Applied Intelligence
https://doi.org/10.1007/s10489-021-02559-0

Unified model for collective and point anomaly detection using stacked temporal convolution networks

Zehui Li[1] · Zhijie Xiang[1] · Weijia Gong[1] · Hong Wang[1,2]

Accepted: 21 May 2021

Abstract
Time-series anomaly detection utilizing deep learning methods is widely used in fraud detection, network intrusion detection, and medical anomaly detection. Most deep learning methods exclusively focus on models based on recurrent neural networks (RNNs), such as long short-term memory (LSTM) or gated recurring units (GRUs), rather than on models based on convolutional neural networks (CNNs) or integrated ones. Inspired by the success of CNN-based models in many scenarios, we propose a single model that can be used for detecting both collective and point anomalies using stacked temporal convolution networks (CPA-TCN). Compared with state-of-the-art models, the CPA-TCN model boasts the following advantages. First, the CPA-TCN model reconstructs sequential features with current inputs and historical features and is only trained on normal datasets. Second, the CPA-TCN model outperforms RNN-based models in terms of speed and accuracy across diverse tasks and datasets and demonstrates more effective memory. Third, the CPA-TCN model can effectively detect both collective and point anomalies by detecting point anomalies before collective anomalies, overcoming the shortcomings of models that can either detect point or collective anomalies. Fourth, the two-part anomaly detection module can significantly improve the accuracy of point anomaly detection. Extensive experiments on real-world datasets demonstrate that our CPA-TCN model achieves better prediction results with the ROC-AUC of 98%–99% compared to state-of-the-art methods and thus has a competitive advantage.

Keywords Time-series · Anomaly detection · Collective anomaly · Point anomaly · Temporal convolution networks

1 Introduction

Anomaly detection helps identify anomalous data, namely data that considerably differ from a dataset's normal behavior. It is widely used in fraud detection, network intrusion detection, and medical anomaly detection. According to the anomalous data types, anomaly detection is classified into three categories: point, contextual, and collective anomaly detection. [1].

Collective and point anomalies typically occur in time series. Although several classical anomaly detection methods such as Isolation Forest [2], One-Class Support Vector Machine [3], and Boltzmann machines [4] are used to detect these anomalies, they are not sufficiently effective as they exclusively focus on current data rather than historical data. Thus, these methods fail to consider the temporal nature of anomalous data. Recent studies have reported certain deep learning models [5–8] for detecting time-series anomalies that can significantly render anomaly detection more effective.

Herein, we propose a model based on stacking temporal convolution networks (CPA-TCN). Compared with the abovementioned models, our model demonstrates several improvements.

First, our CPA-TCN model is a "unified" model. This means that it can be used for detecting both collective and

✉ Hong Wang
111052@sdnu.edu.cn

Zehui Li
1309729828@qq.com

Zhijie Xiang
1353753283@qq.com

Weijia Gong
2792207485@qq.com

1 School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China

2 Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Shandong Normal University, Jinan, China

point anomalies. Specifically, a parameter named timestep is introduced to make this model "unified".

Second, our CPA-TCN model outperforms RNN-based models across diverse tasks and datasets, particularly in terms of memory. Because, we improve the efficiency of our model using one-dimensional convolution networks with only the forward information rather than the backward information, which prevents the information from the future leaking into the past.

Third, the model first detects the point anomaly with $timestep = 1$, and then detects the collective anomaly with $timestep = n$, overcoming shortcomings of single-function models and improving the accuracy of anomaly detection.

Finally, we conduct experiments on different datasets to assess the performance of our CPA-TCN model. Experimental results show that our method has better prediction accuracy than state-of-the-art methods and is thus more competitive.

The rest of the paper is organized as follows. In Section 2, we briefly describe the related work activities. Section 3 introduces the framework of the CPA-TCN model, definitions, and algorithms related to our model. Experimental datasets and results are presented in Section 4. Finally, the paper concludes with highlights and future directions.

2 Related work

Anomaly detection is a popular research subject that can benefit a wide range of fields, including recommendation systems, time-series analysis [9], path planning and social network analysis [10]. Although there are several classical anomaly detection methods [2, 3],they are not very effective, as they exclusively focus on current data values rather than historical values, thereby neglecting the temporal nature of anomalous data. To address this issue, most researchers currently focus on RNN-based models, such as LSTMs or GRUs [11–14]. Similar to the point anomaly detection methods, most collective anomaly methods focus on LSTMs, proving highly beneficial in this field [15, 16]. We will review only certain relevant works. For more studies on the field, please refer to the literature [17].

Compared with the classical methods, deep learning methods represent time series more effectively and eliminate the need for tedious feature engineering. Bontemps presents a collective anomaly detection model based on LSTMs [16]. He only selects the KDD 1999 dataset ("https://www.unb.ca/cic/datasets/nsl.html") for experiments. He first trains the model on normal time-series data and calculates a threshold for each timestep. Finally, he detects a collective anomaly when the prediction error in a timestep exceeds the threshold value.

Zheng Fengming [18] presents an anomaly detection model based on an encoder-decoder framework with recurrent neural networks (RNN). In the model, he reconstructs the input time series and uses reconstruction errors to detect anomalies. Both Manhattan and edit distances are used to estimate the difference between an input time series and its reconstructed one. Results demonstrate that the encoder-decoder framework can successfully detect anomalies with a precision higher than 95%. Inspired by the success of this framework, we treat time-series reconstruction as a fundamental principle of anomaly detection.

Bai systematically evaluates CNN-based and RNN-based architectures for sequence modeling [19] with various standard tasks, such as the adding problem and memory copying. Sequence modeling is currently the benchmark for recurrent deep learning methods. His results indicate that convolution networks outperform canonical recurrent networks such as LSTMs across different datasets and tasks and demonstrate longer and more effective memory. Bai concludes that CNN networks should be regarded as a natural starting point and a powerful toolkit for sequence modeling.

Owing to their advantages, CNNs are used in many anomaly detection models. All models proved the superiority of CNN in anomaly detection. Based on the deep convolutional neural network (CBR-CNN) architecture, Chouhan [7] proposes a novel channel boosted and residual learning method to detect network intrusions. The methodology is based on the inherent nature of anomaly detection that employs one-class classification to detect network intrusion. Zhang [20] proposes an online transaction fraud detection model based on a convolution neural network, which constructs a sequence input layer that reorganizes raw transaction features to form different convolutional patterns. The significance of this model is that different feature combinations will produce different derivative features when they enter the convolution kernel. In [21], Fu proposes a CNN-based fraud detection framework to capture the intrinsic patterns of fraud behaviors learned from labeled data. A feature matrix represents abundant transaction data. A convolutional neural network is also applied to identify a set of latent patterns for each sample. In [22], researchers use deep learning techniques to effectively detect fraudsters in mobile communications. Different experiments are performed to assess the performance of their proposed model. They found that deep convolution neural networks (DCNN) outperformed other traditional machine learning algorithms, such as support vector machines, random forest, and gradient boosting classifier in terms of accuracy (82%) and training duration.

In summary, although the methods reported in previous works have achieved good results, many challenges remain to be resolved. For instance, RNN-based methods are relatively complex compared with CNN-based methods and cannot maintain a long effective history. Furthermore, CNN-based models [20] that can either detect point or collective anomalies cannot deal with complex data. To address these challenges, we constructed the innovative CPA-TCN model for detecting time-series anomalies. Our CPA-TCN can effectively detect both collective and point anomalies and combines all advantages of CNN-based models such as simplicity and efficiency.

3 CPA-TCN model

In this section, we will present our CPA-TCN model in detail. First, we provide related definitions. Then, we present the framework of the model. Finally, we will describe the model composition and explain how the model detects both point and collective anomalies.

3.1 Definition

For the sake of clarity, we first present the definitions which are associated with the CPA-TCN model.

– Definition 1. Point Anomaly (PA)

A data value is considered a point anomaly if it considerably differs from the majority of the data points.

– Definition 2. Collective Anomaly (CA)

A collective anomaly refers to a group of data points that differ from the majority of the data, wherein a single data point is not treated as an anomaly.

– Definition 3. Timestep

Timestep refers to the minimum time interval that the CPA-TCN needs to detect anomalies. It determines the type of anomaly detection based on the model. The timestep is defined as in (1).

$$timestep = \begin{cases} = 1, & for\ PA\ detection \\ > 1, & for\ CA\ detection \end{cases} \tag{1}$$

A critical aspect of anomaly detection methods is the criteria used to identify the anomalies. Typically, the outputs of anomaly detection methods are either anomaly scores or binary labels. We adopt the anomaly scores in our method as they disclose more information than binary labels. The anomaly score describes the deviation level of each data point from the mean. The data instances are ranked according to the anomaly score. A domain-specific threshold, commonly known as the decision score, is used to calculate the anomaly score for every point.

– Definition 4. Anomaly Score (AS)

Anomaly score, also known as reconstruction error, is defined as the absolute error between the input values x and the reconstructed values y, as shown by (2).

$$AS(x, y) = |x - y| \tag{2}$$

– Definition 5. Total Anomaly Score (TAS)

The total anomaly score refers to the total score over a timestep, as shown in (3).

$$TAS = \sum_{i=1}^{timestep} AS_i \tag{3}$$

– Definition 6. Threshold

The threshold indicates whether an anomaly occurs within a timestep. A TAS value above the threshold indicates an anomaly. The domain-specific threshold is often selected by subject matter experts to identify anomalous data.

3.2 Model framework

Now, we present our novel framework for the model and perform anomaly detection tasks.

Specifically, we first extract dataset features and then feed these features into the TCNs to reconstruct the input sequence. Subsequently, we detect point anomaly first before collective anomaly over each timestep. The model structure is shown in Fig. 1.

3.3 Temporal convolution networks module

Inspired by the work conducted by Bai [19], we leverage one-dimensional convolution networks for anomaly detection, which can retain historical information and capture local semantic relationships. The temporal convolution network comprises a causal convolution layer, dilated convolution layers, and a residual block, as shown in Fig. 2.

The main characteristics of our TCNs architecture are as follows.

(1) Convolution in the architectures is causal, which means there is no information leakage from the future to the past.
(2) The model produces an output of the same length as the input.
(3) Combining deep networks (augment with residual layers) and dilated convolutions, our TCNs can deeply examine the past to make predictions.

– Causal Convolution

As mentioned above, causal convolutions can achieve the first point of the main characteristics. The architecture of the causal convolution layer incorporates the one-dimensional convolutional layer. In causal convolution layers, the value

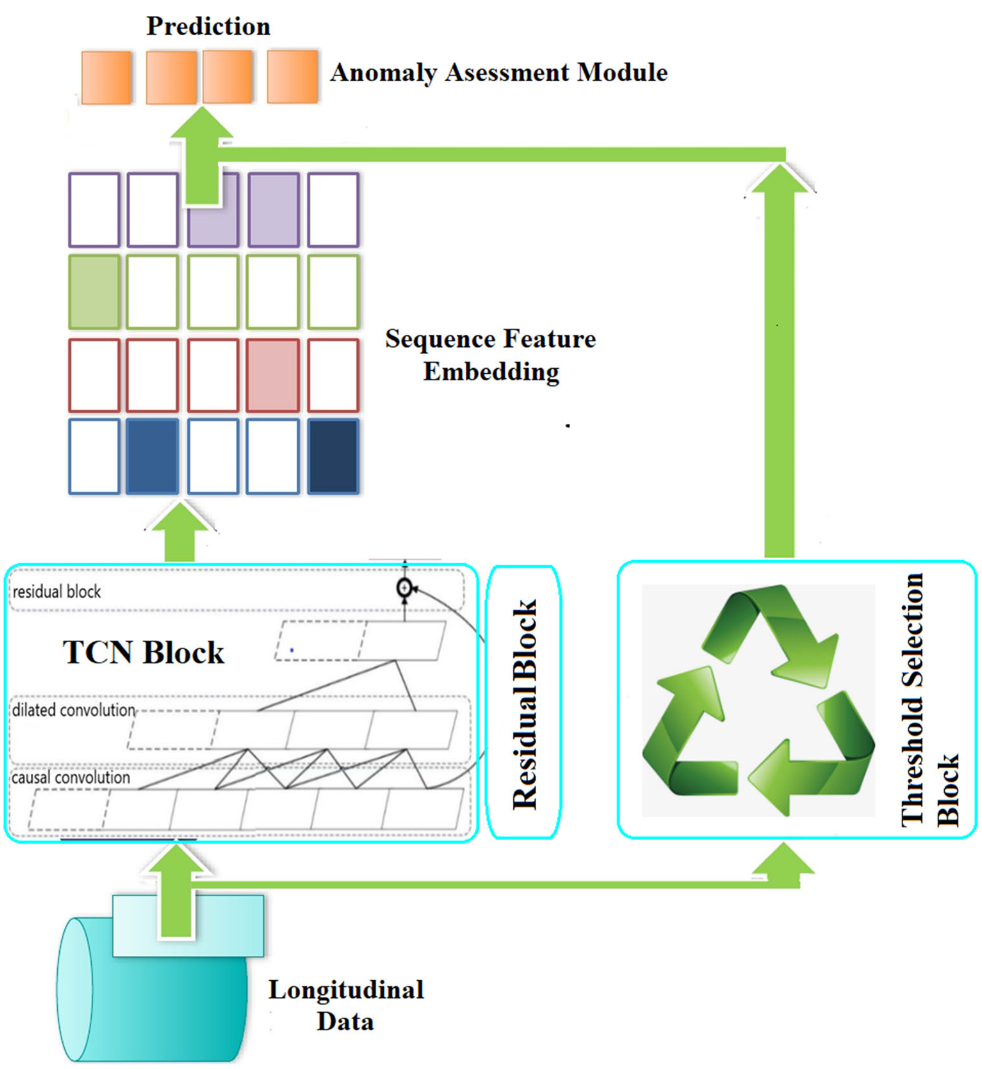


Fig. 1 Framework of the CPA-TCN model

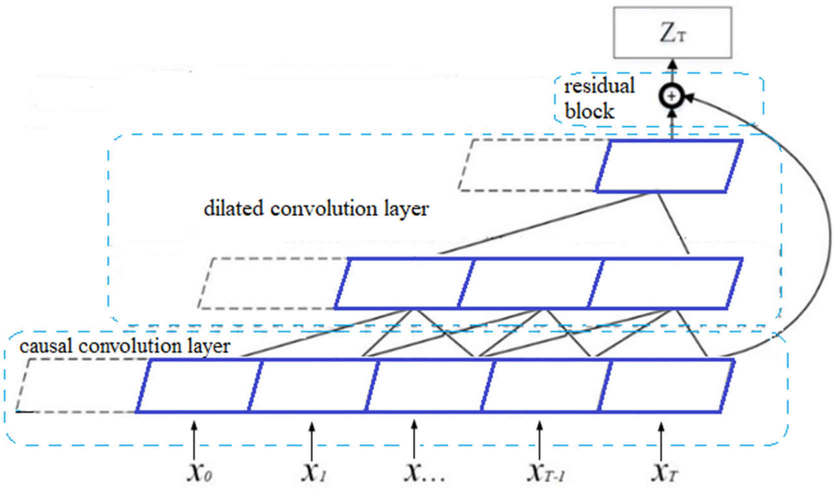


Fig. 2 Temporal Convolution Networks

at time t depends only on values from time t and previous layers. However, a simple causal convolution can only review a linear history in the network's depth, which makes it challenging to apply the causal convolution to long sequence tasks. Dilated convolutions are proposed to solve this problem.

– Dilated Convolutions

By increasing the dilation factor d and filter size k, dilated convolutions enable an output at the top layer to represent extensive inputs and allow probing extensive valid histories. An example is shown in Fig. 3.

Figure 3 shows that the expansion rate of the first layer is 1, and the size of the convolution kernel is 3. The first hidden layer has a dilation factor of 2, and the size of the convolution kernel is 3. It also shows that the second hidden layer has a dilation factor of 4, and the convolution kernel size is 3. Notably, no inputs forward in the sequence contribute to the next layer's node, which makes no information from the future to be learned by the model. Furthermore, the output layer is initially zero to ensure the output length is the same as the input.

In practice, we can stack multiple hidden layers to make the causal dilated convolution networks deeper, as the model is required to remember varying levels of sequence history to make predictions. Otherwise, if the model does accept as much history in the old task, it would cause issues, and the model would demonstrate poor performance.

To further clarify dilated convolution, we provide the following definition. Suppose that the input sequence is $x_0, \ldots, x_T$, the output sequence is $y_1 \ldots, y_T$ and a filter $f : \{0, \ldots, k-1\} \rightarrow R_n$, the dilated convolution operation $F(s)$ on element s of the sequence is defined as (4), where d is the dilation factor,k is the filter size, s is the element in the sequence, x is the input code, d is the expansion rate, and $s - d \cdot i$ is the direction of the previous data.

$$F(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d\cdot i} \tag{4}$$

As shown in formula 4, when handling a large amount of time-series data and the long historical connection distance, we construct multiple dilation convolution layers based on the causal convolution to make holes injected into the ordinary convolution and further expand the receptive field of the model. Increases in the expansion rate and the receptive field range indicate that the convolution output is related to a long history.

– Residual Connections

As mentioned in the above section, stacking a more causal dilated convolution layer helps extract multiple levels of features, making the neural network deeper. However, the deeper network structure often faces gradient disappearance or gradient explosion. One way to solve this problem is to introduce more reference information for the network. Therefore, the residual connector is integrated into the causal dilated convolution layer to improve the generalization ability of the convolutional neural network. The residual connector structure is shown in Fig. 4.

Suppose the module's input is X and the causal convolution module output is $H(X)$. Simultaneously, we set a 2-layer causal dilated convolution network and a nonlinear activation function. To improve the generalization ability, we normalize the convolution kernel's weight [23] and set the dropout [24] layer in the residual module. The output of the entire residual module is expressed in formula 5, Here, $H(X)$ represents the output of the causal

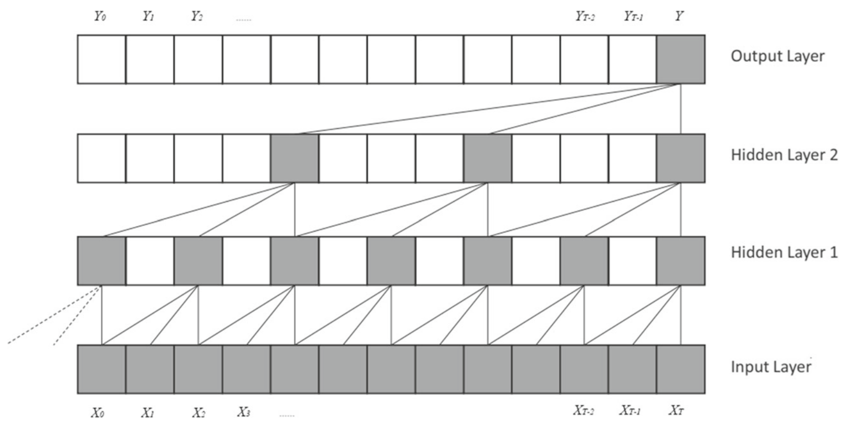


Fig. 3 A dilated causal convolution layer

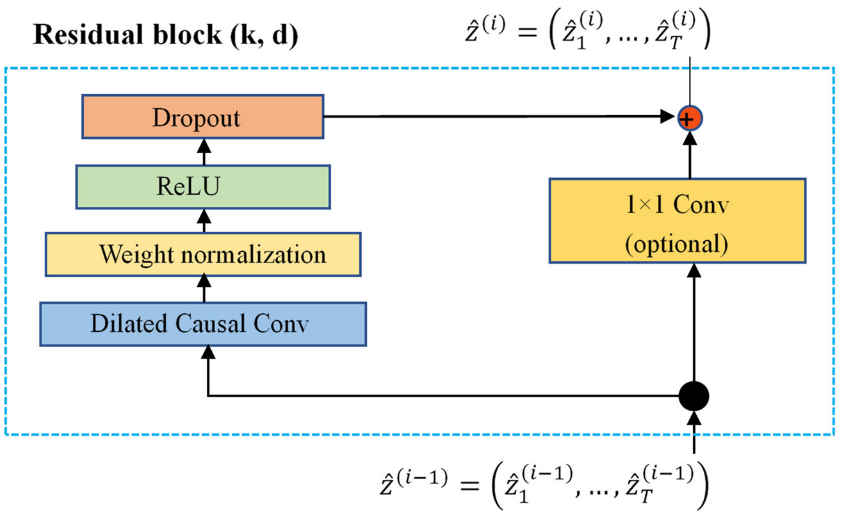

Residual block (k, d)
$\hat{z}^{(i)} = (\hat{z}_1^{(i)}, \ldots, \hat{z}_T^{(i)})$
Dropout
ReLU
Weight normalization
Dilated Causal Conv
1×1 Conv (optional)
$\hat{z}^{(i-1)} = (\hat{z}_1^{(i-1)}, \ldots, \hat{z}_T^{(i-1)})$

Fig. 4 TCN residual block

convolutional layer, and $Activation(.)$ represents the activation function.

$$Z = Activation(X + H(X)) \tag{5}$$

The causal convolution layer, the causal dilated convolution layer, and the residual connection layer jointly form the temporal convolutional networks module, which outputs the feature embedding matrix Z.

Ultimately, the Temporal Convolution Networks Module demonstrates many advantages, such as a flexible receptive field size, stable gradients, and low memory requirements for training. The experimental results reported by Bai [19] indicate that in case of dilated convolutions and residual connections, a simple convolutional architecture is more effective across diverse sequence modeling tasks than recurrent architectures, such as LSTMs. Considering the above advantages, we construct the temporal convolution network module for time-series anomaly detection.

The process of our temporal convolution networks module is described in algorithm 1.

Algorithm 1 Process of TCNs.

```
Input:  Data sequence for anomaly detection
Output: Reconstruction sequence of input data
1: $Dilation\_rates = [0, 2, 4, 8]$
2: for all $i$ in $Dilation\_rates$ do
3:     Add Conv1D (32, 2, $activation = 'relu'$, $dilation\_rate = i, padding = 'causal'$)
4: end for
5: Add Dense Layer (128, $activation = 'relu''$)
6: Add Drop Layer ()
7: Add Dense Layer (1)
```

In algorithm 1, lines 1-8 define the temporal convolution networks.

3.4 Input sequence reconstruction module

After constructing the Temporal Convolution Networks Module, we use the normal data to train the TCNs model. Then the model learns an efficient way to reconstruct the current input value based on the normal historical sequence and the current input value. When we feed the input data with anomalies into the TCNs, the model reconstructs the data according to the parameters fitted during training. The normal data will reconstruct accurately. However, the reconstruction of the abnormal data that has never been observed before will greatly deviate. Finally, we calculate the list of anomaly scores for the input data by (2). More details are shown in algorithm 2.

Algorithm 2 Input sequence reconstruction module.

```
Input:  Original data sequence
Output: Reconstruction sequence of original data and the Anomaly Score list $L_{AS}$
1: Obtain the test data sequence and build the list $L_{in}$
2: Obtain the training data and build the list $L_{train}$
3: Build the reconstruction sequence list $L_{out}$
4: Build the anomaly score list $L_{AS}$
5: Train the TCNs model, $TCN.fit(x\_train = L\_train, y\_train = L\_train)$
6: $L_{out} = TCN.predict(L_{test})$
7: for $i = 0$ to $length(L_{AS})$ do
8:     $L_{AS}[i] = |L_{in}[i] - L_{out}[i]|$
9: end for
```

In algorithm 2, lines 1–5 train the TCNs model. Line 6 reconstructs the input sequence. Lines 7–8 calculate the Anomaly Score of original data.

3.5 Anomaly assessment module

In this module, we will first explain how to detect anomaly within a timestep and will then explain in detail how the model detects both point and collective anomalies.

After calculating the Abnormal Score between each input value and its reconstructed value, we obtain the Total Abnormal Score over a timestep by adding all the Abnormal Score values in a timestep. Obviously, AS is equal to TAS for point anomaly detection in a timestep. By analyzing the TAS at every timestep, we evaluate the possibility of detecting an anomaly. An anomaly will be identified if the TAS exceeds the Threshold. Usually, we sort the TAS list from small to large and select a larger value or a value greater than the maximum value as the threshold. The formula is shown in (6):

$$TAS > Threshold \tag{6}$$

Point anomaly detection module. As point and collective anomalies differ, we detect point anomaly before collective anomaly. For point anomaly detection, we set the timestep value to one $timestep = 1$, whereas we define the Total Abnormal Score list L_{PA-TAS} as $L_{PA-TAS} = L_{AS}$. If the TAS within a timestep exceeds the preset threshold, $PA\_Threshold$, we mark the point as a point anomaly.

Collective anomaly detection module. To detect collective anomalies, we first select a random AS value out of the normal data and define it as $Normal_AS$. Then we set the value of timestep, $timestep$, which should not be too large, as it will otherwise affect the accuracy of anomaly detection. When calculating the TAS value within a timestep, if there is data within the interval marked as point anomalies, we replace the AS value of the point with the selected constant $Normal\_AS$. This prevents point anomalies from interfering in the collective anomaly detection. If the TAS within a timestep exceeds the preset threshold, $CA\_Threshold$, we mark points that are not point anomalies in this timestep as collective anomalies.

In this manner, we can detect point and collective anomalies in the data. The process of anomaly assessment is described in algorithm 3.

In algorithm 3, lines 6–10 detect point anomalies. Lines 11–28 detect collective anomalies in each timestep.

Algorithm 3 Anomaly assessment.

```
Input:  The Anomaly Score list $L_{AS}$
Output:  None
1: $timestep = 1$
2: Build the Total Abnormal Score list $L_{PA-TAS}$ for point anomaly detection.
3: Build the Total Abnormal Score list $L_{CA-TAS}$ for collective anomaly detection.
4: Build the label list $L_{label} = [0, 0 \cdots 0]$
5: $L_{PA-TAS} = L_{AS}$, $L_{CA-TAS} = [0, 0 \cdots 0]$
6: for $i$ in $range(0, length(L_{AS}), timestep)$ do
7:     if $L_{PA-TAS}[i] > PA\_Threshold$ then
8:         $L_{label}[i] = 2$
9:     end if
10: end for
11: $timestep = n(n > 3)$
12: $Normal\_AS = get\_normal_AS(L_{AS})$
13: for $i$ in $range(0, length(L_{AS}), timestep)$ do
14:     for $j$ in $range(timestep)$ do
15:         if $(i + j) < length(L_{AS})$ AND $L_{label}[i + j] == 0$ then
16:             $L_{CA-TAS}[int(\frac{i}{timestep})] = L_{AS}[i + j] + L_{CA-TAS}[int(\frac{i}{timestep})]$
17:         else
18:             $L_{CA-TAS}[int(\frac{i}{timestep})] = Normal\_AS + L_{CA-TAS}[int(\frac{i}{timestep})]$
19:         end if
20:     end for
21: end for
22: $num = length(L_{CA-TAS})$
23: for $i$ in $range(num)$ do
24:     if $L_{CA-TAS}[i] > CA\_Threshold$ then
25:         for $j$ in $range(timestep)$ do
26:             if $i * timestep + j < length(L_{AS})$ AND $(L_{label}[(i * timestep) + j] \neq 2$ then
27:                 $L_{label}[i * timestep + j] = 1$
28:             end if
29:         end for
30:     end if
31: end for
32: for $i$ in $range(L_{label})$ do
33:     if $L_{label}[i] == 2$ then
34:         Print("The data is a point anomaly!")
35:     else if $L_{label}[i] == 1$ then
36:         Print("The data is a collective anomaly!")
37:     else
38:         Print("The data is no anomaly!")
39:     end if
40: end for
```

4 Experiments

4.1 Datasets

We choose five datasets to assess the ability of our unified model to efficiently detect anomalies in time-series problems. The details of the dataset are as follows.

The NSL-KDD dataset ("https://www.unb.ca/cic/datasets/nsl.html"). The NSL-KDD dataset is a revised and cleaned-up version of the KDD 1999 dataset. The dataset contains the records of the internet traffic identified by a simple intrusion detection network. Each record contains 43 features, 41 of which refer to the traffic input itself, whereas the last two are labels and score. 4 different classes of attacks exist within the dataset: Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). We restrict the experiment to a specific anomaly, Neptune, in Dos-of-Service (DoS) attacks. Neptune typically occurs for a certain period, which is represented by a set of anomalies. We use the number of data bytes from the source host to the target host to feature anomaly detection through analysis. In the experiments, the training set consists of 140000 normal data, whereas the test set consists of 7121 normal data and 2880 abnormal data. Significantly, these data should be normalized before the experiments. We use the Min-Max Normalization method to map data values between 0 and 1.

The KPI dataset ("http://iops.ai/dataset_detail/?id=10") ("http://iops.ai/competition_detail/?competition_id=5&flag=1). The dataset records key performance indicators used to determine the stability of web services. These KPIs are generally classified into two types: service and machine. Such as response time of a web page, page visits, number of connection errors, and so on. Each data has four features: timestamp, value, label, and KPI id. In the experiment, the feature value is used to detect anomalies. Further, we select two parts of the dataset for anomaly detection. Data from Experiment1 is extracted from data with timestamp between 1494200520 and 1498677480. The training set contains 59820 normal data and the test set contains 3520 normal data and 480 abnormal data. Data from Experiment 2 is extracted from data with timestamp between Data from Experiment1is taken from data with timestamp between 1494200520 and 1501432620. The training set contains 4800 normal data and the test set contains 1390 normal data and 10 abnormal data.

The Twitter dataset ("https://cgithub.com/numenta/NAB/tree/master/data/realTweets"). A collection of Twitter mentions of large publicly traded companies such as Google and IBM. It contains two features, timestamp, and value. The metric value represents the number of mentions for a given ticker symbol every 5 min. We choose the data with the timestamp from 2015-02-26 21:42:53 to 2015-04-22 23:47:53 for the experiment. The training set comprises 15160 normal data, whereas the test set comprises 2903 normal data and 87 abnormal data.

The Traffic dataset ("https://github.com/numenta/NAB/tree/master/data/realTraffic"). Real-time traffic data from the Twin Cities Metro area in Minnesota, collected by the Minnesota Department of Transportation. Metrics include occupancy, speed, and travel time from specific sensors. In our experiments, we use occupancy as a metric of anomaly detection. We selected data with timestamp from 2015-09-01 13:45:00 to 2015-09-17 16:24:00. The training set contains 17500 normal data, whereas the test set contains 2350 normal data and 30 abnormal data.

The AWS dataset ("https://github.com/numenta/NAB/tree/master/data/realAWSCloudwatch"). Amazon Web Services (AWS) server metrics as collected by the Amazon CloudWatch service. Example metrics include CPU Utilization, Network Bytes In, and Disk Read Bytes. We use CPU Utilization as a metric for our experiment. We selected data with timestamp from 2014-04-02 14:25:00 to 2014-04-16 14:20:00. The training set contains 3500 normal data and the test set contains 873 normal data and 127 abnormal data.

4.2 Evaluation metrics

We use ROC-AUC and PR-AUC as model evaluation metrics. The receiver operating characteristic curve (ROC) is a curve drawn according to a series of different dichotomy methods, with the true positive rate as ordinate and the false positive rate as abscissa. The precision-recall (PR) curve has the precision as ordinate and the recall as abscissa. The area under the curve (AUC) is defined as the area under the ROC or PR curve surrounded by the coordinate axis. The closer the PR-AUC and ROC-AUC are to 1.0, the higher the model accuracy. We can use the ROC-AUC to assess multiple classifiers and the PR-AUC to measure the classifier's ability to classify unbalanced data.

To fully evaluate the accuracy of the CPA-TCN model, we compare the performances of the vertical and horizontal experiments from different angles. The comparison model is set as follows:

LSTM [16]: The classical deep learning time series model.

GRU [14]: A variant of LSTM, which combines forget gate and input gate into a single update gate.

CNN [20]: A type of feedforward neural network with convolution computation and depth structure.

Table 1 Parameters in KDD

PA_timestep	PA_Threshold	CA_timestep	Normal_AS	CA_Threshold
1	0.3	10	0.0008	0.462629

4.3 TCNs model's results

In this part, we analyze the experimental results on five datasets separately.

– KDD dataset

When training the model, we use the mean absolute error as the loss function. The hyperparameters are set as follows: $batch\_size = 1024, epochs = 27$. The values of all important parameters are listed in Table 1.

The predicted results are shown below. The green curve represents the input time series, whereas the yellow curve represents the reconstruction of the input time series. The red point represents the collective anomaly detected by the model, whereas the blue point represents the detected point anomaly. As Fig. 5 shows, the reconstructed value of the normal value is remarkably close to the input value, while the reconstructed value of the abnormal value is far from the input value.

In the test set, the model detects 16800 collective anomalies and 0 point anomalies with $ROC\_AUC = 99\%$, $PR\_AUC = 97.619048\%$.

– KPI dataset

We conduct two sets of experiments using this dataset. Both experiments utilize the mean absolute error as the loss function. The hyperparameters of the first experiment are: batch_size=1024,epochs=27. The hyperparameters of the second experiment are: $batch\_size = 512, epochs = 49$. The parameters are shown in Table 2.

In experiment 1, the model detects 498 collective anomalies and 39 point anomalies with $ROC\_AUC = 98.71685\%$, $PR\_AUC = 93.95363\%$.

In experiment 2, TCNs model detects 0 collective anomalies and 15 point anomalies with $ROC\_AUC = 99.8201\%$, $PR\_AUC = 83.3333\%$.

– Twitter dataset

During TCN training, the hyperparameters are $batch\_size = 2048, epochs = 42$. Parameters are shown in Table 3.

In the test set, the model detects 3 collective anomalies and 107 point anomalies with $ROC\_AUC = 97.23611\%$, $PR\_AUC = 85.49312\%$.

– Traffic dataset

When training the model, the hyperparameter is set to: $batch_size = 1024, epochs = 90$. And the parameters are shown in Table 4.

In the test set, the model detects 0 collective anomalies and 35 point anomalies with $ROC\_AUC = 99.8936\%$, $PR\_AUC = 92.8571\%$.

– AWS dataset

The hyperparameters are set as follows: batch_size=1024, epochs=131. The values of the important parameters are shown in Table 5.

In the test set, the model detects 0 collective anomalies and 129 point anomalies with $ROC_AUC = 99.8854\%, tPR_AUC = 99.2248\%$.

Figures 5, 6, 7, 8, and 9 clearly show that the TCN model can detect both point and collective anomalies. When the model reconstructs the time series, we find that the reconstructed abnormal data largely deviates. That is the core principle of how the TCN model detects anomalies. For data containing both types of point anomalies, such as in Experiment 1 from the KPI dataset, the accuracy of point anomaly detection can be improved by first detecting point anomalies and then collective ones. This is because point anomalies that are not detected by the point anomaly detection module can be detected by the collective anomaly detection module.

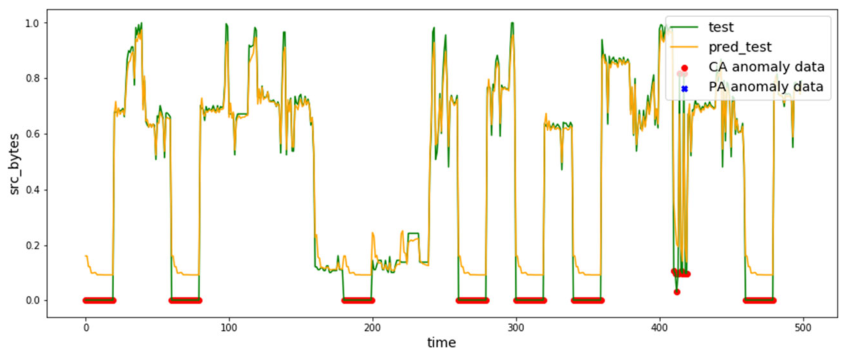


Fig. 5 Anomaly detection in KDD

Table 2 Parameters in KPI

Experiment	PA_timestep	PA_Threshold	CA_timestep	Normal_AS	CA_Threshold
Experimnet 1	1	0.21828	15	0.005	0.6639
Experimnet 2	1	52.2809	5	0.5	98.6383

Table 3 Parameters in Twitter dataset

PA_timestep	PA_Threshold	CA_timestep	Normal_AS	CA_Threshold
1	25.9062	3	0.15	44.5489

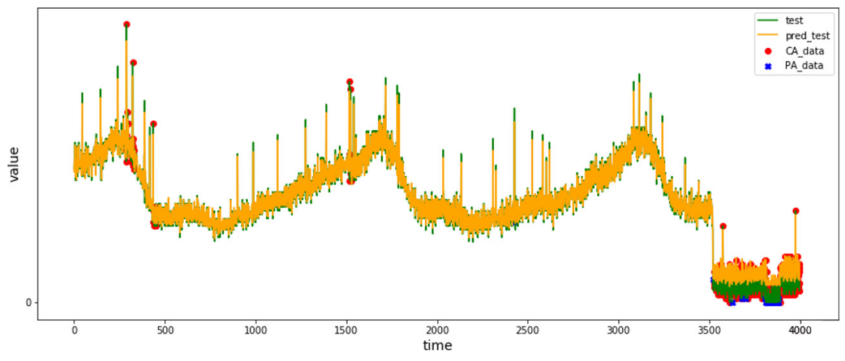
Table 4 Parameters in Traffic dataset

PA_timestep	PA_Threshold	CA_timestep	Normal_AS	CA_Threshold
1	4.5887	6	0.15	24

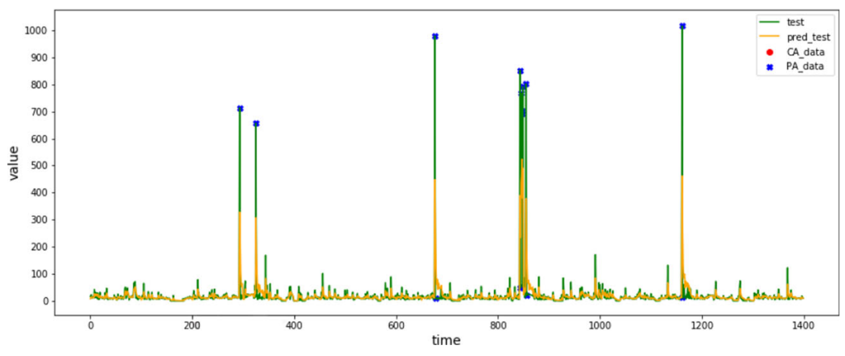
Table 5 Parameters in AWS dataset

PA_timestep	PA_Threshold	CA_timestep	Normal_AS	CA_Threshold
1	0.6328	20	0.15	2.2814

Fig. 6 Anomaly detection in KPI



(a) Experiment 1's reconstruction result.



(b) Experiment 2's reconstruction result.

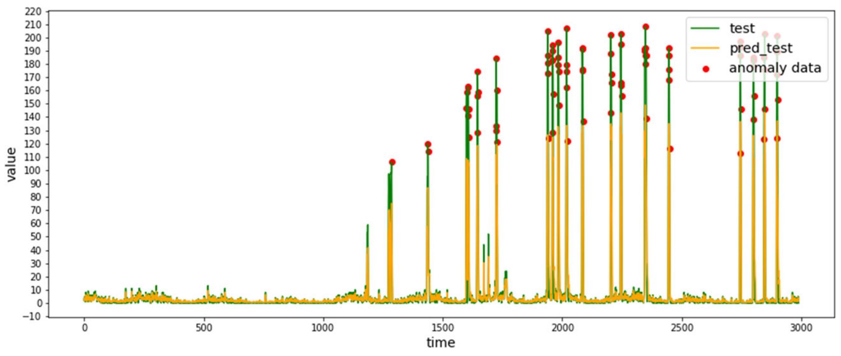


Fig. 7 Anomaly detection in Twitter dataset

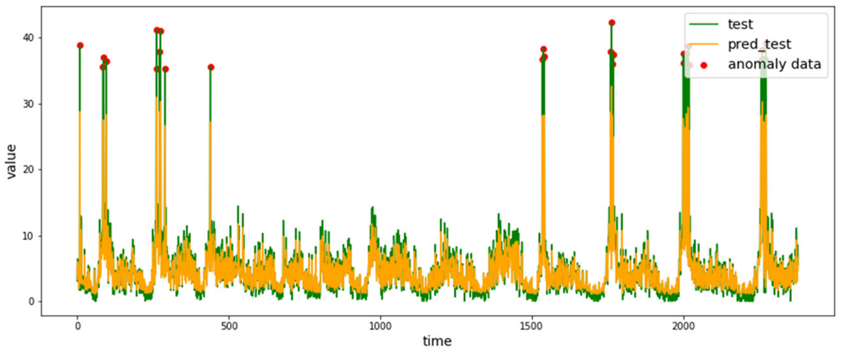


Fig. 8 Anomaly detection in Twitter dataset

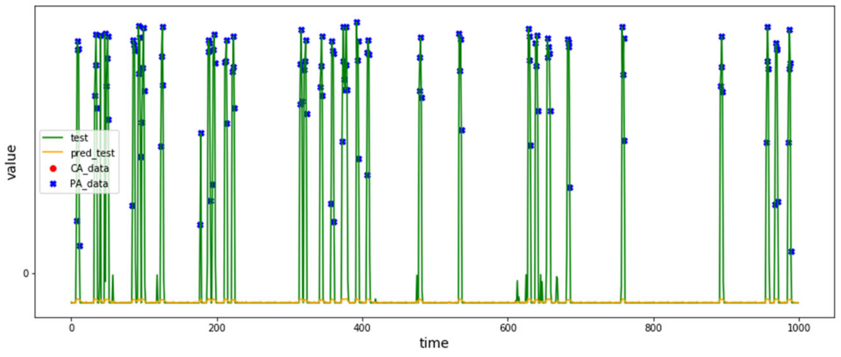


Fig. 9 Anomaly detection in AWS dataset

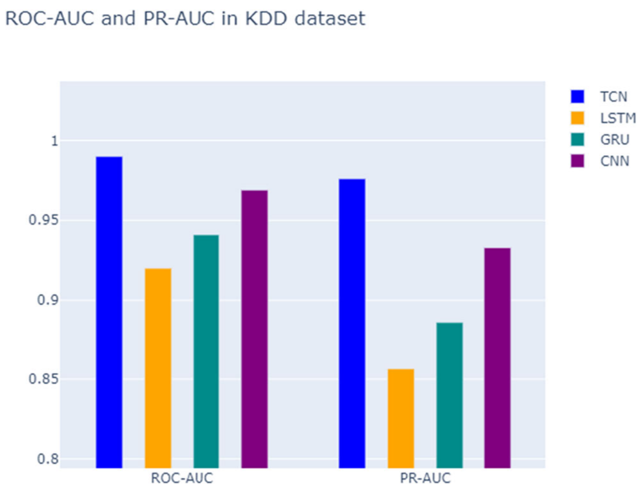


Fig. 10 Performances of four models in KDD dataset

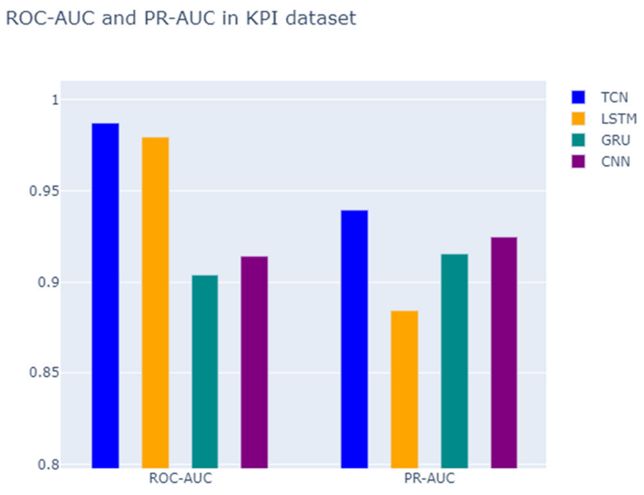


Fig. 11 Performances of four models in experiment 1 of KPI dataset

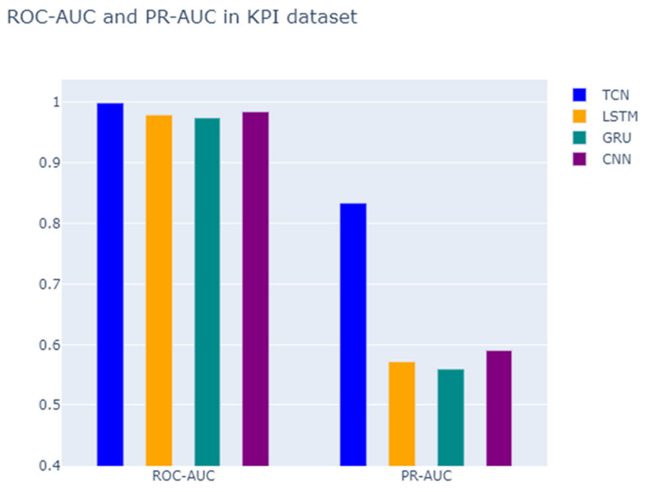


Fig. 12 Performances of four models in experiment 2 of KPI dataset

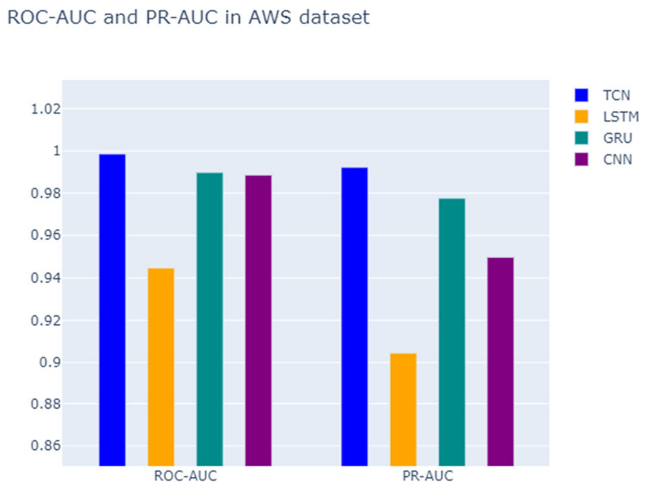


Fig. 14 Performances of four models in AWS dataset

The above experimental results show that: the CPA-TCNs can overcome the shortcomings of models that either detect point or collective anomalies. The disadvantages of a single model are the following. First, according to the nature of point and collective anomalies, the model that only detects point anomaly cannot detect the possible collective anomaly in the data. Second, models that detect only collective anomalies cannot easily eliminate the interference of point anomalies.

4.4 Results of comparative experiments

In this part, we compare the CPA-TCN model with the LSTM, GRU, and CNN models on five datasets.

Figures 10, 11, 12, 13, 14, and 15 show that the values of ROC-AUC and PR-AUC of the TCN model are greater than those of comparison models. In particular, the ROC-AUC of TCN is as high as 99%. That means that the CPA-TCN model can detect anomalies more accurately than the LSTM, GRU and CNN models on different datasets. Compared with RNN-based models, the experimental results show that CPA-TCN, which combines dilations and residual connections with the causal convolutions, is indeed superior to them in time series anomaly detection. This also proves that TCNs exhibit longer memory than recurrent architectures with the same capacity. Compared with CNN-based models, the experiment shows that CPA-TCN is also superior to the existing CNN-based models in the effect of each indicator. Because the CNN-based model used for anomaly detection only can detect point anomaly or collective anomaly, which limits the accuracy of anomaly detection of traditional CNN-based models.

In Experiment 2 of the KPI dataset, the ratio of abnormal to normal data is extremely unbalanced, with only 10 abnormal data included in the test set. The experimental results show that the PR-AUC value of the TCN model is 20% higher than those of the LSTM, GRU, and CNN

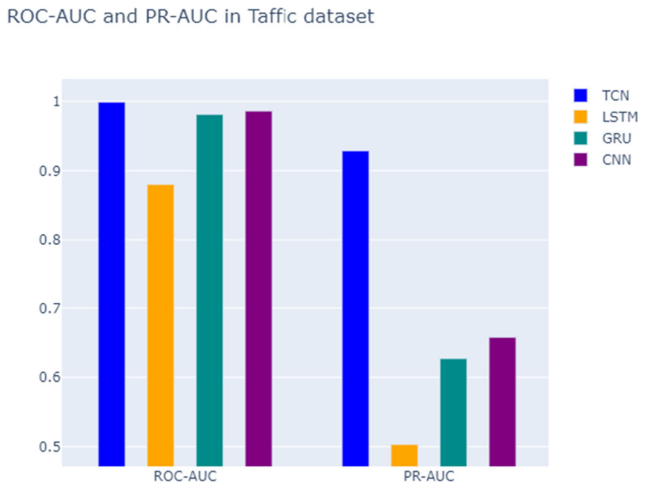


Fig. 13 Performances of four models in Traffic dataset

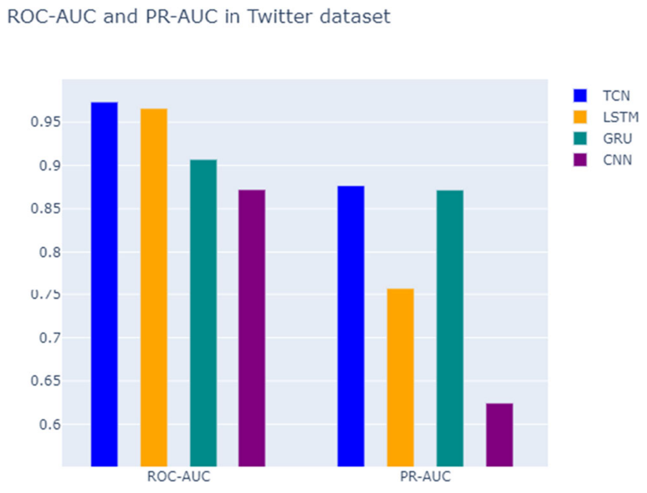


Fig. 15 Performances of four models in Twitter dataset

models. This indicates that the TCN model also has a better ability to classify unbalanced data.

5 Conclusion

This paper proposes a unified model for detecting both collective and point anomalies based on stacked temporal convolution networks. The CPA-TCN is a powerful toolkit for sequence modeling that outperforms RNN-based models. The most outstanding advantage of this model is that it can detect data mixed with point and collective anomalies even if the proportion of normal and anomalous data in the dataset is extremely unbalanced. Further, we detect point anomalies before detecting collective anomalies, which not only ensures that point anomalies that are not detected by the point anomaly detection module have the opportunity to be detected by the collective anomaly detection module but also enables the detected point anomalies to not interfere with the detection of collective anomalies. Our model is examined on the KDD, KPI, Traffic, AWS, and Twitter datasets for detecting anomalies. With the ROC-AUC of 0.98–0.99, the results suggest that our model can detect anomalies more effectively and accurately compared with LSTM, GRU, and CNN models. Based on the above advantages, we conclude that our model will provide a new direction for anomaly detection. In the future, we will continue to work on anomaly detection based on TCNs and we will attempt to apply this unified model to more different datasets to show the effectiveness of the model in anomaly detection.

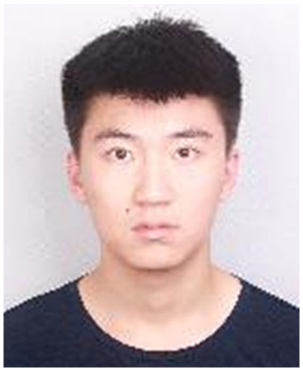
Acknowledgements This work is supported by the National Nature Science Foundation of China (No. 61672329, No. 62072290), Shandong Provincial Project of Graduate Education Quality Improvement (No.SDYY18058).

References

1. Han J, Kamber M, Pei J (2011) Data mining concepts and techniques third edition. Morgan Kaufmann Ser Data Manag Syst 5(4):83–124
2. Xu D, Wang Y, Meng Y, Zhang Z (2017) An improved data anomaly detection method based on isolation forest. In: 2017 10th international symposium on computational intelligence and design (ISCID), vol 2. IEEE, pp 287–291
3. Miao X, Liu Y, Zhao H, Li C (2018) Distributed online one-class support vector machine for anomaly detection over networks. IEEE Trans Cybern 49(4):1475–1488
4. de Rosa GH, Roder M, Santos DFS, Costa KAP (2021) Enhancing anomaly detection through restricted boltzmann machine features projection. Int J Inf Technol 13(1):49–57
5. Fang Y, Wang H, Zhao L, Yu F, Wang C (2020) Dynamic knowledge graph based fake-review detection. Appl Intell 50(12):4281–4295
6. Luo W, Liu W, Gao S (2017) Remembering history with convolutional lstm for anomaly detection. In: 2017 IEEE international conference on multimedia and expo (ICME). IEEE, pp 439–444
7. Chouhan N, Khan A et al (2019) Network anomaly detection using channel boosted and residual learning based deep convolutional neural network. Appl Soft Comput 83:105612
8. Vinayakumar R, Soman KP, Poornachandran P (2017) Applying convolutional neural network for network intrusion detection. In: 2017 international conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 1222–1228
9. Pereira J, Silveira M (2018) Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In: 2018 17th IEEE international conference on machine learning and applications (ICMLA). IEEE, pp 1275–1282
10. Yazdi HS, Bafghi AG et al (2020) A drift aware adaptive method based on minimum uncertainty for anomaly detection in social networking. Expert Syst Appl 162:113881
11. Kim T-Y, Cho S-B (2018) Web traffic anomaly detection using c-lstm neural networks. Expert Syst Appl 106:66–76
12. Provotar OI, Linder YM, Veres MM (2019) Unsupervised anomaly detection in time series using lstm-based autoencoders. In: 2019 IEEE international conference on advanced trends in information theory (ATIT). IEEE, pp 513–517
13. Nanduri A, Sherry L (2016) Anomaly detection in aircraft data using recurrent neural networks (rnn). In: 2016 integrated communications navigation and surveillance (ICNS). IEEE, pp 5C2–1
14. Qu Z, Su L, Wang X, Zheng S, Song X, Song X (2018) A unsupervised learning method of anomaly detection using gru. In: 2018 IEEE international conference on big data and smart computing (BigComp). IEEE, pp 685–688
15. Thi NN, Le-Khac N-A et al (2017) One-class collective anomaly detection based on lstm-rnns. In: Transactions on large-scale data-and knowledge-centered systems XXXVI. Springer, pp 73–85
16. Bontemps L, McDermott J, Le-Khac N-A et al (2016) Collective anomaly detection based on long short-term memory recurrent neural networks. In: International conference on future data and security engineering. Springer, pp 141–152
17. Ahmed M, Mahmood AN, Hu J (2016) A survey of network anomaly detection techniques. J Netw Comput Appl 60:19–31
18. Fengming Z, Shufang L, Zhimin G, Bo W, Shiming T, Mingming P (2017) Anomaly detection in smart grid based on encoder-decoder framework with recurrent neural network. J Chin Univ Posts Telecommun 24(6):67–73
19. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271
20. Zhang Z, Zhou X, Zhang X, Wang L, Wang P (2018) A model based on convolutional neural network for online transaction fraud detection. Security and Communication Networks 2018
21. Fu K, Cheng D, Tu Y, Zhang L (2016) Credit card fraud detection using convolutional neural networks. In: International conference on neural information processing. Springer, pp 483–490
22. Chouiekh A, Haj EL, Hassane IEL (2018) Convnets for fraud detection analysis. Procedia Comput Sci 127:133–138
23. Salimans T, Kingma DP (2016) Weight normalization: A simple reparameterization to accelerate training of deep neural networks. arXiv:1602.07868
24. Ha C, Tran V-D, Van LN, Than K (2019) Eliminating overfitting of probabilistic topic models on short and noisy text: The role of dropout. Int J Approx Reason 112:85–104

Zuhui Li is currently a student at the School of Information Science and Engineering, Shandong Normal University, China. She is under the supervision of Prof. Hong Wang at Shangdong Normal University (2018-). Her research interest is in the area of machine learning and anomaly detection.

Zhijie Xiang is currently a student at the School of Information Science and Engineering, Shandong Normal University, China. He is under the supervision of Prof. Hong Wang at Shangdong Normal University (2018-). His research interest is in the area of complex networks and personized recommendation.



Weijia Gong is currently a student at the School of Information Science and Engineering, Shandong Normal University, China. He is under the supervision of Prof. Hong Wang at Shangdong Normal University (2018-). His research interest is in the area of machine learning and data mining.

Hong Wang received the Ph.D. degree in computer applications from the Computing Institute of the Chinese Academy of Sciences, in 2002, and the Postdoctoral degree from the Postdoctoral Station of Computer Science and Technology, Shandong University, in 2009. She is currently a Ph.D. Professor and a Doctoral Tutor. She has published more than 60 articles in academic journals at home and abroad. As the person in charge of the project, she presided over the completion of the Shandong Young Scientist Award Fund, the Postdoctoral Grant Selection Program in Shandong, and the Natural Science Foundation of Shandong. Her research interest is in the area of machine learning, deep learning, graph learning, data mining, bioinformatics, personized recommendation.