and its nearest neighbours while minimising the similarity between each window representation and its furthest neighbours. The loss function can be defined as follows: At the beginning of this stage, we have $\mathcal{B}$ from the Pretext Stage, which is the set of all original window representations (we call them anchors in this stage) and their corresponding anomalous representations. Let C be the number of classes. We also have the Q nearest neighbours of the anchors $\mathcal{N}$ and the Q furthest neighbours of the anchors $\mathcal{F}$.

We aim to learn a classification neural network function ϕ_s – initialised by ϕ_p from the Pretext Stage – that classifies a window w_i and its Q nearest neighbours to the same class, and w_i and its Q furthest neighbours to different classes. The neural network ϕ_s terminates in a softmax function to perform a soft assignment over the classes $C = \{1, \dots, C\}$, with $\phi_s(w) \in [0, 1]^C$.

To encourage similarity between the anchors and their nearest neighbours, we compute the pairwise similarity between the probability distributions of the anchor and its neighbours, as shown in Eq. (2). The dot product between an anchor and its neighbour will be maximal when the output of the softmax for them is close to 1 or 0 and consistent in that it is assigned to the same class. Then, we define a consistency loss in Eq. (3) using the binary cross entropy to maximise the similarity between the anchor and nearest neighbours.

$$\text{similarity}(\phi_s, w_i, w_j) = \langle \phi_s(w_i) \cdot \phi_s(w_j) \rangle = \phi_s(w_i)^\top \phi_s(w_j) \tag{2}$$

$$\mathcal{L}_{consistency}(\phi_s, \mathcal{B}, \mathcal{N}) = -\frac{1}{|\mathcal{B}|} \sum_{w \in \mathcal{B}} \sum_{w_n \in \mathcal{N}_w} \log(\text{similarity}(\phi_s, w, w_n)) \tag{3}$$

The consistency loss aims to strengthen the alignment of anchors with their nearest neighbours, promoting cohesion by enhancing the similarity within these neighbours.

We define an inconsistency loss to encourage dissimilarity between the anchors and their furthest neighbours in Eq. (4). In this regard, we compute the pairwise similarity between the probability distributions of the anchor and furthest neighbour samples as well. Then, use the binary cross entropy loss to minimise the similarity to the furthest neighbours in the final loss function. While the mathematical form mirrors the consistency loss, its application diverges. Here, the similarity measure is used inversely; we seek to minimise this similarity, thereby driving a distinction between the anchor and its furthest neighbours to underscore class separation.

$$\mathcal{L}_{inconsistency}(\phi_s, \mathcal{B}, \mathcal{F}) = -\frac{1}{|\mathcal{B}|} \sum_{w \in \mathcal{B}} \sum_{w_n \in \mathcal{F}_w} \log(\text{similarity}(\phi_s, w, w_n)) \tag{4}$$

To encourage class diversity and prevent overfitting, we apply entropy loss on the distribution of anchor and neighbour samples across classes. Assuming the classes set is denoted as $C = \{1, \dots, C\}$, and the probability of window w_i being assigned to class c is denoted as $\phi_s^c(w_i)$:

$$\mathcal{L}_{entropy}(\phi_s, \mathcal{B}, C) = \sum_{c \in C} \hat{\phi}_s^c \log(\hat{\phi}_s^c) \quad \text{where } \hat{\phi}_s^c = \frac{1}{|\mathcal{B}|} \sum_{w_i \in \mathcal{B}} \phi_s^c(w_i) \tag{5}$$

The final objective in the Self-supervised Classification Stage is to minimise the total loss. This loss is calculated by the difference between the consistency and inconsistency losses, reduced by the entropy loss multiplied by a weight parameter, β:

$$\begin{aligned} &\mathcal{L}_{Self-supervised}(\phi_s, \mathcal{B}, \mathcal{N}, \mathcal{F}, C, \beta) = \\ &\mathcal{L}_{consistency}(\phi_s, \mathcal{B}, \mathcal{N}) - \mathcal{L}_{inconsistency}(\phi_s, \mathcal{B}, \mathcal{F}) - \beta \cdot \mathcal{L}_{entropy}(\phi_s, \mathcal{B}, C) \end{aligned} \tag{6}$$

The goal of the loss function is to learn a representation that is highly discriminative, with the nearest neighbours assigned to the same class as the anchors and the furthest neighbours assigned to a distinct class. By incorporating the semantically meaningful nearest and furthest neighbours, the model is able to produce more consistent and confident predictions, with the probability of a window being classified as one particular class is close to 1 or 0.

Overall, the loss function $\mathcal{L}_{Self-supervised}$ represents a critical component of our approach to time series anomaly detection, as it allows us to effectively learn a discriminative feature representation that can be utilised to distinguish between normal and anomalous windows.

Table 1
Statistics of the benchmark datasets used.

Benchmark	# datasets	# dims	Train size	Test size	Anomaly%
MSL	27	55	58,317	73,729	10.72%
SMAP	55	25	140,825	444,035	13.13%
SMD	28	38	708,405	708,420	4.16%
SWaT	1	51	496,800	449,919	12.33%
WADI	1	123	784,568	172,801	5.77%
Yahoo-A1	67	1	46,667	46,717	1.76%
KPI	29	1	1,048,576	2,918,847	1.87%

3.4. CARLA's inference

Upon completing Part Two of our approach, we determine the class assignments for set $\mathcal{D}$ and majority class C_m, where $C_m = \arg\max_{C_j \in C}(n(C_j))$ which comprises the class with the highest number of anchors (i.e. original windows). For every new window w_t during inference, we calculate $\phi_s^{C_m}(w_t)$, representing the probability of window w_t being assigned to the majority class C_m. Using an end-to-end approach, we classify a given window w_t as normal or anomalous based on whether it belongs to the majority class. Specifically, Eq. (7) describes how we infer a label for a time series window w_t. Additionally, we can employ Eq. (8) to generate an anomaly score of w_t for further analysis.

$$\text{Anomaly label } (w_t) : \begin{cases} 0, & \text{if } \forall c \in C, \phi_s^{C_m}(w_t) \geq \phi_s^c(w_t) \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

$$\text{Anomaly score } (w_t) : 1 - \phi_s^{C_m}(w_t) \tag{8}$$

By assigning a window to a specific class, our model can determine the likelihood of a given window being normal or anomalous with greater precision.

Furthermore, the probability $\phi_s^{C_m}(w_t)$ can be used to calculate an anomaly score, with lower values indicating a higher probability of anomalous behaviour. This score can be useful for further analysis and can aid in identifying specific characteristics of anomalous behaviour.

4. Experiments

The objective of this section is to thoroughly assess CARLA's performance through experiments conducted on multiple benchmark datasets and to compare its results with alternative methods. Section 4.1 provides an overview of the benchmark datasets used in the evaluation, highlighting their significance in assessing our model's effectiveness. Section 4.2 delves into the benchmark methods employed for comparing the performance of different models. In Section 4.3, we discuss the evaluation setup, including the hyper-parameters chosen for our approach. Moving forward, Section 4.4 provides results for all benchmark methods based on the respective datasets, facilitating a comprehensive comparison. Additionally, we explore the behaviour of CARLA across diverse data configurations and variations in the ablation studies (Section 4.5) and investigate its sensitivity to parameter changes in Section 4.6. All evaluations were conducted on a system equipped with an A40 GPU, 13 CPUs, and 250 GB of RAM.

4.1. Benchmark datasets

We evaluate the performance of the proposed model and make comparisons of the results across the seven most commonly used real benchmark datasets for TSAD. Table 1 summarises key statistics for each dataset. All datasets, except Yahoo, have a predefined train/test split with unlabelled training data.

NASA Datasets – Mars Science Laboratory (MSL) and **Soil Moisture Active Passive (SMAP)**[1] – [8] are collected from NASA spacecraft, contain anomaly information from incident reports for a spacecraft monitoring system.

Server Machine Dataset (SMD)[2] [13] is gathered from 28 servers over 10 days, with normal data observed for the first 5 days and anomalies sporadically injected in the last 5 days.

Secure water treatment (SWaT)[3] [33] is data from a water treatment platform with 51 sensors over 11 days, including 41 anomalies deliberately generated over the last 4 days using a wide range of attacks.

Water distribution testbed (WADI)[4] [34] is data from a scaled-down urban water distribution system that included a combined total of 123 actuators and sensors. It covers a span of 16 days, including anomalies in the final two days.

Yahoo[5] dataset [35] Consists of hourly 367 sampled time series with labels. We focus on the A1 benchmark, which includes "real" production traffic data from Yahoo properties in 67 univariate time series.

Key performance indicators (KPI)[6] contains data of on service and machine key performance indicators from real Internet company scenarios, including response time, page views, CPU, and memory utilisation.

4.2. Benchmark methods

We provide a description of the ten prominent TSAD models that were used for comparison with CARLA, spanning various TSAD categories. For the semi-supervised approach, the LSTM-VAE [12] is highlighted, which is a point-wise anomaly detection model. In the realm of unsupervised reconstruction-based models, Donut [36] stands out for UTS, while OmniAnomaly [13] and AnomalyTransformer [17] are noted for MTS, alongside TranAD [16] which caters to both UTS and MTS. The unsupervised forecasting-based model, THOC [4] and TimesNet [37], are recognised for their predictive capabilities. Within the self-supervised category, MTAD-GAT [19] emerges as a hybrid model, with TS2Vec [7] and DCdetector [29] offering representation-based methodologies. Recent advancements are represented by TimesNet [37] and DCdetector [29], showcasing the ongoing evolution in TSAD models. The "Random Anomaly Score" model is designed to generate anomaly scores based on a normal distribution ($\mathcal{N}(0,1)$) with mean 0 and standard deviation 1. This model is specifically developed to illustrate evaluation metrics and strategies in the field of TSAD.

[1] https://www.kaggle.com/datasets/patrickfleith/nasa-anomaly-detection-dataset-smap-msl.

[2] https://github.com/NetManAIOps/OmniAnomaly/tree/master/ServerMachineDataset.

[3] https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/.

[4] https://itrust.sutd.edu.sg/testbeds/water-distribution-wadi/.

[5] https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70.

[6] https://github.com/NetManAIOps/KPI-Anomaly-Detection.

4.3. Evaluation setup

In our study, we evaluate ten TSAD models on benchmark datasets previously mentioned in Section 4.1 using their best hyper-parameters, as they stated, to ensure a fair evaluation. The default hyperparameters for our implementation are as follows: The CARLA model consists of a 3-layer ResNet architecture with three different kernel sizes [8, 5, 3] to capture temporal dependencies, with a representation dimension of 128. We use the same hyper-parameters across all datasets: window size = 200, number of classes = 10, number of nearest/furthest neighbours (Q) = 5, and coefficient of entropy = 5. For detailed information about all experiments involving the aforementioned hyper-parameter choices, please refer to Section 4.5. We run the Pretext Stage for 30 epochs and the Self-supervised Classification Stage for 100 epochs on all datasets.

It is important to note that we do not use Point Adjustment (PA) in our evaluation process. Despite its popularity, [38] found that applying PA leads to an overestimation of TSAD models' capability and can bias results towards methods that produce extreme anomaly scores. To ensure accuracy, we present conventional F1 scores and relegate PA results to CARLA's Github repository. The F1 score without PA is referred to as F1.

4.4. Benchmark comparison

The performance metrics employed to evaluate the effectiveness of all models consist of precision, recall, traditional F1 score (F1 without PA), FPR and AU-PR. Additionally, we computed the average ranks of the models based on their F1. For all methods, we used the precision–recall curve on the anomaly score for each time series in the datasets to find the best F1 score based on precision and recall for the target time series.

Since certain benchmark datasets such as MSL contain multiple time series datasets (as shown in Table 1), we cannot merge or combine these time series due to the absence of timestamp information. Additionally, calculating the F1 score for the entire dataset by averaging individual scores is not appropriate. In terms of precision and recall, the F1 score represents the harmonic mean, which makes it a non-additive metric. To address this, we get the confusion matrix for each time series, i.e., we calculate the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each dataset of a benchmark such as MSL. Then, we sum up the TP, FP, TN, and FN values from all the confusion matrices to get an overall confusion matrix of the entire benchmark dataset. After that, we use the overall confusion matrix to calculate the overall precision, recall, and F1 score. This way, we ensure that the F1 score is calculated correctly for the entire dataset rather than being skewed by averaging individual F1 scores.

Furthermore, for datasets with more than one time series, we report the average and standard deviation of AU-PR across all time series. AU-PR is advantageous in imbalanced datasets because it is less sensitive to the distribution of classes [39]. It considers the trade-off between precision and recall across all possible decision thresholds, making it more robust in scenarios where the number of instances in classes is imbalanced. By focusing on the positive class (i.e. anomalous) and its predictions, AU-PR offers a more precise evaluation of a model's ability to identify and prioritise the minority class correctly.

Tables 2 and 3 show the performance comparison between CARLA and all benchmark methods for multivariate and univariate datasets, respectively. These two tables show our model outperforms other models with higher F1 and AU-PR across all datasets (except SWaT for F1 and AU-PR and Yahoo-A1 for AU-PR), in which CARLA is the third best. This shows the strength of CARLA in generalising normal patterns due to its high precision. CARLA's precision is the highest across all seven datasets. At the same time, CARLA has high enough recall, and as a result, its F1 is the highest across all datasets except SWaT. While the recall is higher in other benchmarks, such as OmniAnomaly, it is

Table 2
Precision (Prec), recall (Rec), F1 and AU-PR results for various models on multivariate time series datasets. The best results are in bold, and the second-best is indicated by underline. Due to the single time series in both the SWaT and WADI datasets, the standard deviation of AU-PR is not available.

Model	Metric	MSL	SMAP	SMD	SWaT	WADI	Rank
LSTM-VAE [12] (2018)	Prec	0.2723	0.2965	0.2045	0.9707	0.0596	5.5
	Rec	0.8083	0.8303	0.5491	0.5769	1.0000	
	F1	0.4074	0.4370	0.2980	0.7237	0.1126	
	AU-PR	0.285 ± 0.249	0.258 ± 0.305	0.395 ± 0.257	0.685	0.039	
OmniAnom [13] (2019)	Prec	0.1404	0.1967	0.3067	0.9068	0.1315	5.0
	Rec	0.9085	0.9424	0.9126	0.6582	0.8675	
	F1	0.2432	0.3255	0.4591	**0.7628**	0.2284	
	AU-PR	0.149 ± 0.182	0.115 ± 0.129	0.365 ± 0.202	**0.713**	0.120	
MTAD-GAT [19] (2020)	Prec	0.3559	0.3783	0.2473	0.1387	0.0706	4.2
	Rec	0.7067	0.8239	0.5834	0.9585	0.5838	
	F1	0.4734	0.5186	0.3473	0.2423	0.1259	
	AU-PR	0.335 ± 0.259	0.339 ± 0.300	0.401 ± 0.263	0.095	0.084	
THOC [4] (2020)	Prec	0.1936	0.2039	0.0997	0.5453	0.1017	6.8
	Rec	0.7718	0.8294	0.5307	0.7688	0.3507	
	F1	0.3095	0.3273	0.1679	0.6380	0.1577	
	AU-PR	0.239 ± 0.273	0.195 ± 0.262	0.107 ± 0.126	0.537	0.103	
AnomTran [17] (2021)	Prec	0.2182	0.2669	0.2060	0.9711	0.0601	5.4
	Rec	0.8231	0.8600	0.5822	0.5946	0.9604	
	F1	0.3449	0.4074	0.3043	0.7376	0.1130	
	AU-PR	0.236 ± 0.237	0.264 ± 0.315	0.273 ± 0.232	0.681	0.040	
TranAD [16] (2022)	Prec	0.2957	0.3365	0.2649	0.1927	0.0597	6.2
	Rec	0.7763	0.7881	0.5661	0.7965	1.0000	
	F1	0.4283	0.4716	0.3609	0.3103	0.1126	
	AU-PR	0.278 ± 0.239	0.287 ± 0.300	0.412 ± 0.260	0.192	0.039	
TS2Vec [7] (2022)	Prec	0.1832	0.2350	0.1033	0.1535	0.0653	7.7
	Rec	0.8176	0.8826	0.5295	0.8742	0.7126	
	F1	0.2993	0.3712	0.1728	0.2611	0.1196	
	AU-PR	0.132 ± 0.135	0.148 ± 0.165	0.113 ± 0.075	0.136	0.057	
DCdetector [29] (2023)	Prec	0.1288	0.1606	0.0432	0.1214	0.1417	9.1
	Rec	0.9578	0.9619	0.9967	0.9999	0.9684	
	F1	0.2270	0.2753	0.0828	0.2166	0.2472	
	AU-PR	0.129 ± 0.144	0.124 ± 0.153	0.043 ± 0.036	0.126	0.121	
TimesNet [37] (2023)	Prec	0.2257	0.2587	0.2450	0.1214	0.1334	6.1
	Rec	0.8623	0.8994	0.5474	1.0000	0.1565	
	F1	0.3578	0.4019	0.3385	0.2166	0.1440	
	AU-PR	0.283 ± 0.213	0.208 ± 0.211	0.385 ± 0.225	0.083	0.084	
Random	Prec	0.1746	0.1801	0.0952	0.1290	0.0662	8.4
	Rec	0.9220	0.9508	0.9591	0.9997	0.9287	
	F1	0.2936	0.3028	0.1731	0.2166	0.1237	
	AU-PR	0.172 ± 0.133	0.140 ± 0.148	0.089 ± 0.058	0.129	0.067	
CARLA	Prec	0.3891	0.3944	0.4276	0.9886	0.1850	1.6
	Rec	0.7959	0.8040	0.6362	0.5673	0.7316	
	F1	**0.5227**	**0.5292**	**0.5114**	0.7209	**0.2953**	
	AU-PR	**0.501 ± 0.267**	**0.448 ± 0.326**	**0.507 ± 0.195**	0.681	**0.126**	

with the expense of very low precision (with median precision <19.67% across all datasets). This means that a good balance between precision and recall is achieved by CARLA. This is also shown in our consistently high AU-PR compared to others and indicates that it is proficient in accurately identifying anomalous instances with higher recall while minimising false alarms with higher precision on imbalanced datasets where normal instances are predominant. Finally, CARLA achieved the lowest average rank (best rank), based on F1, in both multivariate and univariate benchmarks (see last column in Tables 2 and 3). Critical difference diagrams have been added to CARLA's GitHub, illustrating its statistically significant difference compared to others.

We now evaluate the performance of all models in terms of FPR. We also incorporate AU-PR in this comparison, as some models can achieve a low FPR at the expense of lowering AU-PR. In the scatter plots provided in Fig. 4, we show FPR on the x-axis and AU-PR on the y-axis. The ideal point on this scatter plot is in the leftmost top corner. CARLA consistently shows a lower FPR compared to other models across various multivariate time series datasets. This is particularly noteworthy in the context of anomaly detection, where the cost of false positives can be substantial, leading to wasted resources and potential overlook of true anomalies. CARLA's position in the plots is close to the leftmost top corner (in the optimal quadrant). This indicates fewer false alarms while maintaining competitive AU-PR scores, which measure the precision and recall balance of anomaly detection.

For instance, on the MSL dataset, CARLA is among the models with the lowest FPR, and its AU-PR score is in the upper tier, showcasing its ability to identify true anomalies accurately. On the SMAP and SMD datasets, CARLA maintains a significantly lower FPR than most models, suggesting its robustness to varying data conditions. In the SWaT dataset, CARLA excels with the lowest FPR and the third highest AU-PR, marking it suitable for precision-critical applications. Even on WADI, while CARLA's FPR is slightly higher, it remains in the optimal quadrant, reflecting a strong FPR and AU-PR balance vital for operational continuity and effective anomaly detection.

Adding to CARLA's impressive performance on multivariate datasets, its capabilities in UTS analysis, as shown in Fig. 4, further highlighting its robustness. On the Yahoo-A1 dataset, while CARLA does not achieve the highest AU-PR, it still demonstrates a commendable balance of a low FPR and a high AU-PR, ranking as the third-best model. Its place in the upper left quadrant indicates a strong ability to correctly identify anomalies with a minimal rate of false alarms. On the KPI dataset, CARLA maintains a competitive edge with a low FPR that

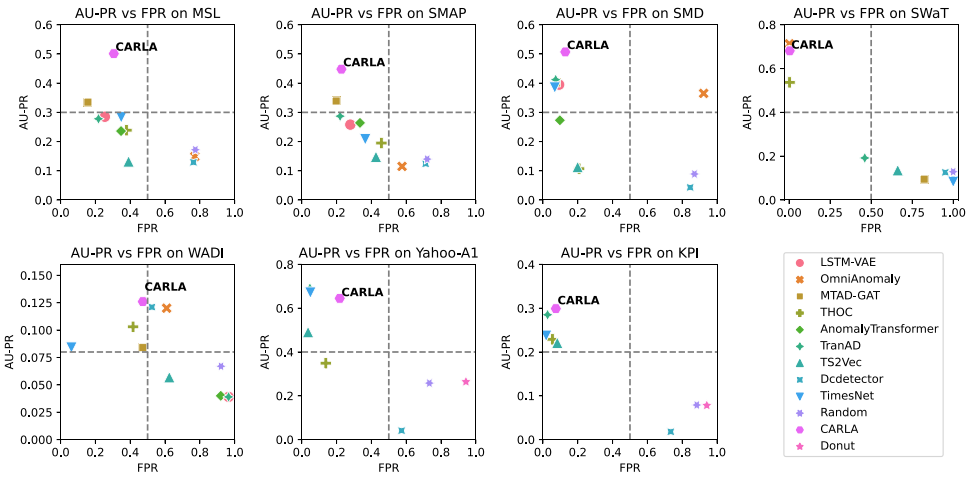

Fig. 4. Models' performance comparison on MTS & UTS datasets regarding their FPR and AU-PR. Models in the optimal quadrant (low FPR, high AU-PR) have superior anomaly detection with minimal false alarms.

Table 3
Precision (Prec), recall (Rec), F1, and AU-PR results for various models on univariate time series datasets. The best results are in bold, and the second-best is indicated by underline.

Model	Metric	Yahoo-A1	KPI	Rank
Donut [36] (2018)	Prec	0.3239	0.0675	5.0
	Rec	0.9955	0.9340	
	F1	0.4888	0.1259	
	AU-PR	0.264 ± 0.244	0.078 ± 0.073	
THOC [4] (2020)	Prec	0.1495	0.1511	5.5
	Rec	0.8326	0.5116	
	F1	0.2534	0.2334	
	AU-PR	0.349 ± 0.342	0.229 ± 0.217	
TranAD [16] (2022)	Prec	0.4185	0.2235	2.0
	Rec	0.8712	0.4016	
	F1	0.5654	0.2872	
	AU-PR	**0.691 ± 0.324**	0.285 ± 0.206	
TS2Vec [7] (2022)	Prec	0.3929	0.1333	5.0
	Rec	0.6305	0.4329	
	F1	0.4841	0.2038	
	AU-PR	0.491 ± 0.352	0.221 ± 0.156	
DCdetector [29] (2023)	Prec	0.0598	0.0218	8.0
	Rec	0.9434	0.8589	
	F1	0.1124	0.0425	
	AU-PR	0.041 ± 0.059	0.018 ± 0.017	
TimesNet [37] (2023)	Prec	0.3808	0.2174	3.0
	Rec	0.7883	0.2713	
	F1	0.5135	0.2414	
	AU-PR	0.671 ± 0.307	0.237 ± 0.148	
Random	Prec	0.2991	0.0657	6.5
	Rec	0.9636	0.9488	
	F1	0.4565	0.1229	
	AU-PR	0.258 ± 0.184	0.079 ± 0.064	
CARLA	Prec	0.5747	0.1950	1.0
	Rec	0.9755	0.7360	
	F1	**0.7233**	**0.3083**	
	AU-PR	0.645 ± 0.352	**0.299 ± 0.245**	

surpasses most other models. While the AU-PR is the highest, CARLA's ability to limit false positives is a notable strength.

Overall, the consistent performance of CARLA across multivariate and univariate datasets and its proficiency in dealing with a variety of data types, as shown by its low FPR, highlights its strength as a reliable model for anomaly detection in MTS and UTS data. Its ability to minimise false positives without significantly sacrificing true positive detection makes it an excellent choice for scenarios where high-confidence alerts are vital.

4.5. Ablation study

For a deeper understanding of the contributions of different stages and components in our proposed model, we conduct an ablation study. Our analysis focused on: (i) **Effectiveness of CARLA's two stages**, (ii) **Positive pair selection strategy** (iii) **Effectiveness of Different Anomaly Types** (iv) **Effectiveness of the loss components**.

4.5.1. Effectiveness of carla's stages

In this section, we delve into an in-depth evaluation of the stages of our proposed model, utilising the M-6 time series from the MSL dataset. Our primary objective is to develop a comprehensive grasp of the patterns inherent in the features extracted and learned by our model throughout its stages. For this purpose, we employ t-distributed Stochastic Neighbour Embedding (t-SNE) to visualise the output of the Pretext Stage and the Self-supervised Classification stage, shown in Figs. 5(a) and 5(b), respectively. From these graphical representations, it is palpable that the second stage significantly enhances the discrimination between normal and anomalous samples. It is important to highlight that the first stage reveals some semblance of anomalous samples to normal ones, which might be due to the existence of anomalies close to normal boundaries. However, the Self-supervised Classification stage efficiently counteracts this ambiguity by effectively segregating these instances, thereby simplifying subsequent classification tasks.

Further, we assess the efficacy of the Self-supervised Classification Stage in refining the representation generated by the Pretext Stage. This was accomplished by juxtaposing the anomaly scores derived from the output of each stage. In Fig. 5(c), the distribution of anomaly scores at the Pretext Stage is displayed, computed using the Euclidean distances of the test samples relative to the original time series windows in the training set. Conversely, Fig. 5(d) depicts the subsequent alteration in the distribution of anomaly scores after applying the Self-supervised Classification Stage.

In the Pretext Stage, the anomaly score is computed as the minimum distance between the test sample and all original training samples in the representation space, aiming to identify the closest match among the training samples for the given test sample. A smaller distance implies a greater probability that the test sample belongs to the normal data distribution.

In the Self-supervised Classification Stage, we use the inference step in Section 3.3 and the anomaly label is computed as detailed in Eq. (8).

As we can observe from Figs. 5(a) and 5(b), the self-supervised classification Stage has resulted in a significant improvement in the separation of normal and anomalous windows. The distribution of anomaly scores in Fig. 5(d) is more clearly separated than in Fig. 5(c).

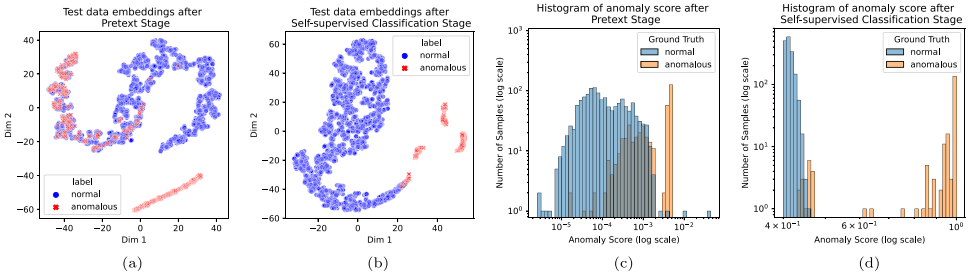

Fig. 5. Comparative analysis of the model's stages utilising t-SNE and anomaly score distributions on M-6 dataset in MSL. (a) t-SNE after the Pretext stage. (b) t-SNE after the Self-supervised Classification stage. (c) Anomaly score distribution in the Pretext Stage. (d) The distribution of anomaly scores after applying the Self-supervised Classification Stage.

Table 4
Positive pair selection results on MSL dataset.

Pos pair selection	Prec	Rec	F1	AU-PR
CARLA-noise	0.3086	0.7935	0.4433	0.3983
CARLA-temporal	0.3891	0.7959	**0.5227**	**0.5009**

Table 5
Effectiveness of different anomaly types on MSL dataset.

Anomaly type	Prec	Rec	F1	AU-PR
All types	0.3891	0.7959	**0.5227**	**0.5009**
w/o trend	0.3185	0.7849	0.4513	0.4972
w/o contextual	0.3842	0.7935	0.5177	0.4683
w/o shapelet	0.3564	0.8370	0.4999	0.4328
w/o global	0.3383	0.8404	0.4824	0.4340
w/o seasonal	0.2400	0.8428	0.3737	0.3541

This indicates that the second stage has produced more consistent representations. Furthermore, the anomaly scores in Fig. 5(d) are relatively closer to 0 or 1, indicating improved discrimination between normal and anomalous windows in the second stage of CARLA.

4.5.2. Positive pair selection

To evaluate the effectiveness of the positive pair selection method in CARLA, we compared two approaches in training: random temporal neighbour from y temporally closest window samples and weak augmentation with noise (add normal noise with sigma 0.01 to a window). We used MSL benchmark and used identical configuration and hyper-parameters for both approaches and evaluated their performance shown in Table 4. Our experiment demonstrates that selecting positive pairs using a random temporal neighbour is more effective than weak augmentation with noise. Since anomalies occur rarely, choosing a positive pair within the target window's temporal proximity is a better strategy.

4.5.3. Effectiveness of different anomaly types

CARLA's performance is evaluated on the MSL dataset by systematically removing different types of anomalies during the Pretext Stage. The evaluation metrics used are precision, recall, F1 score, and AU-PR. The results are sorted based on F1 from the least significant to the most significant types of anomalies in Table 5. This result represents CARLA's overall performance when using all types of anomalies during the anomaly injection process. It serves as a baseline for comparison with the subsequent results.

In the experiments, anomalies were injected into the training data on a per-window basis. For each window in the multivariate time series, a random number of dimensions was selected, ranging from 1 to $\lceil Dim/10 \rceil$ of the total dimensions. These selected dimensions were injected with anomalies, starting from the same point across all the chosen dimensions. The injected anomaly portion for each dimension varied from 1 data point to 90% of the window length. This approach ensured a diverse and controlled injection of anomalies within the training data (For more detail, see Algorithm 1)

Table 6
Effectiveness of $\mathcal{L}_{inconsistency}$ and $\mathcal{L}_{entropy}$ on MSL dataset.

$\mathcal{L}_{inconsistency}$	$\mathcal{L}_{entropy}$	Prec	Rec	F1	AU-PR
✗	✗	0.3092	0.7155	0.4318	0.3984
✗	✓	0.3453	0.8112	0.4846	0.4424
✓	✗	0.3219	0.7534	0.4511	0.4175
✓	✓	0.3891	0.7959	**0.5227**	**0.5009**

Eliminating trend anomalies has a significant negative impact on CARLA's precision. However, the recall remains relatively high, indicating that trend anomalies are important for maintaining a higher precision level. Similar to the previous case, removing contextual anomalies leads to a slight decrease in precision but an improvement in recall. The exclusion of shapelet anomalies leads to a moderate decrease in precision, while the recall remains relatively high. This suggests that shapelet anomalies contribute to the model's precision but are not as crucial for capturing anomalies in general. By removing global anomalies, CARLA's precision slightly decreases, indicating that it becomes less accurate in identifying true anomalies. However, the recall improves, suggesting that the model becomes more sensitive in detecting anomalies overall. Removing seasonal anomalies significantly drops precision, though the high recall indicates effective anomaly detection without relying on seasonal patterns. Overall, the analysis suggests that global and seasonal anomalies significantly influence the CARLA model's performance, while contextual, shapelet, and trend anomalies have varying impacts. These findings provide insights into the model's sensitivity to different anomaly types and can guide further improvements in anomaly injection strategies within CARLA's framework.

4.5.4. Effectiveness of loss components

To assess the effectiveness of two loss components, namely $\mathcal{L}_{inconsistency}$ and $\mathcal{L}_{entropy}$, in the total loss function for self-supervised classification on the MSL dataset, we conduct experiments using identical architecture and hyper-parameters. The evaluation metrics employed to measure the performance are precision, recall, F1, and AU-PR, as presented in Table 6.

In cases where only one loss component is used, the model's performance is relatively lower across all metrics, indicating that without incorporating these loss components, the model struggles to detect anomalies in the MSL dataset effectively. Where both $\mathcal{L}_{inconsistency}$ and $\mathcal{L}_{entropy}$ loss components are included in the total loss function, the model achieves the best performance among all scenarios, surpassing the other combinations. The precision, recall, F1 score, and AU-PR are the highest in this case, indicating that the combination of both loss components significantly improves the model's anomaly detection capability.

4.6. Parameter sensitivity

We examine the sensitivity of CARLA's parameters, investigating and analysing the effect of the **window size**, **number of classes**, **number of NN/FN**, and **entropy coefficient**.

Table 7
Exploring the effect of window size on MSL, SMD, and Yahoo datasets.

WS	MSL				SMD				Yahoo			
	Prec	Rec	F1	AUPR	Prec	Rec	F1	AUPR	Prec	Rec	F1	AUPR
50	0.2755	0.7702	0.4058	0.3173	0.4569	0.4885	0.4722	0.4636	0.3046	0.8800	0.4526	0.5533
100	0.3233	0.7666	0.4548	0.4155	0.2584	0.5938	0.3598	0.4136	0.3709	0.9713	0.5368	0.5331
150	0.3606	0.7823	0.4936	0.4595	0.3599	0.6592	0.4655	0.4661	0.5720	0.9578	0.7162	0.6685
200	0.3891	0.7959	0.5227	0.5009	0.4276	0.6362	0.5114	0.5070	0.5747	0.9755	0.7233	0.6450
250	0.4380	0.8037	0.5412	0.5188	0.4224	0.6596	0.5150	0.4850	0.7129	0.9781	0.8247	0.7164

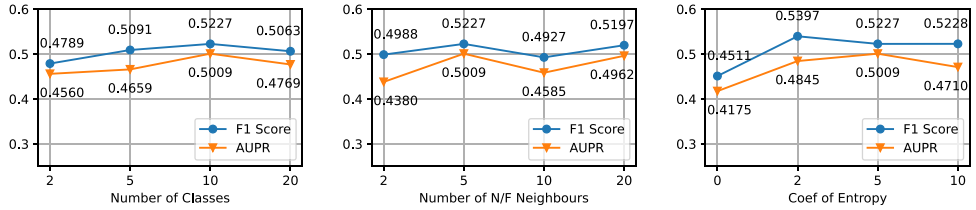

Fig. 6. F1 Score and AU-PR for (a) different number of classes, (b) different numbers of neighbours in $\mathcal{N}/\mathcal{F}$ and (c) different coefficient of entropy, on MSL dataset.

4.6.1. Effect of window size

As a hyper-parameter in time series analysis, window size holds considerable importance. Table 7 presents the results of exploring the impact of window size on three datasets: MSL, SMD, and Yahoo.

Based on the analysis of both the F1 and AU-PR, it can be concluded that window size 200 consistently outperforms other sizes on the MSL, SMD, and Yahoo datasets overall. This window size strikes a balance between precision and recall, effectively capturing anomalies while maintaining a high discrimination ability. Therefore, window size 200 is selected for all experiments.

4.6.2. Effect of number of classes

The provided results showcase the performance of the model with varying numbers of classes on the MSL dataset. Based on the illustration of both the F1 score and AU-PR in Fig. 6(a), the model performs relatively well across different numbers of classes. While CARLA's performance is pretty stable across all number of classes denoted on the x-axis of the plot in Fig. 6(a), it can be concluded that using 10 classes yields the highest performance. It strikes a balance between capturing anomalies and minimising false positives while effectively discriminating between anomalies and normal samples.

The results suggest that increasing the number of classes beyond 10 does not significantly improve the model's performance. When the number of classes increases, normal representations divide into the different classes and, in CARLA, are detected as anomalies (lower probability of belonging to the major class). However, using 2 classes leads to a lower performance compared to 10 classes, implying that a more fine-grained classification with 10 classes provides the assumption that various anomalies are spread in the representation space.

4.6.3. Effect of number of neighbours in $\mathcal{N}$ and $\mathcal{F}$

The provided results in Fig. 6(b) display the evaluation metrics for different numbers of nearest neighbours ($\mathcal{N}$) and furthest neighbours ($\mathcal{F}$) noted as Q.

Based on the analysis of both the F1 score and AU-PR, we can see that while CARLA's performance is pretty stable across the different numbers of neighbours in $\mathcal{N}$ and $\mathcal{F}$ denoted on the x-axis of the plot in Fig. 6(b), it can be concluded that employing 5 nearest/furthest neighbours outperforms the other options. This number of parameters strikes a favourable balance between precision and recall, allowing for effective anomaly detection while maintaining a high discrimination ability. Therefore, selecting 5 neighbours is deemed the optimal choice, as it consistently achieves higher F1 and AU-PR.

4.6.4. Effect of entropy coefficient

We explore the impact of the entropy coefficient in this experiment. Fig. 6(c) shows the results on MSL dataset for entropy coefficients of 0, 2, 5, and 10.

Based on the analysis of both the F1 score and AU-PR, a coefficient value of 5 emerges as the most effective choice for the entropy component in the loss function. It achieves the highest scores for both metrics, indicating better precision–recall balance and accurate ranking of anomalies. However, it is worth noting that coefficients of 2 and 10 also demonstrate competitive performance, slightly lower than the value of 5. The coefficient of 0 significantly reduces the model's performance, further emphasising the importance of incorporating the entropy component for anomaly detection performance. Overall, the analysis indicates that coefficients of 2, 5, and 10 are viable for the entropy component in the loss function, with 5 being optimal due to its superior F1 and AU-PR.

5. Conclusion

Our innovative end-to-end self-supervised framework, CARLA, utilises contrastive representation learning with anomaly injection to generate robust representations for time series and classify anomalous windows. The use of semantically meaningful nearest and furthest neighbours as a prior allows us to capture underlying patterns in time series and learn a representation that is well-aligned with these patterns, thereby enhancing detection accuracy. Our extensive experimental evaluation of the seven most commonly used time series benchmarks, encompassing 208 datasets with diverse real-world anomalies, shows promising results in detecting anomalies, demonstrating the effectiveness of CARLA in this domain.

This research has demonstrated the potential of contrastive learning combined with synthetic anomaly injection to address the limitations of the lack of labelled data in TSAD. To enhance our model's performance, future research will focus on refining positive and negative sample selection strategies. Moreover, we believe there is much scope to overcome the limitations of the anomaly injection component by broadening the forms of injected anomalies used in our framework. We further hypothesise that varying the severity of injected anomalies may improve the model's ability to detect anomalies in representation space. We commend the investigation of these intriguing prospects in future research.

Additionally, considering that some datasets have a significant number of abnormal samples, resulting in more anomalous windows during the Pretext Stage, it becomes critical to revisit the impact of selecting

positives and negatives for abnormal anchors in unsupervised settings. This reconsideration is crucial for enhancing the model's effectiveness in TSAD. Our results, based on appropriate metrics, contribute significantly to further investigations and advancements in TSAD, particularly on datasets like WADI and KPI. By employing the correct evaluation metrics, such as the conventional F1 score without point adjustment and AU-PR, this research provides a solid foundation for assessing the performance of future models in this field. The findings presented herein enable researchers to benchmark their models and compare their results against established standards. Reliable evaluation metrics and validated results serve as valuable resources, driving innovation in TSAD research.

CRediT authorship contribution statement

Zahra Zamanzadeh Darban: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Conceptualization. **Geoffrey I. Webb:** Writing – review & editing, Methodology, Conceptualization. **Shirui Pan:** Writing – review & editing, Methodology, Conceptualization. **Charu C. Aggarwal:** Writing – review & editing. **Mahsa Salehi:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Links to codes and data have been shared in the manuscript.

References

[1] R. Domingues, M. Filippone, P. Michiardi, J. Zouaoui, A comparative evaluation of outlier detection algorithms: Experiments and analyses, Pattern Recogn. 74 (2018) 406–421.

[2] G. Pang, C. Shen, L. Cao, A.V.D. Hengel, Deep learning for anomaly detection: A review, CSUR 54 (2) (2021) 1–38.

[3] S. Schmidl, P. Wenig, T. Papenbrock, Anomaly detection in time series: a comprehensive evaluation, VLDB 15 (9) (2022) 1779–1797.

[4] L. Shen, Z. Li, J. Kwok, Timeseries anomaly detection using temporal hierarchical one-class network, NeurIPS 33 (2020) 13016–13026.

[5] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: ICML, PMLR, 2020, pp. 1597–1607.

[6] L. Logeswaran, H. Lee, An efficient framework for learning sentence representations, in: ICLR, 2018.

[7] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, B. Xu, Ts2vec: Towards universal representation of time series, in: AAAI, Vol. 36, 2022, pp. 8980–8987.

[8] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, in: KDD, 2018, pp. 387–395.

[9] Z.Z. Darban, G.I. Webb, S. Pan, C.C. Aggarwal, M. Salehi, Deep learning for time series anomaly detection: A survey, 2022, arXiv preprint arXiv:2211.05244.

[10] G.E. Box, D.A. Pierce, Distribution of residual autocorrelations in autoregressive-integrated moving average time series models, J. Amer. Statist. Assoc. 65 (332) (1970) 1509–1526.

[11] K.M.T. Fei Tony Liu, Z.-H. Zhou, Isolation forest, in: ICDM, IEEE, 2008, pp. 413–422.

[12] D. Park, Y. Hoshi, C.C. Kemp, A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder, IEEE Robot. Autom. Lett. 3 (3) (2018) 1544–1551.

[13] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in: KDD, 2019, pp. 2828–2837.

[14] Y. Yao, J. Ma, Y. Ye, Regularizing autoencoders with wavelet transform for sequence anomaly detection, Pattern Recognit. 134 (2023) 109084.

[15] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, S.-K. Ng, MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks, in: ICANN: Text and Time Series, Springer, 2019, pp. 703–716.

[16] S. Tuli, G. Casale, N.R. Jennings, TranAD: Deep transformer networks for anomaly detection in multivariate time series data, Proc. VLDB Endow. 15 (2022) 1201–1214.

[17] J. Xu, H. Wu, J. Wang, M. Long, Anomaly transformer: Time series anomaly detection with association discrepancy, in: ICLR, 2021.

[18] M. Giannoulis, A. Harris, V. Barra, DITAN: A deep-learning domain agnostic framework for detection and interpretation of temporally-based multivariate anomalies, Pattern Recognit. 143 (2023) 109814.

[19] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, Q. Zhang, Multivariate time-series anomaly detection via graph attention network, in: ICDM, IEEE, 2020, pp. 841–850.

[20] T. Milbich, O. Ghori, F. Diego, B. Ommer, Unsupervised representation learning by discovering reliable image relations, Pattern Recognit. 102 (2020) 107107.

[21] Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, M. Auli, Self-training and pre-training are complementary for speech recognition, in: ICASSP, IEEE, 2021, pp. 3030–3034.

[22] F.M. Bianchi, L. Livi, K.Ø. Mikalsen, M. Kampffmeyer, R. Jenssen, Learning representations of multivariate time series with missing data, Pattern Recognit. 96 (2019) 106973.

[23] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, C. Eickhoff, A transformer-based framework for multivariate time series representation learning, in: KDD, 2021, pp. 2114–2124.

[24] S. Tonekaboni, D. Eytan, A. Goldenberg, Unsupervised representation learning for time series with temporal neighborhood coding, in: ICLR, 2021.

[25] J.-Y. Franceschi, A. Dieuleveut, M. Jaggi, Unsupervised scalable representation learning for multivariate time series, in: NeurIPS, Vol. 32, Curran Associates, Inc., 2019.

[26] H. Xu, Y. Wang, G. Pang, S. Jian, N. Liu, Y. Wang, RoSAS: Deep semi-supervised anomaly detection with contamination-resilient continuous supervision, IP&M 60 (5) (2023) 103459.

[27] P.H. Le-Khac, G. Healy, A.F. Smeaton, Contrastive representation learning: A framework and review, IEEE Access 8 (2020) 193907–193934.

[28] A.v.d. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, 2018, arXiv preprint arXiv:1807.03748.

[29] Y. Yang, C. Zhang, T. Zhou, Q. Wen, L. Sun, Dcdetector: Dual attention contrastive representation learning for time series anomaly detection, in: KDD, 2023.

[30] H. Xu, Y. Wang, S. Jian, Q. Liao, Y. Wang, G. Pang, Calibrated one-class classification for unsupervised time series anomaly detection, TKDE (2024).

[31] D. Hendrycks, M. Mazeika, T. Dietterich, Deep anomaly detection with outlier exposure, in: ICLR, 2019.

[32] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, DMKD 33 (4) (2019) 917–963.

[33] A.P. Mathur, N.O. Tippenhauer, SWaT: A water treatment testbed for research and training on ICS security, in: CySWater, IEEE, 2016, pp. 31–36.

[34] C.M. Ahmed, V.R. Palleti, A.P. Mathur, WADI: a water distribution testbed for research in the design of secure cyber physical systems, in: CySWater, 2017, pp. 25–28.

[35] N. Laptev, Y. B., S. Amizadeh, A benchmark dataset for time series anomaly detection, 2015, URL: https://yahooresearch.tumblr.com/post/114590420346/a-benchmark-dataset-for-time-series-anomaly.

[36] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, et al., Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications, in: WWW, 2018, pp. 187–196.

[37] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, TimesNet: Temporal 2D-variation modeling for general time series analysis, in: ICLR, 2023.

[38] S. Kim, K. Choi, H.-S. Choi, B. Lee, S. Yoon, Towards a rigorous evaluation of time-series anomaly detection, in: AAAI, Vol. 36, 2022, pp. 7194–7201.

[39] T. Saito, M. Rehmsmeier, The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets, PLoS One 10 (3) (2015) e0118432.

Zahra Zamanzadeh Darban is a Ph.D. researcher at the Department of Data Science and AI at Monash University, Australia. She focuses on anomaly detection in time series using deep learning. Supervised by Dr. Salehi, Professor Pan, and Professor Webb, she previously spent seven years as a software engineer, machine learning engineer, and system analyst in the industry before joining Monash.

Geoffrey I. Webb is the research director at Monash University Data Futures Institute, Australia. Former editor-in-chief of DMKD (2005–2014), he chaired ACM SIGKDD and IEEE ICDM conferences. Advisor to BigML Inc and FROOMLE, he innovated in association discovery and rule search. Awards include IEEE Fellow and the 2017 Eureka Prize in Data Science.

Shirui Pan received a Ph.D. in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia. He is a Professor with the School of Information and Communication Technology, Griffith University, Australia.

Prior to this, he was a Senior Lecturer with the Faculty of IT at Monash University.

His research interests include data mining and machine learning. To date, Dr Pan has published extensively in top-tier journals and conferences, including TPAMI, TKDE, TNNLS, ICML, NeurIPS, and KDD. His research received the 2024 CIS IEEE TNNLS Outstanding Paper Award and the 2020 IEEE ICDM Best Student Paper Award. He is recognised as one of the AI 2000 AAAI/IJCAI Most Influential Scholars in Australia. He is an ARC Future Fellow and a Fellow of Queensland Academy of Arts and Sciences (FQA).

Charu C. Aggarwal (Fellow, IEEE) received the BS degree from IIT Kanpur, in 1993, and the Ph.D. degree from the Massachusetts Institute of Technology, in 1996. He is a research scientist with the IBM T.J. Watson Research Center in York-town Heights, New York. He has since worked in the field of performance analysis, databases, and data mining. He has served on the program committees of most major database/data mining conferences, and served as program vice-chairs of SDM 2007, ICDM 2007, WWW 2009, and ICDM 2009. He served as an associate editor of the IEEE Transactions on Knowledge and Data Engineering (TKDE) from 2004 to 2008. He is an associate editor of the ACM Transactions on Knowledge Discovery from Data (TKDD), an action editor of the Data Mining and Knowledge Discovery, an associate editor of SIGKDD Explorations, and an associate editor of Knowledge and Information Systems. He is a fellow of the ACM.

Mahsa Salehi received the Ph.D. degree in computer science from the University of Melbourne, Australia in 2016. She then joined IBM Research, as a postdoctoral researcher. In 2017 she joined Monash University, Faculty of IT where she is currently a Senior Lecturer. Her research includes time series analytics and anomaly detection. She serves as an associate editor in Transactions on Knowledge Discovery from Data.