



An effective parallel convolutional anomaly multi-classification model for fault diagnosis in microservice system

Xi Li¹ · Peian Wen¹ · Peng Chen¹ · Juan Chen¹ · Xuming Wen¹ · Yunni Xia²

Accepted: 1 April 2024 / Published online: 21 May 2024

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract

Microservice architecture is a new technology for deploying large-scale applications and services in the cloud. But multivariate time series data with anomalies are increasingly generated in the cloud. Effectively diagnosing the runtime system anomalies is necessary to ensure the quality of service of microservice systems. Typical anomaly detection methods are effective in data quality and computing reliability of cloud computing. However, they all focus on one-class anomaly detection, which may not perform on practical microservice frameworks with diverse types of anomalies. Furthermore, locating the root cause of anomalies to eliminate after detection is essential. To address these issues, we propose an effective parallel convolutional anomaly multi-classification model (PCAC) based on an attention mechanism for fault diagnosis in microservice system. We first construct a parallel convolutional structure that allows subnetworks to extract features independently. Then, channel and spatial attention mechanisms are applied in the parallel convolutional layers to mitigate the loss of feature representation. Finally, causal inference based on the anomalous graph is used to locate the fault in the microservice system. The experimental results clearly show that the proposed model achieves the highest F1 scores on six public microservice datasets, improved by 37.9% in average macro-F1 and 4.4% in average micro-F1 scores respectively, outperforming eight state-of-the-art methods.

Keywords Microservice · Attention mechanism · Anomaly multi-classification · Fault diagnosis

1 Introduction

With the rapid development of cloud computing technology, a substantial amount of multivariate time series data is generated in microservices stored system (Di Francesco et al., 2017). The microservice structure involves creating multiple applications that can work interdependently. It creates a streamlined delivery pipeline that decomposes the application into multiple small services, speeding up development and maintenance and providing greater flexibility. The microservice's distributed nature allows for its high scalability properties.

Xi Li and Peian Wen contributed equally to this work.

Extended author information available on the last page of the article

In real-world applications, a critical task is to detect anomalies in multivariate time series data in microservice systems. Diverse anomalies would be produced, such as memory leaks, network delays, and high CPU usage between the cooperations of microservice components. Currently, most anomaly detection methods aim at one-class anomalies (Wen et al., 2022; Chen et al., 2022; Song et al., 2023). Compared with traditional anomaly detection methods, multi-classification based on diverse anomalies in microservices architecture is becoming more complex. Convolutional neural networks (CNN) are used in general time series classification methods. But CNNs may not perform well since they cannot capture spatial information completely or lack attention to the correlations between convolutional channels in feature extraction (Fauvel et al., 2021). The deep learning network GDN (Deng & Hooi, 2021) uses graphs to model multivariate time series spatial features but does not consider temporal features. Multivariable time series data has become a typical data type, but for multivariable streams, both time dependence and correlation between observations should be considered. Thus, how to extract both spatial and temporal features from monitoring data is the challenge of multivariate time series anomaly detection.

The attention mechanism plays an important role in deep neural networks. Because attention gives the model the ability to discriminate, the machine will pay attention to the information that is more critical to the current task in a lot of information (Woo et al., 2018). The attention mechanism increases the performance of extracting diverse features and makes the neural network models more flexible.

An anomaly will propagate with the connection among microservices and eventually affect the whole system performance if the faults cannot be located in time. Log-based methods (Yang et al., 2021) detect and locate bugs based on log parsing. Even though discovering more informational causes, they are hard to work in real time and require abnormal information in log files. Thus, efficiently diagnosing the runtime system fault and being able to identify and locate it is a great challenge after anomaly detection. The graph structure provides the idea for fault diagnosis. We use a graph model and localize root causes with an algorithm similar to the random walk.

The main motivation is to accurately detect anomalies in the multivariate time series data from the monitoring data in microservices scenarios. And our main research questions are as follows: (1) How to improve the accuracy of anomalous multi-classification in microservice system? Specifically, microservice data collected from monitors are generally stored as multivariate time series; (2) when anomalies occur in the monitoring data, how to effectively identify the dimensions that have the greatest impact on generating the anomaly, to better locate the root cause? We approach a method to classify the various anomaly events in monitoring microservice data in the cloud and identify the abnormal time series that are most likely to be the causes of each anomaly in system. The proposed PCAC model includes two parts: anomaly detection and fault localization, as shown in Fig. 1. In the part of anomaly detection, there are two modules: feature capturing and anomaly multi-classify. First, we construct a convolutional structure, which contains two branches in parallel. To capture the association features in microservice data, one branch extracts the channel features with attention, and another extracts the spatial feature with attention. Then, based on both features, anomaly multi-classification is achieved. Another part of fault localization includes anomalous graph and causal inference modules. In this part, we use causal inference methods to learn the fault propagation paths generated by graph methods.

The main contributions of this work are summarized as follows:

- To address the difficulty of extracting spatial and temporal features from multivariate time series, we designed a parallel convolution architecture, which can better capture

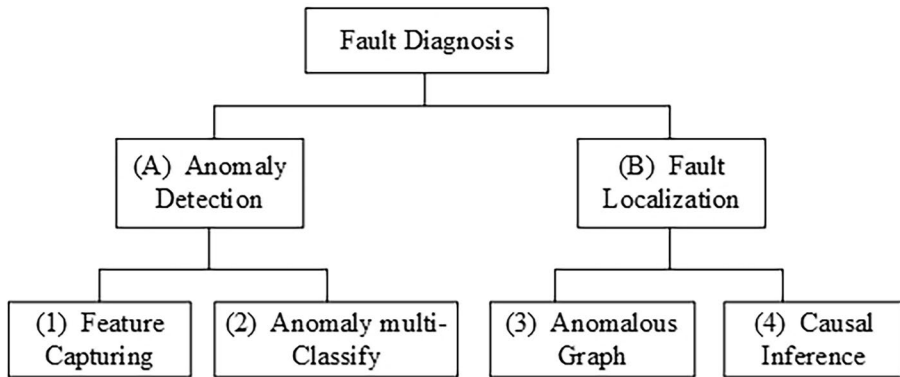


Fig. 1 PCAC module structure

the spatio-temporal dependencies of multivariate time series simultaneously to achieve better anomaly detection for microservice system.

- To solve the problem of incomplete feature extraction in ordinary CNNs, we propose a method with channel and spatial attention mechanisms to extract features in subnetworks independently and reduce the loss of feature representations.
- To effectively determine the fault cause after anomaly detection, we analyze and compare several causal inference-based cause localization methods to identify the specific fault service.
- We conduct experiments on eight state-of-the-art baseline methods on six public microservice datasets, with a 37.9% improvement in average macro-F1 and a 4.4% improvement in average micro-F1 scores, respectively.

Section 2 reviews different anomaly detection methods for microservice systems. Section 3 introduces the proposed model in detail. Section 4 evaluates the effectiveness of the model through comparative experiments and ablation experiments, analyzes the abnormal cause, and diagnoses the fault service. Section 5 summarizes the work and presents potential future research.

2 Related work

The study of anomaly detection has been carried out for several decades and is an active research area gaining increasing attention in deep learning. At the same time, many anomaly multi-classification methods for microservice system are proposed. We mainly review the related work on statistical model-based, machine learning-based, deep learning-based, and root cause localization methods.

2.1 Statics-based method

Generalized autoregressive conditional heteroskedasticity (GARCH) (Engle, 1982) is a method for modeling the volatility caused by conditional mean and conditional heteroscedasticity of monitoring microservice metrics. It calculates each point's anomaly score,

clusters the points, and then detects multiple categories of anomalies in the microservice system. Principal component analysis (PCA) (Shyu et al., 2003) extracts data features by dimensionality reduction and achieves anomaly classification on the low-dimensional data.

2.2 Machine learning-based methods

Support vector machine (SVM) (Kriegel et al., 2011) is a binary classification model. Multiple binary classifiers will be constructed in the microservice system to obtain the predicted probabilities by comparing each classifier in the testing set. The goal is to find a hyperplane with a maximum margin that can separate points of different classes. K -nearest neighbor (KNN) algorithm (Kiss et al., 2014) predicts the label of data by the labels of its K -nearest pre-marked neighbors. A decision tree (Lewis, 2000) is a classification model that formulates the features and anomaly labels that indicate the relations among the data points.

2.3 Deep learning-based method

Autoencoder (AE) (Fan et al., 2018; Xin et al., 2023) is a common neural network model that consists of an encoder and a decoder. The encoder extracts features by defining the neural architecture, and the decoder is similar to the encoder, to convert the encoding back to the original data. After training AE, the encoded features are used to train a classifier for anomaly classification. CNN (convolutional neural network) (LeCun et al., 1998) extracts microservice system monitoring metrics features by setting convolution kernels based on sliding windows and performs classification using the cross-entropy function. FCN (fully convolutional network) (Long et al., 2017) replaces fully connected layers with convolutional layers based on the convolutional neural network. LSTM (long short-term memory) (Graves & Graves, 2012) is a variation of RNN (recurrent neural network), which mitigates the problem of exploding gradients to some extent. The output of LSTM will go through a fully connected layer and softmax function to convert it into the probability distribution of each category. TapNet (Zhang et al., 2020; Xu et al., 2022) uses LSTM and CNN stacking to model microservice system monitoring metrics and classify them via softmax. MTEXCNN (Assaf et al., 2019) employs three cascaded 2D convolutions to extract spatial information, followed by a 1D convolution to extract temporal information as well as classify microservice system monitoring metrics. TranAD (Tuli et al., 2022) utilizes transformer-based adversarial training to detect anomalies, while GDN (Deng & Hooi, 2021) employs graph structure learning to capture the relationships between different sensors. Both TranAD and GDN are relatively novel methods for anomaly detection, demonstrating excellent performance. We apply the same modifications as those used for LSTM mentioned above to adapt these two models for multi-class anomaly classification, enabling a comparison with our proposed model.

2.4 Root cause localization method based on causal inference

The dependencies between services in a microservice application may cause the propagation of faults. Root cause localization helps our anomaly multi-classification models diagnose the source for anomalies and find the most fundamental reason for their occurrence. Based on the fault propagation paths, graphs methods to locate the root cause of fault are

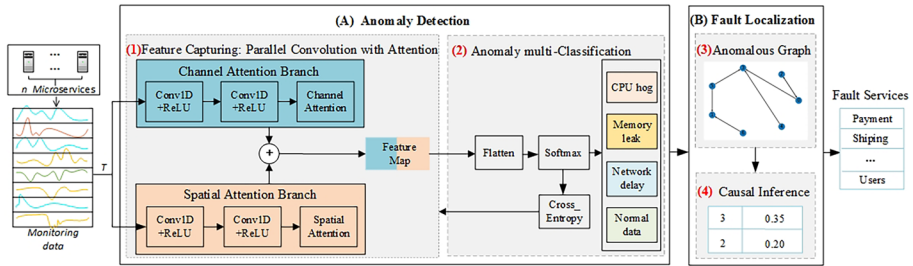


Fig. 2 Architecture of PCAC

developed. For example, AutoMap (Deng & Hooi, 2020) treats the different components in the system as individual nodes, and their interdependencies form a graph, and then finds the root cause based on the PC (Spirtes et al., 2000) algorithm and PageRank (Page et al., 1999) algorithm. Causeinfer (Chen et al., 2016) uses the PC algorithm to build a causal graph and then uses Breadth First Search (BFS) to infer the root cause of the causal graph. MicroDiag (Wu et al., 2021) uses linear non-Gaussian acyclic model (LiNGAM) (Hyvärinen et al., 2010) to learn the fault propagation relationship between microservices, build a fault propagation graph, and use PageRank to perform root cause localization on the propagation graph. The above methods ignore the capture of fault patterns of entity measurement data. However, some faults in the measurement data related to entities during a system fault may affect the final root cause localization results (Dongjie et al., 2023). Thus, capturing the fault patterns of measurement data in root cause localization and improving localization accuracy become challenges.

3 Method

3.1 Overall architecture

The architecture of the proposed PCAC is shown in Fig. 2. PCAC is composed of four modules: feature capturing (1), anomaly multi-classification (2), anomalous graph (3), and inferred causal (4). The input T of the model is system metrics of multivariate time series from microservice data monitoring. Firstly, it enters the feature capturing module (1), which consists of the channel attention branch and spatial attention branch. Two branches in parallel process data and a weighted feature map are generated. Module (1) solves the problem of incomplete feature extraction and gains spatio-temporal dependencies. Then, anomaly multi-classification is achieved through the operations of flattening and softmax on the attention map in module (2). Module (2) uses the cross-entropy to update the parameters to reduce the loss of features. Based on the multi-classification results, the root cause analysis is carried out by an anomalous graph generation in module (3). Finally, the probabilities of fault services are output in module (4) for cause location, effectively avoiding spreading among microservices.

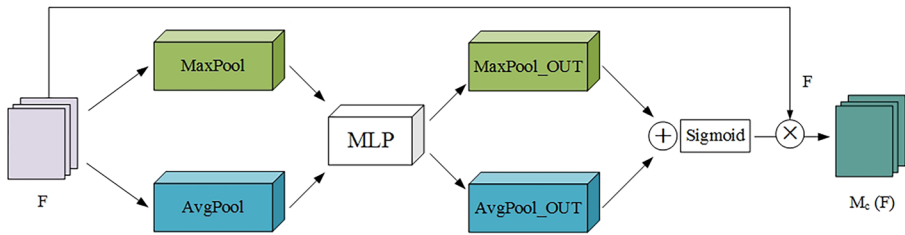


Fig. 3 Channel attention

3.2 Feature capturing: parallel convolution with attention

Data is input into the upper and lower branches simultaneously. In the upper branch, the data is passed through two one-dimensional convolution operations and corresponding activation functions before entering the channel attention function, which adjusts the importance of each channel in the model by learning attention weights and strengthens the ability to capture the correlation between multiple channels. In the lower branch, spatial attention branch, the data undergoes the same process as in the upper branch and then enters the spatial attention function, which adds weights to different positions to enable the model to focus better on critical local features in the system metrics. Finally, the feature maps output by the two branches are concatenated to obtain the final feature map.

Conv1D represents the one-dimensional convolution, and ReLU represents the activation function ReLU. Feature map represents the weighted feature map obtained by concatenating the features from the channel attention and spatial attention branches based on attention mechanism. For the given input feature tensor F , we compute the channel attention map $M_c(F)$ and the spatial attention map $M_s(F)$ at two separate branches, then compute the attention map $M(F)$ as follows:

$$M(F) = M_c(F) + M_s(F) \tag{1}$$

Channel attention The process of channel attention based on attention mechanism (Fauvel et al., 2021) is shown in detail in Fig. 3. To aggregate the feature map in each channel, we take two global pools on the feature F and produce the channel attention feature $M_c(F)$. As shown in Fig. 3, the channel attention mainly includes a shared multi-layer perceptron (MLP) network, a maximum pooling (MaxPool), and an average pooling (AvgPool).

Firstly, use MaxPool and AvgPool to extract feature information and input them to the shared MLP network to obtain the corresponding MaxPool_OUT and AvgPool_OUT. After that, the MaxPool_OUT and AvgPool_OUT are spliced and activated by the sigmoid function to obtain the attention score matrix. Finally, multiply the attention score matrix with the original input feature tensor F to get the channel attention map $M_c(F)$. The calculation formula is as follows:

$$M_c(F) = F \times \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \tag{2}$$

where σ is the sigmoid function, and \times is the matrix multiplication. Each channel has an attention score because channel attention adds weight to feature information. Using the sigmoid function ensures that the scores of different channels are independent. The attention

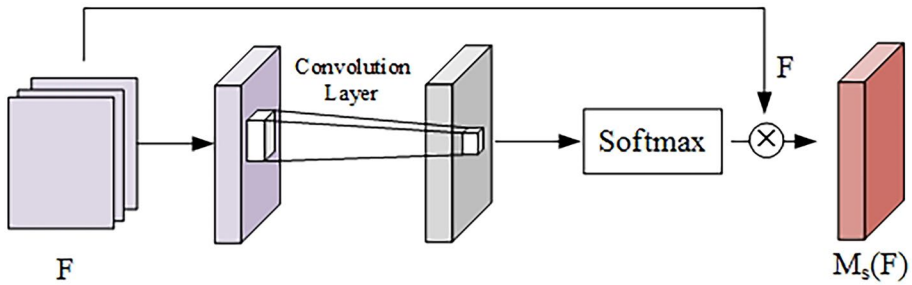


Fig. 4 Spatial attention

score matrix multiplies the original input, and then different weights are given according to the degree of importance. That means the original data is filtered and selected.

Spatial attention The proposed model introduces a spatial attention mechanism (Fauvel et al., 2021) to enhance the ability to capture features in different spatial locations. As shown in Fig. 4, the spatial attention map $M_s(F)$ is calculated as follows:

$$M_s(F) = F \times \sigma(\text{Conv1D}(\text{ReLU}(\text{Conv1D}(F)))) \tag{3}$$

where σ is the softmax function. The spatial attention mechanism weights different features at different positions, considering the relationship between the score at each position and other positions. Using the softmax function ensures that the sum of the scores is 1, thereby ensuring global consistency.

3.3 Anomaly multi-classification

The final attention map from parallel convention with attention module inputs the flatten and softmax function, respectively. Data is converted into one-dimension vectors through the flatten function and then output into probabilities p for multi-classification through the softmax function. That is, date is labeled as normal and abnormal. Furthermore, the abnormal data is predictably labeled as different types of anomalies based on probabilities, such as memory leak, network delay, or high CPU hog, which usually occurs during service invocation in the microservice system.

In the training phase, the loss is calculated using a cross-entropy loss function defined as Eq. 4. We use the calculated loss to perform parameter updating. Using the cross-entropy loss function as the optimization objective during training allows the model to continuously adjust its parameters during training to minimize the difference between the predicted probability and the actual label.

$$loss = -\frac{1}{n} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} y_{ij} \log(p_{ij}) \tag{4}$$

where n is the number of training samples, m is the number of classes, y represents the actual label, and p represents the probability of the label predicted by the model.

In the testing phase, the test data is processed the same way as in the training phase, but the model is not updated. Instead, the macro-F1 and micro-F1 scores of the model are calculated.

3.4 Fault localization

Once anomalies are detected, the fault location engine in the microservice system starts to trace the execution paths and then locate faulty services. The engine is composed of two main procedures: anomalous graph construction and causal inference. Fault localization procedure is as follows:

Step 1: Select a causal inference algorithm and construct a directed acyclic graph (DAG) with minimum loss information as anomalous graph G based on data after anomaly multi-classification.

Step 2: Use PageRank algorithm on G to compute the score of each anomalous node.

Output: Anomalous graph G and probability of each anomalous node.

We choose four common causal inference algorithms to construct causal graphs in order to find the best one to root causes analysis, including Peter-Clark (PC) (Spirtes et al., 2000) algorithm, Greedy Equivalence Search (GES) (Chickering & Boutilier, 2003) algorithm, and linear non-Gaussian acyclic model (LINGAM) (Hyvärinen et al., 2010) algorithm, in which ICA-LINGAM (Shimizu et al., 2006) and Direct-LINGAM (Shimizu et al., 2011) are included. In order to locate the faulty services, a graph centrality algorithm named PageRank (Page et al., 1999) is used on the anomalous graph and outputs the probability of each anomalous node. In the root causal inference phase, these probabilities serve as the basis to diagnose which microservice is most likely to cause faults.

4 Experiments

4.1 Datasets and experimental setup

Datasets Sock Shop¹ is a widely used microservice benchmark designed to test and evaluate microservices technology. It consists of 13 microservices, in which we mainly choose the front-end, catalogue, users, orders, payment, and shipping. The microservice architecture of Sock Shop is shown in Fig. 5. The complex connections between them make the multi-classification task of our model more challenging for those multivariate time series data in the microservice system. We deploy the Sock Shop using Kubernetes on multiple virtual machines (VMs) in the cloud. The Kubernetes cluster includes one master node and three worker nodes. We deploy open-source monitoring and visualization tools Prometheus² and Grafana³ on the master node to monitor the application and collect data. Furthermore, we use the load generation tool Locust⁴ on the master node to simulate

¹ <https://github.com/microservices-demo/microservices-demo>.

² <https://github.com/prometheus/prometheus>.

³ <https://github.com/grafana/grafana>.

⁴ <https://locust.io/>.

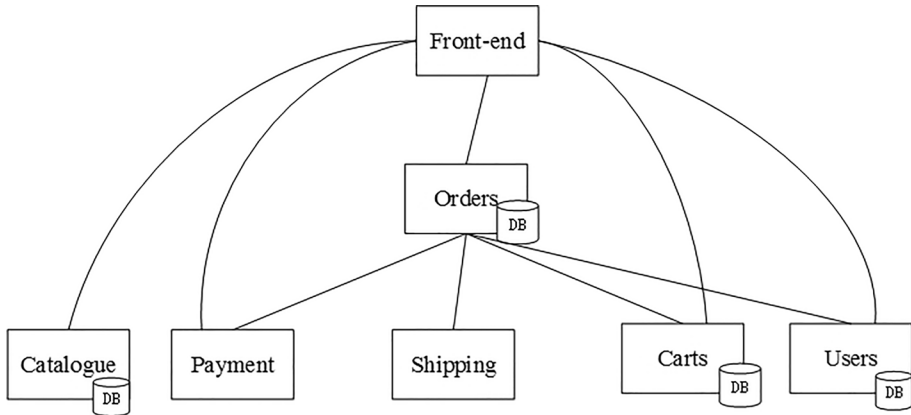


Fig. 5 The microservice architecture of Sock Shop

workloads for the microservice application. All services of Sock Shop are deployed on the nodes allocated to different VMs automatically.

To simulate realistic scenarios, we inject three types of anomalies into our experiment: CPU hog, memory leak, and network latency (Mariani et al., 2018; Chen et al., 2015). The Pumba⁵ tool was utilized to simulate network failures, and Docker container resources were subjected to stress tests to induce anomalies. The duration of each anomaly ranged from 1 to 5 min, while the application ran normally for 10 to 30 min, after which the process was repeated for each anomaly at least five times. Data is collected in real time every 5 s, based on Prometheus configuration, including service-level and resource-level data. At the service level, the latency of each service is recorded. At the resource level, metrics related to container resources are collected, including CPU hog, memory leak, and network transmit bytes.

Table 1 shows the details of six microservice datasets, including the size of the training set and test set and the number of feature dimensions. The ratio of training and test size is seven to three. In addition, three types of anomaly proportions are represented.

Metrics We use marco-F1 and micro-F1 scores as evaluation indicators to verify the performance of the model and compare it to other baseline anomaly detection methods. Both marco-F1 and micro-F1 are commonly used to evaluate models in multi-classification scenarios. Macro-F1 is calculated by average precision and recall score regardless of the importance of the different classes. Micro-F1 is suitable for a dataset with an unbalanced multi-classification distribution.

Baseline methods We compared different types of anomaly multi-classification models to validate the effectiveness of our model. These include (i) classical machine learning models GaussianNB, KNN, SVM, and SGD and (ii) the deep learning models CNN, DNN, LSTM, OmniAnomaly, transformer-based TranAD, and graph structure-based GDN. TranAD and GDN are relatively new anomaly detection models.

⁵ <https://github.com/alexei-led/pumba>.

Table 1 The details of datasets used in experiments

Microservice	Train size	Test size	Feature dimension	CPU hog (%)	Memory leak (%)	Network latency (%)
Catalogue	3229	1384	35	5.22	2.62	4.94
Front-end	3449	1478	36	4.87	5.38	4.18
Orders	3273	1403	35	3.89	3.08	7.12
Payment	3081	1320	36	4.66	5.18	4.11
Shipping	3097	1327	36	3.82	3.84	5.45
Users	3175	1360	36	3.97	3.75	2.69

Experimental settings All experiments are implemented in Python 3.7.11 and PyTorch 1.6.0 using a single NVIDIA GeForce 940MX (12 G) GPU, Intel (R) Core (TM) i7-7500U CPU @ 2.70GHz, and 12 G RAM. The convolutional kernel size in Conv1D is 3, 5, the number of epochs is 80, the batch size is 128, and the neural networks are optimized by the Adam optimizer, with the initial learning rate set to 10^4 .

4.2 Main results

We compare PCAC with eight baseline methods on six microservice datasets in Table 2 and Fig. 6 in terms of macro-F1 and micro-F1. The best performance is bolded.

Table 2 shows that PCAC achieves the highest macro-F1 and micro-F1 scores overall in eight baseline methods on the six datasets. Furthermore, we provide the ranking of PCAC and all baseline methods on macro-F1. The micro-F1 specific ranking score differs slightly from the macro-F1, but the model ranking is the same. Our model exceeds the other methods from the ranking results, proving that our method is effective in multi-classify.

The details of anomaly detection results of our method are shown in Fig. 7. It can be seen that on catalogue, front-end, orders, payment, shipping, and users datasets from Fig. 7a–f, the detection error rate is only 1.95%, 3.18%, 2.78%, 1.59%, 2.86%, and 4.12%, respectively. Users dataset requires more accurate detection of CPU hog and memory leak. In summary, our method has demonstrated an average false alarm rate of only 2.75% on the six datasets, indicating its effectiveness for detecting the three types of anomalies and cascading them to achieve better performance in fault diagnosis.

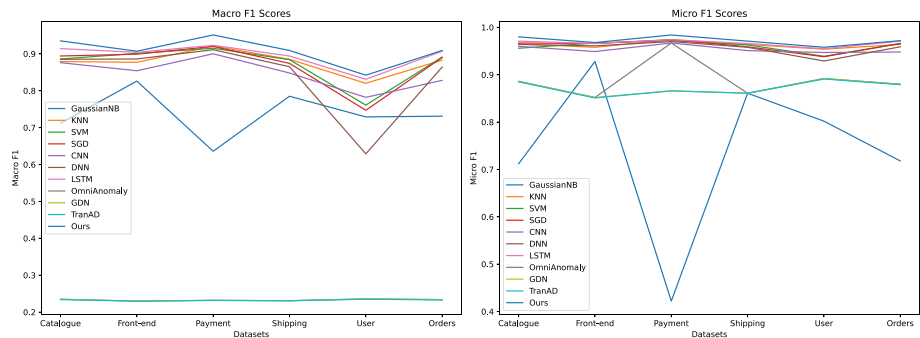


Fig. 6 Performance comparison

Table 2 Performance of baseline models and ours

Method	Catalogue		Front-end		Orders		Rank
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	
GaussianNB	0.712	0.712	0.826	0.928	0.731	0.718	
KNN	0.879	0.965	0.877	0.958	0.883	0.964	
SVM	0.886	0.956	0.901	0.967	0.891	0.966	
SGD	0.894	0.967	0.899	0.966	0.89	0.966	
CNN	0.876	0.96	0.854	0.949	0.828	0.948	
DNN	0.885	0.964	0.886	0.961	0.864	0.959	
LSTM	0.914	0.971	0.904	0.966	0.907	0.971	
OmniAnomaly	0.235	0.886	0.23	0.852	0.234	0.88	
GDN	0.235	0.886	0.23	0.852	0.234	0.88	
TranAD	0.234	0.885	0.23	0.851	0.233	0.879	
Ours	0.935	0.98	0.907	0.968	0.909	0.972	

Method	Payment		Shipping		Users		Rank
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	
GaussianNB	0.636	0.422	0.785	0.861	0.729	0.802	7.8
KNN	0.921	0.974	0.885	0.964	0.82	0.954	4.3
SVM	0.915	0.972	0.884	0.963	0.761	0.939	4
SGD	0.92	0.974	0.874	0.959	0.747	0.938	4.2
CNN	0.9	0.967	0.848	0.951	0.782	0.947	6.5
DNN	0.911	0.97	0.865	0.958	0.629	0.929	6
LSTM	0.923	0.974	0.894	0.966	0.831	0.955	2.1
OmniAnomaly	0.232	0.867	0.231	0.861	0.236	0.892	9
GDN	0.232	0.866	0.231	0.861	0.236	0.892	9
TranAD	0.232	0.866	0.231	0.861	0.235	0.891	10
Ours	0.951	0.984	0.909	0.971	0.842	0.958	1

Values in bold indicate the best performance

4.3 Ablation study

To investigate the impact of each component branch on the PCAC performance, we repeat the experiments without channel attention or spatial attention successively on the six datasets.

Channel attention The results of macro-F1 and micro-F1 in the experiment of PCAC and of PCAC without channel attention (CA) are shown in Fig. 8. The values of both macro-F1 and micro-F1 of PCAC without CA decrease on all datasets. Particularly, a decrease of 7.45% in macro-F1 and -2.10% in micro-F1 on the shipping dataset indicates that channel attention can increase the model's attention to specific channels and improve performance.

Spatial attention Similarly, as shown in Fig. 9, the performance of PCAC is better than PCAC without spatial attention (SA) on most datasets. The values of macro-F1 and micro-F1 decrease by 2.15% and 0.62% on the payment and front-end datasets, indicating that spatial attention is beneficial for capturing spatial information.

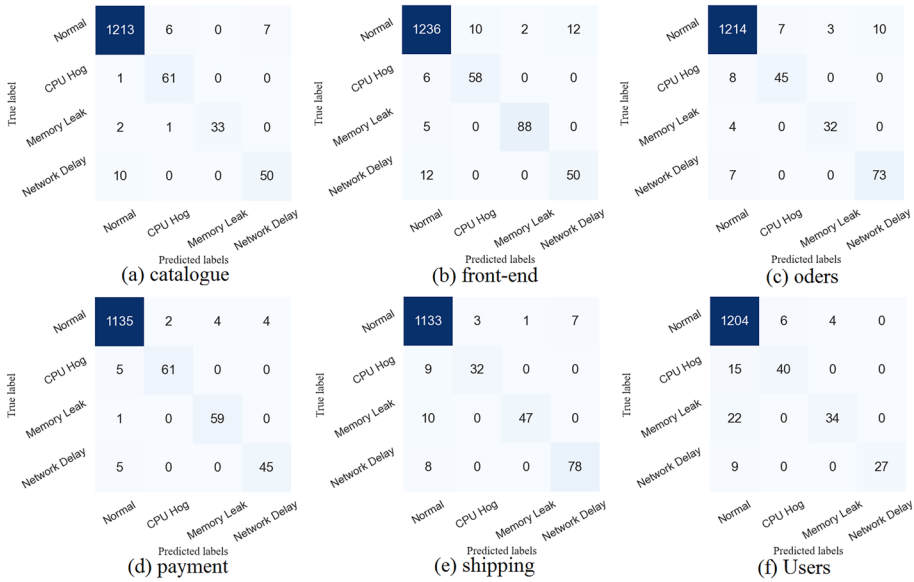


Fig. 7 Confusion matrix of anomaly detection results under the different microservices

4.4 Parameter sensitivity analysis

Batch_size This represents the size of the batch. Sensitivity analysis of batch size is helpful for hyperparameter adjustment. We applied different batch sizes on the catalogue datasets used in the experiment. The experimental results are shown in Table 3.

It can be seen from Table 3 that the Macro-F1 score is the best when batch_size is 128, but there is little difference in the Macro-F1 scores produced by other batch_sizes. It shows that the size of batch_size has little effect on the classification results of the model. However, the performance decreases when the batch_size is 256, while the performance is optimal when the batch_size is 128. Thus, the batch_size can be adjusted dynamically. In practice, we can choose different batch_sizes to make full use of computational resources without causing low computational efficiency.

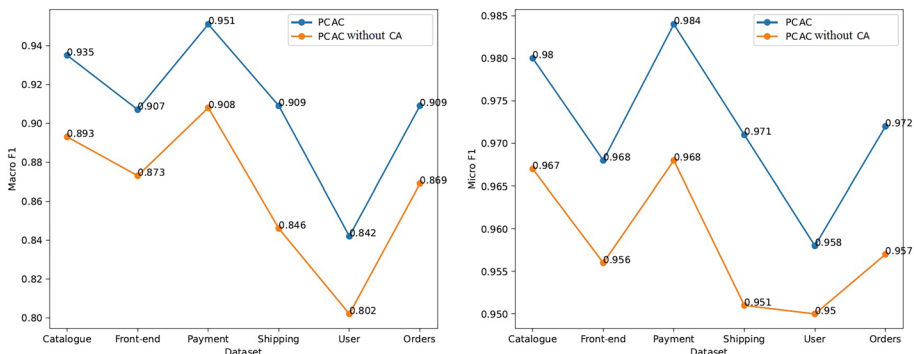


Fig. 8 Performance comparison between PCAC and PCAC without CA

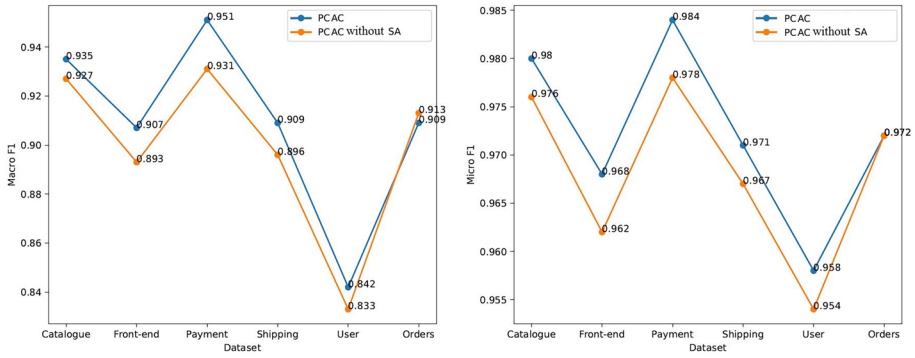


Fig. 9 Performance comparison between PCAC and PCAC without SA

Reduction_ratio This controls the dimension reduction ratio of the fully connected layer in the channel attention module. For example, when reduction_ratio is 16, it means that the output dimension of the fully connected layer will be 1/16 of the input dimension. We also performed an analysis on the catalogue dataset used in the experiment, and the experimental results are shown in Table 4.

It can be seen from Table 4 that the macro-F1 score is highest when the reduction_ratio is 16, but there is not much difference with the macro-F1 scores generated by other reduction_ratio values. Therefore, similar to batch_size, we can choose different reduction_ratio values according to the size of the dataset dimensions to balance the model’s performance and computational cost.

Parameter sensitivity experiments show that the two parameters batch_size and reduction_ratio in the model are stable, and their values do not have much influence on the results of anomaly detection. Therefore, the model proposed in this paper has strong robustness, and the performance does not fluctuate greatly with the values of the two parameters. In practical applications, we can adjust the values of batch_size and reduction_ratio according to the computational cost consideration.

4.5 Fault location

In this subsection, we select the front-end dataset as a sample to complete the fault diagnosis based on the anomaly multi-classification results of our model. Firstly, we choose the common four algorithms, including PC, GES, ICA-LINGAM, and Direct-LINGAM, to find a directed acyclic graph corresponding to anomalies with minimum loss. Then, we use the PageRank algorithm to perform a random walk on the anomalous graph and calculate the probability of each anomaly node. Finally, based on the ranking of these probabilities, we analyze the most likely fault cause in the system.

Table 3 Results of different batch_size on catalogue dataset

Batch_size	Macro-F1	Micro-F1
32	0.922	0.975
64	0.921	0.975
128	0.935	0.980
256	0.915	0.971

Table 4 Results of different reduction_ratio on catalogue dataset

Reduction_ratio	Macro-F1	Micro-F1
4	0.921	0.974
8	0.918	0.973
16	0.935	0.980
32	0.923	0.974

The anomalous graphs generated by the Direct-LINGAM algorithm are shown in Table 5. The nodes in the figure represent 0 (front-end), 1 (user), 2 (catalogue), 3 (orders), 4 (carts), 5 (payment), and 6 (shipping), seven microservices.

Different microservices may cause various anomalies. We aim to obtain the most fundamental microservice that caused the anomaly and diagnose the source, so we choose the PageRank algorithm to perform in anomalous propagation graphs in Table 5 to calculate the probabilities of an abnormality occurring in each node, and finally, we select the top 5 nodes as the most fundamental cause of the anomaly. The results of PageRank are shown in Table 6.

The above table shows that different microservices may cause various anomalies, and the fraction of exceptions occurring for each microservice varies. For example, regarding CPU anomalies, the most likely are orders and payment; memory anomalies, the most likely are front-end and shipping; and latency anomalies, the most likely are carts and front-end.

In the network, most failures may not be caused by a single cause. Specially, in microservice systems, an exception in any service may lead to the failure of a series of related services, because these microservices call each other and affect each other. Table 4 describes the probability of failure for each microservice. For example, network latency in someplace of microservice systems may be caused by 4 (carts) service and 0 (front-end) service because the two services have a combined failure probability of 79%.

Metrics To quantify the performance of each algorithm on a set of anomalies A , we use two wide metrics: $PR@k$ and $AVG@k$. $PR@k$ represents the probability that the top k results given by an algorithm include the real root cause. A higher $PR@k$ score, especially for small values of k , represents the algorithm correctly and identifies the root cause according to each anomaly a . $AVG@k$ evaluates the overall performance of a method by computing the average $PR@k$. They are defined as follows:

$$PR@k = \frac{1}{|A|} \sum_{a \in A} \frac{\sum_{i < k} R^a(i) \in V^a}{\min(k, |V^a|)} \tag{5}$$

Table 5 Anomalous graph

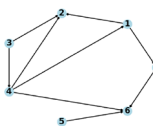
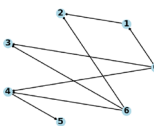
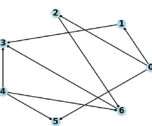
	CPU hog	Memory leak	Network latency
Anomalous graph			

Table 6 Probabilities of anomalous nodes

	CPU hog	Memory leak	Network latency
Probabilities of anomalous nodes	(3, 0.45)	(0, 0.43)	(4, 0.4)
	(5, 0.21)	(6, 0.31)	(0, 0.39)
	(4, 0.17)	(4, 0.08)	(2, 0.05)
	(1, 0.07)	(1, 0.07)	(6, 0.04)
	(0, 0.03)	(2, 0.04)	(1, 0.04)

$$AVG@k = \frac{1}{k} \sum_{i \leq j \leq k} PR@j \tag{6}$$

Let $R^a(i)$ be the rank of each cause, and V^a be the set of the cause in a . This paper uses $A = [cpu\ hog', memory\ leak', network\ latency']$. Furthermore, V^a is the real root cause, and R^a is the predicted root cause. We set k from 1 to 5 for $PR@k$ and calculate $AVG@5$ as the average localization accuracy.

Table 7 shows the performance of cause locating three types of faults under different anomalous graphs.

It shows that the anomalous graph based on the Direct-LINGAM algorithm achieves the best in $AVG@5$ and effectively locates root causes in all three types of anomalies. The causal propagation graph obtained by the Direct-LINGAM algorithm can most accurately show the connection among different microservices. More other root cause algorithms would like to be investigated to improve the accuracy of fault diagnosis and be suitable for more diverse time series distributions.

Table 7 Performance of cause locating

	PC	GES	ICA-LINGAM	Direct-LINGAM
CPU hog				
PR@1	0	0	0	0.333
PR@2	0.167	0	0.167	0.333
PR@3	0.167	0.167	0.333	0.5
PR@4	0.167	0.5	0.333	0.667
PR@5	0.5	0.833	0.833	0.833
AVG@5	0.2	0.3	0.333	0.533
Memory leak				
PR@1	0.167	0.167	0.167	0.333
PR@2	0.167	0.5	0.167	0.333
PR@3	0.333	0.5	0.333	0.5
PR@4	0.5	0.667	0.5	0.667
PR@5	0.5	0.833	0.833	0.833
AVG@5	0.333	0.533	0.4	0.533
Network latency				
PR@1	0	0	0.167	0.333
PR@2	0.167	0.167	0.167	0.333
PR@3	0.333	0.5	0.333	0.5
PR@4	0.333	0.667	0.333	0.667
PR@5	0.5	1	0.667	0.833
AVG@5	0.267	0.467	0.333	0.533

Values in bold indicate the best performance

5 Summary

Since multivariate time series data monitored in microservices can be occasionally and unexpectedly abnormal, it is necessary to classify the anomalies and analyze the root cause. This paper proposes an effective convolutional model with attention using a parallel structure to classify diversity anomalies and analyze the root cause from the classified anomaly data. Our model has better anomaly classification ability and achieved state-of-the-art results on a detailed set of empirical studies. For future research, there is hope to design an unsupervised model to address the challenge of label collection in microservices environments. Furthermore, we would like to explore the root cause of anomalies at a more granular level, not only at the service level but also based on the host and server. More generally, we will conduct in-depth work to increase the generalization and universality of the model in non-microservices environments in the future. This hopefully could be applied in non-microservices environments such as the Internet of Things (IoT) systems.

Acknowledgements The authors would like to thank all the staff and students of the School of Computer and Software Engineering in Xihua University for their contribution during this research process.

Author contribution Problem formulation: Xi Li, Peng Chen. The proposed algorithm: Peng Chen, Peian Wen. Computer simulations: Xi Li, Peian Wen, Juan Chen. Article preparation: Xi Li, Xuming Wen. Review: Peng Chen, Yunni Xia. All the authors read and approved the final manuscript.

Funding This research is supported by the National Natural Science Foundation under Grant No.62376043, Science and Technology Program of Sichuan Province under Grant No.2020JDRC0067, No.2023JDRC0087 and No.24NSFTD0025, and Chunhui Project of Ministry of Education of China under Grant No.Z2011085.

Data availability The data used during the current study are available from the corresponding author on reasonable request.

Code availability The codes used during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication Not applicable

Conflict of interest The authors declare no competing interests.

References

- Assaf, R., Giurgiu, I., Bagehorn, F., & Schumann, A. (2019). MTEX-CNN: Multivariate time series explanations for predictions with convolutional neural networks. In: *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 952–957. IEEE
- Chen, P., Liu, H., Xin, R., Carval, T., Zhao, J., Xia, Y., & Zhao, Z. (2022). Effectively detecting operational anomalies in large-scale IoT data infrastructures by using a GAN-based predictive model. *The Computer Journal*, *65*(11), 2909–2925.
- Chen, P., Qi, Y., & Hou, D. (2016). CauseInfer: Automated end-to-end performance diagnosis with hierarchical causality graph in cloud environment. *IEEE Transactions on Services Computing*, *12*(2), 214–230.

- Chen, P., Xia, Y., Pang, S., & Li, J. (2015). A probabilistic model for performance analysis of cloud infrastructures. *Concurrency and Computation: Practice and Experience*, 27(17), 4784–4796.
- Chickering, D. M., & Boutilier, C. (2003). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 507–554.
- Deng, A., & Hooi, B. (2020). AutoMAP: Diagnose your microservice-based web applications automatically. In: *Proceedings of The Web Conference 2020*, pp. 246–258.
- Deng, A., & Hooi, B. (2021). Graph neural network-based anomaly detection in multivariate time series. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, pp. 4027–4035.
- Di Francesco, P., Malavolta, I., & Lago, P. (2017). Research on architecting microservices: Trends, focus, and potential for industrial adoption. In: *2017 IEEE International Conference on Software Architecture (ICSA)*, pp. 21–30. IEEE
- Dongjie, W., Zhengzhang, C., Jingchao, N., Liang, T., Zheng, W., Yanjie, F., & Haifeng, C. (2023). Hierarchical graph neural networks for causal discovery and root cause localization. In: *Proceedings of 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. <https://doi.org/10.48550/arXiv.2302.01987>
- Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society*, 987–1007.
- Fan, C., Xiao, F., Zhao, Y., & Wang, J. (2018). Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data. *Applied Energy*, 211, 1123–1135.
- Fauvel, K., Lin, T., Masson, V., Fromont, É., & Termier, A. (2021). XCM: An explainable convolutional neural network for multivariate time series classification. *Mathematics*, 9(23), 3137.
- Graves, A., & Graves, A. (2012). Long short-term memory. *Supervised Sequence Labelling with Recurrent Neural Networks*, 37–45.
- Hyvärinen, A., Zhang, K., Shimizu, S., & Hoyer, P. O. (2010). Estimation of a structural vector autoregression model using non-Gaussianity. *Journal of Machine Learning Research*, 11(5), 1709–1731.
- Kiss, I., Genge, B., Haller, P., & Sebestyén, G. (2014). Data clustering-based anomaly detection in industrial control systems. In: *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 275–281. IEEE.
- Kriegel, H.-P., Kroger, P., Schubert, E., & Zimek, A. (2011). Interpreting and unifying outlier scores. In: *Proceedings of the 2011 SIAM International Conference on Data Mining (ICDM)*, pp. 13–24. SIAM.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
- Lewis, R. J. (2000). An introduction to classification and regression tree (CART) analysis. In: *Annual Meeting of the Society for Academic Emergency Medicine in San Francisco, California (Acad Emerg Med)*, vol. 14. Citeseer
- Long, J., Shelhamer, E., & Darrell, T. (2017). Fully convolutional networks for semantic segmentation.
- Mariani, L., Monni, C., Pezzè, M., Riganelli, O., & Xin, R. (2018). Localizing faults in cloud systems. In: *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, pp. 262–273. IEEE
- Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Stanford Digital Libraries Working Paper.
- Shimizu, S., Hoyer, P. O., & Hyvärinen, A. (2006). A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7, 2003–2030.
- Shimizu, S., Inazumi, T., Sogawa, Y., Hyvärinen, A., Kawahara, Y., Washio, T., Hoyer, P. O., & Bollen, K. (2011). DirectLiNGAM: A direct method for learning a linear non-gaussian structural equation model. *Journal of Machine Learning Research*, 12(2), 1225–1248.
- Shyu, M.-L., Chen, S.-C., Sarinapakorn, K., & Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. *Technical Report, Miami Univ Coral Gables FL Dept of Electrical and Computer Engineering*.
- Song, Y., Xin, R., Chen, P., Zhang, R., Chen, J., & Zhao, Z. (2023). Identifying performance anomalies in fluctuating cloud environments: A robust correlative-GNN-based explainable approach. *Future Generation Computer Systems*, 145, 77–86.
- Spirtes, P., Glymour, C. N., & Scheines, R. (2000). Causation, prediction, and search [electronic resource].
- Tuli, S., Casale, G., & Jennings, N. R. (2022). TranAD: Deep transformer networks for anomaly detection in multivariate time series data. arXiv preprint [arXiv:2201.07284](https://arxiv.org/abs/2201.07284).
- Wen, P., Yang, Z., Wu, L., Qi, S., Chen, J., & Chen, P. (2022). A novel convolutional adversarial framework for multivariate time series anomaly detection and explanation in cloud environment. *Applied Sciences*, 12(20), 10390.
- Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). CBAM: Convolutional block attention module. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19.

- Wu, L., Tordsson, J., Bogatinovski, J., Elmroth, E., & Kao, O. (2021). MicroDiag: Fine-grained performance diagnosis for microservice systems. In: *Proceedings of 2021 IEEE/ACM International Workshop on Cloud Intelligence*, pp. 31–36.
- Xin, R., Liu, H., Chen, P., & Zhao, Z. (2023). Robust and accurate performance anomaly detection and prediction for cloud applications: A novel ensemble learning-based framework. *Journal of Cloud Computing*, 12(1), 1–16.
- Xu, X., Chen, P., Xia, Y., Long, M., Peng, Q., & Long, T. (2022). MRoCO: A novel approach to structured application scheduling with a hybrid vehicular cloud-edge environment. In: *2022 IEEE International Conference on Services Computing (SCC)*, pp. 84–92. IEEE
- Yang, Z., Ying, S., Wang, B., Li, Y., Dong, B., Geng, J., & Zhang, T. (2021). A system fault diagnosis method with a reclustering algorithm. *Scientific Programming*. <https://doi.org/10.1155/2021/6617882>
- Zhang, X., Gao, Y., Lin, J., & Lu, C.-T. (2020). TapNet: Multivariate time series classification with attentional prototypical network. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 34, pp. 6845–6852.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Xi Li¹ · Peian Wen¹ · Peng Chen¹ · Juan Chen¹ · Xuming Wen¹ · Yunni Xia²

✉ Peng Chen
chenpeng@mail.xhu.edu.cn

✉ Yunni Xia
xiayunni@hotmail.com

Xi Li
lixixi@mail.xhu.edu.cn

Peian Wen
wenpeian@stu.xhu.edu.cn

Juan Chen
chenjuan@mail.xhu.edu.cn

Xuming Wen
wenxuming@mail.xhu.edu.cn

¹ School of Computer and Software Engineering, Xihua University, Jinzhou Road, Chengdu 610039, Sichuan, China

² College of Computer Science, Chongqing University, Shazheng Street, Chongqing 400044, Chongqing, China