Future Generation Computer Systems 159 (2024) 77–90

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Multi-task federated learning-based system anomaly detection and multi-classification for microservices architecture

Junfeng Hao, Peng Chen *, Juan Chen, Xi Li

School of Computer and Software Engineering, Xihua University, Chengdu 610039, China

ARTICLE INFO

Keywords:
Microservices architecture
Multi-task federated learning
Feature fusion
System anomaly detection and classification

ABSTRACT

The microservices architecture is extensively utilized in cloud-based application development, characterized by the construction of applications through a series of functionally independent, small, autonomous services. This architectural approach is renowned for its attributes such as high cohesion, availability, low coupling, and exceptional scalability. The detection of runtime system point anomalies in microservices architectures is crucial for enhancing the Quality of Service(QoS). Furthermore, identifying the classes of detected anomalies is critical in practical applications. However, given the highly dynamic nature of microservices systems as a distributed computing architecture, conducting real-time system anomaly detection on distributed independent microservices poses a challenging task. To address these challenges, we propose the System Anomaly Detection and Multi-Classification based on Multi-Task Feature Fusion Federated Learning (SADMC-MT-FF-FL) framework. Initially, we introduce a distributed learning framework based on Multi-task Federated Learning (MT-FL) to construct multi-classification anomaly detection models for each microservice. Secondly, to identify complex system anomaly patterns and features during the runtime of microservices, we develop a feature extractor based on External Attention Mechanism and Multi-channel Residual Structure (EA-MRS). Finally, we design a Local–Global Feature-based Parallel Knowledge Transfer (LGF-PKT) framework, utilizing parallel knowledge transfer to parallelize weight updates for local and global features. To validate the effectiveness of our approach, we conducted comprehensive comparative experiments on the microservices benchmark platforms Sock-Shop and Train-Ticket. The experimental results on anomaly detection for multiclassification systems demonstrate that SADMC-MT-FF-FL outperforms the best baseline method by 28.3% and 27.8% for Macro F1 and Micro F1 on Train-Ticket, and by 8.8% and 8.6% on Sock-Shop, respectively. Additionally, we conducted comparison experiments on three public datasets, SWaT, SMD, and SKAB. The F1 scores were 0.5% higher than those of the centralized methods on SMD, respectively, 6% and 2.8% higher than those of the federated learning based method on SWaT and SKAB. Source codes are available at: https://github.com/icc-lab-xhu1/SADMC-MT-FF-FL.

1. Introduction

Microservices epitomize a contemporary paradigm in software architecture, emphasizing the creation of highly maintainable and scalable software solutions. This methodology entails disassembling extensive systems into a collection of autonomous services that function independently. By nurturing independence in both development and deployment, microservices advocate loose coupling and high cohesion, thereby enhancing modularity to unprecedented levels. Consequently, these attributes confer substantial benefits in terms of maintainability, scalability, and beyond [1]. As depicted in Fig. 1, the autonomous deployment and decentralization inherent in microservices architecture align with the current imperatives of software systems to accommodate changes, exhibit elastic deployment, and achieve high scalability [2,3]. While microservices architectures offer greater flexibility and scalability compared to traditional monolithic applications, they also confront challenges inherent to distributed systems, such as network latency, service failures, and other intricacies. Consequently, enhancing the Quality of Service(QoS) [4,5] necessitates the detection and classification of anomalies in microservices architecture, effectively achievable through an anomaly detection system [6,7]. Anomaly classification serves to categorize detected anomalies.

Anomalies are abnormal events or patterns that do not conform to expected events or patterns. Anomalies can be categorized into point anomalies, context anomalies, and collective or pattern anomalies [8]. We focus on detecting and categorizing point anomalies in microservices architecture in a cloud environment. This analysis aims

* Corresponding author.
E-mail address: chenpeng@mail.xhu.edu.cn (P. Chen).

https://doi.org/10.1016/j.future.2024.05.006
Received 28 December 2023; Received in revised form 1 May 2024; Accepted 4 May 2024
Available online 10 May 2024
0167-739X/© 2024 Elsevier B.V. All rights reserved.

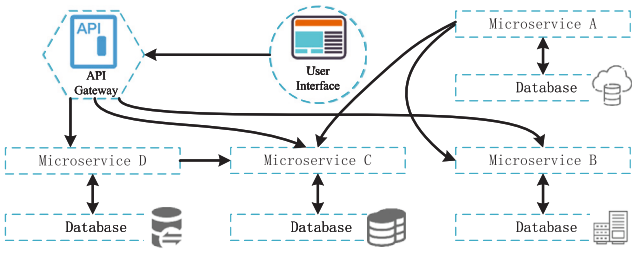


Fig. 1. Microservices Application Framework. Microservice is a distributed system structure that can divide an application into a group of small service units. These service units cooperate with each other through lightweight communication mechanisms and can be independently developed, deployed, and scaled.

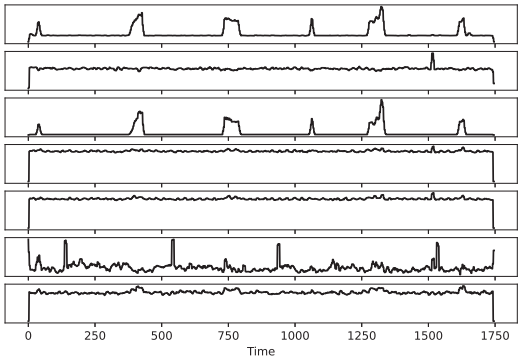


Fig. 2. Seven monitoring metrics under a typical microservice. The horizontal coordinate represents the monitored time period, and the vertical coordinates of the seven KPIs represent Payment's network latency with respect to other microservices.

to enhance the system's overall quality of service. In a microservices architecture, detecting and classifying runtime system point anomalies involves analyzing system monitoring performance indicators in the deployed cloud environment, including CPU utilization and network latency. In this context, system point anomalies refer to factors such as CPU hogging, network delay, memory leaks, etc [9,10], as opposed to exceptions within individual microservices, thus emphasizing a more universal perspective. Fig. 2 illustrates seven monitoring metrics of a typical microservice.

For instance, when a user places an order from Sock-Shop, the Order service connects to Goods and Inventory services for price and stock information. However, if there is a network issue, occurred in Order service, not only Order service but also Goods and Inventory would be impacted, potentially leading to service unavailability. It is very difficult for traditional centralized methods to deal with such anomaly propagation in a typical distributed microservice architecture. In general, anomaly detection and classification methods for microservices architecture deployed in distributed environments encounter several challenges:

- The traditional centralized learning model is incongruent with the distributed architecture of microservices.
- The microservices architecture exhibits highly dynamic and complex dependencies between independent microservices [11,12].
- Most methods focus solely on detecting whether an anomaly has occurred, lacking the capability to identify the specific type of anomaly. Consequently, operators face challenges in promptly determining the root cause of the anomaly.

The central challenge we aim to address is the detection and classification of system point anomalies within the distributed architecture of microservices, enabling dynamic system fault identification, analysis, supporting subsequent fault handling, and ultimately enhancing the QoS [13–15]. Consequently, a high-performance anomaly detection and

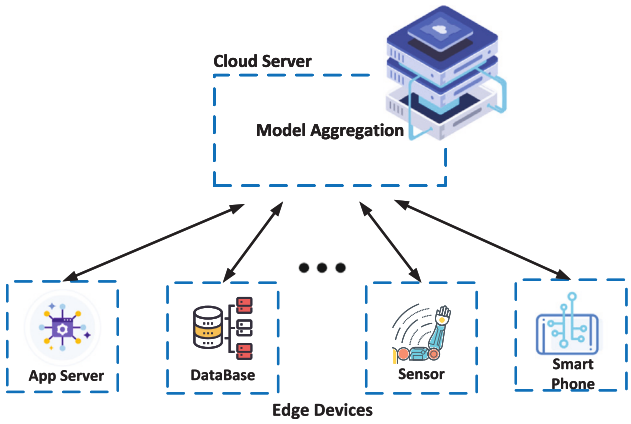


Fig. 3. Typical Architecture for Federated Learning. In the client/server architecture of Federated Learning, the server is responsible for coordinating and managing the training of the global model, while the client is responsible for providing local data for model training.

classification method tailored for microservices distributed architecture is required. Federation Learning (FL) [16] emerges as an effective approach for collaborative computing using distributed resources, where multiple distributed nodes collaborate to train a model or are jointly prepared by multiple distributed computing nodes while preserving the distribution of raw data without the need to centralize it in a single server or data center [17]. Fig. 3 depicts a typical architecture for federated learning. In this model, raw data or data generated after secure processing based on the raw data serves as the training data. Addressing fundamental issues of privacy, ownership, and data locality [18,19], federated learning facilitates multiple participants in training models without sharing raw data. Additionally, it enhances computational resource utilization. This is particularly relevant as microservices architectures commonly rely on distributed computing to manage requests and coordinate services among multiple tasks or services. Within the microservices architecture, where applications are divided into multiple independent services, and different microservices execute distinct tasks, we employ multi-task learning methods to optimize overall system performance. Multi-task Learning (MTL) concurrently considers multiple related tasks, leveraging the inherent relationships between tasks to enhance the generalization ability of single-task learning [20]. The foundational premise of multi-task learning is that the knowledge applied to solve a new problem is typically influenced by experiences gained from solving related existing problems. To surmount the challenges outlined above, we propose the System Anomaly Detection and Multi-Classification based on Multi-Task Feature Fusion Federated Learning (SADMC-MT-FF-FL) framework. The key concepts include:**(1)** Proposing and implementing a multi-task federated learning framework to address distributed challenges within microservices. **(2)** Acknowledging the heterogeneous and diverse nature of exceptions in microservice architectures, we incorporate External Attention Mechanism and Multi-channel Residual Structure (EA-MRS) to capture these features and classify anomalies. **(3)** Leveraging the similarities between each microservice for feature fusion and implementing model updates from global to local.

Our primary contributions are succinctly summarized as follows:

- ***Proposal of a Systems Anomaly Detection and Classification Framework.*** We introduce a framework for anomaly detection and classification in systems based on multi-task federated learning within the context of microservices architecture.
- ***Design of EA-MRS Feature Extractor.*** We devise the EA-MRS feature extractor to proficiently identify intricate abnormal patterns and features within microservices.

- ***Development of the Local–Global Feature-based Parallel Knowledge Transfer (LGF-PKT) Framework.*** The LGF-PKT framework is designed to effectively enhance overall performance by updating the model based on feature similarity.
- ***Empirical Validation.*** Experimental results reveal the superiority of SADMC-MT-FF-FL, presenting performance gains over the best baseline method by 28.3% and 27.8% for Macro F1 and Micro F1 on Train-Ticket, and by 8.8% and 8.6% on Sock-Shop, respectively. Comparative experiments on public datasets SWaT, SMD, and SKAB demonstrate F1 scores that are 5.7% and 0.5% higher than the centralized methods on SWaT and SMD, respectively, and 2.8% higher than the multi-task baseline method on SKAB.

The rest of the paper is organized as follows. Section 2 briefly introduces algorithms related to time series anomaly detection and classification; Section 3 describes our model architecture and provides detailed information on each component. Section 4 presents comprehensive comparative results and experimental analysis. Section 5 summarizes the primary contributions of this study and outlines directions for future work.

2. Related work

As previously mentioned, the core of anomaly detection and classification within microservices architecture involves the application of time-series anomaly detection techniques to analyze the time-series data collected by system monitoring for each performance metric. The objective is to identify anomalies and categorize them. In this section, we commence by introducing deep learning-based methods and models for time-series anomaly detection and classification. Subsequently, we delve into various federated learning frameworks and explore the application of federated learning to time-series anomaly detection.

2.1. Deep learning based methods

In the realm of time series data anomaly detection, Principal Component Analysis(PCA) serves to diminish data dimensionality, eliminate superfluous information, and discern anomalous data features marked by scarcity. K-Nearest Neighbors (KNN) [21] find application in time series anomaly detection for pinpointing anomalies or outliers within the temporal data. For instance, KNN predicts the future time point value by considering information from the K closest data points in the historical time series. The Stochastic Gradient Descent classifier (SGDClassifier) [22] is a machine learning technique aimed at minimizing the loss function iteratively to classify new data points. In the context of time series anomaly detection and classification, SGDClassifier leverages temporal features for enhanced training. Traditional methods prove inadequate for addressing the intricacies of dynamic cloud computing systems, prompting the adoption of deep learning approaches that harness the potent learning capabilities of neural networks. An encoder–decoder structure based on Long Short-Term Memory (LSTM) [23] networks proves instrumental, capitalizing on the sequential modeling proficiency inherent in LSTM networks. This approach adeptly captures temporal relationships within multi-sensor data, thereby elevating the accuracy of anomaly identification. The Deep Autoencoding Gaussian Mixture Model (DAGMM) [24] fuses autoencoders' encoding capabilities for low-dimensional data representation with the modeling capabilities of Gaussian mixture models. By comparing input data reconstruction errors and the distribution of data in latent space, DAGMM discerns anomalies effectively. The CGNN-MHSA-AR [25] introduces a novel approach for detecting performance anomalies in fluctuating cloud environments. This method employs Graph Neural Networks(GNNs) and correlation analysis in an interpretable framework. OmniAnomaly [26] utilizes Stochastic Recurrent Neural Networks(SRNN) for anomaly detection in multivariate time series. SRNN, an extension of Recurrent Neural Networks(RNN), introduces stochasticity to better capture uncertainties and complexities within the data. The Unsupervised Anomaly Detection (USAD) [27] model employs an autoencoder with two decoders and an adversarial training framework to classify normal and anomalous data. TranAD [28], a model based on deep transformer networks, employs attention-based sequential encoders for rapid inference regarding temporal trends. Graph-based Deep Anomaly Detection(GDN) [29] combines structural learning methods with graph neural networks, additionally utilizing attention weights to provide explanations for detected anomalies. The Predictive Wasserstein Generative Adversarial Network with Gradient Penalty (PW-GAN-GP) [30] adopts both Wasserstein Distance and Gradient Penalty, making the adversarial training more stable and helping the generator's output to more closely resemble the real data. TimesNet [31] decomposes complex time variations into intra-period and inter-week variations according to the multi-period nature of time series. In order to solve the limitation of the representation ability of 1-D time series, the analysis of time variation is extended to 2-D space, and the 1-D time series is converted into a set of 2-D tensors based on multiple periods.

The stated anomaly detection algorithms [23–30] are designed to enhance detection accuracy through advanced deep learning methods in the context of multivariate anomaly detection. Despite their efficacy, algorithms are typically centralized for anomaly detection and classification. This centralized approach may not align optimally with the nature of microservices architecture, which inherently embodies a distributed structure. In the microservices environment, where tasks are distributed across various computing nodes, anomaly detection and classification involve multiple tasks. This mirrors the diverse applications running distinct tasks on dispersed computing nodes within a mobile computing framework. Employing solely anomaly detection and classification within the microservices application framework may not effectively leverage the computational capabilities inherent in the distributed microservices architecture. Furthermore, the training process associated with centralized approaches tends to be bandwidth-intensive and raises notable privacy concerns. Given the distributed nature of microservices architecture, exploring decentralized or distributed approaches for anomaly detection and classification becomes crucial to align with the architectural principles and to address potential privacy implications associated with centralized training processes.

2.2. Federated learning based methods

Microservices architectures, characterized by their high dynamism, pose challenges for traditional centralized learning methods. In contrast, federated learning is well-suited for deployment on microservices architectures, enhancing performance incrementally through model updates. FedAvg [32], a key approach, computes average weights from all node models and redistributes these weights to each node within the federated learning system. The analysis of FedAvg in non-independent identically distributed data (Non-IID Data) convergence involves collaborative model learning by multiple nodes [33].

FedTL [34,35] introduces a migration learning technique to facilitate knowledge transfer among diverse nodes, thereby enhancing system accuracy. Notably, Yang et al. developed FedSteg [34], a FedTL framework tailored for secure image privacy analysis. Differing from FedAvg and FedTL, FedKD [36] calculates the average weights of all nodes from all teachers and transmits each teacher's knowledge to corresponding students through knowledge distillation (KD). A knowledge migration training algorithm [37] is employed to train a small convolutional neural network (CNN), transferring its knowledge to a prominent server-side CNN.

Recognizing the limitations of employing a single global model in existing federated learning approaches, which aggregates gradients irrespective of differences in user data distributions, a multicenter federated learning model, FeSEM [38], was proposed. This model assigns users' gradients to different global models (centers) to better capture

the heterogeneity of data distributions among users. MOON [39] uses the principle of contrastive learning to make the feature representation of the local model closer to the feature representation of the global model to reduce the impact of model drift, thereby reducing the impact of data heterogeneity. Scaffold [40] can effectively alleviate the problems of reduced convergence speed and accuracy caused by data heterogeneity, reduce communication volume and communication frequency, improve the efficiency and scalability of federated learning, and ensure data privacy of clients. Heterogeneity is a challenge in federated learning. Chen et al. [41] used elastic aggregation to overcome the gradient dissimilarity between various clients, thereby improving the generalization ability of the model.

In recent years, federated learning has gained prominence as a compelling alternative to centralized methods. Rather than aggregating extensive amounts and types of data to a central location, federated learning distributes the global model training process, allowing data from each participating distributed node to be used locally for training a model in situ [42]. Initially, the simple averaging of model weights was the most commonly used method [43]. And microservice-based deployments exhibit greater dynamism and volatility compared with traditional applications, presenting challenges in monitoring and performance modeling. Additionally, exceptions in microservices architecture may vary in definitions and diversities. Simultaneously, microservices architectures necessitate efficient anomaly detection and classification, with rapid detection and response being crucial for ensuring system stability and reliability. Drawing inspiration from the aforementioned federated learning approach, novel ideas and directions emerge. Consequently, we propose the SADMC-MT-FF-FL framework. This section initiates by providing an overview of the comprehensive architecture of the SADMC-MT-FF-FL framework. Subsequently, the intricacies of each component are expounded in a sequential manner.

Algorithm 1 SADMC-MT-FF-FL

Input: Local system monitoring of real-time data at each node: $D_t = \{X^t, y^t\}$;
Output: Anomaly detection and classification results for each node: $\hat{y}^t$

initial $EA-MRS_t$ and $LGF-PKT$;
procedure Local Train
  for each Node t **in parallel do** ▷ Train to get the set of weights for each node's *Local* model
    $W_{Local_t} = EA-MRS_{Local_t}\left(D_t\right)$
  end for
  Update $\left(W_{Local_1}, W_{Local_2}, ..., W_{Local_t}\right)$
end procedure
procedure Local to Global Model Updates
  $Node_{update}$ receives the weights of all other nodes
  for each Node t **in parallel do** ▷ Parallelized knowledge migration for each node to complete weight updates
    $W_{Transfer_t} = LGF-PKT\left(W_{Local_t}\right)$
  end for
end procedure
procedure Transfer Train
  Each node receives $W_{Transfer_t}$
  for each Node t **in parallel do** ▷ Start training to get the predicted values to get the distribution probability by softmax and then anomaly detection and classification.
    $\hat{y} = EA-MRS_{Transfer_t}\left(D_t\right)$
    $A_t = SoftMax(\hat{y})$
    $\hat{y}^t = argMax(A_t)$
  end for
end procedure

3. Methodology

3.1. Model overview

3.1.1. Overall architecture

In the SADMC-MT-FF-FL framework, we consider the existence of t microservices distributed computing nodes, where $0 < t \leq I_con$ (I_con denotes the number of connected distributed computing nodes). Each microservice node's local system monitoring real-time data, denoted as D_t (not shared with other computing nodes), is utilized as input to the distributed computing system. This data adheres to the characteristics of distributed data distribution. The Weight Distance Calculation scheme making is independent for each distributed computing node, and each microservice node outputs the results of local anomalies. The detailed system structure is depicted in Fig. 4.

3.1.2. Algorithm and complexity analysis

To elaborate on the operational flow, The details are given in Algorithm 1. Each active microservice distributed node initiates the process by sending local real-time system monitoring metrics as input to the feature extractor. This feature extractor is based on the EA-MRS and comprises both a *Local* model and a *Transfer* model. In the commencement of the federated learning cycle, the first feature extraction is performed and transmitted by the *Local* model. Subsequently, after completing the first weight update, all subsequent features are extracted and sent by the *Transfer* model. Once the *Local* model at each node concludes the first round of training, the feature weights are transmitted to the node responsible for weight update to execute the Weight Distance Calculation scheme. This node calculates a similarity index table for each node based on similarity and triggers the LGF-PKT framework. This framework transmits back the top-ranked weights from each node to perform the model update.

The analysis of time complexity and space complexity in anomaly detection within a microservices architecture is crucial as it directly affects system performance and scalability.

In this experiment, time complexity is directly proportional to the data size, and the microservices architecture's high concurrency, enabling the simultaneous execution of multiple microservices, enhances the overall performance of our system algorithms within this architecture. Model training, identified as the phase with the largest time overhead, involves convolution operations. The time complexity of each one-dimensional convolution operation is $O(K \times C)$, where K is the size of the convolution kernel, C is the number of convolution kernels. Since three one-dimensional convolutional layers are employed, the total time complexity is $O(K \times C)$. The time complexity of the attention mechanism is $O(l^2 \times d)$, where l is the length of the sequence, and d is the feature dimension of each position. In the external attention mechanism, each position performs an attention computation with other positions in the sequence. For a single node, the overall time complexity is the sum of the complexities of model training and attention mechanisms. For a computational node performing weight updates, the time complexity is $O(I_{con})$.

Data in a microservices environment predominantly consists of time series data. Consequently, space complexity depends on the dimensionality of the data, sampling rate, storage format, and time span of the data. The decentralized nature of data in a distributed computing environment enables the system to make full use of its high computational performance. Consequently, the space complexity is $O(D_t)$ optimized for efficient anomaly detection within a microservices architecture.

3.2. Multi-task federated learning framework

In the microservices architecture, each microservice operates independently, and the conventional federated learning approach, which aggregates all local models into a single global model, is suboptimal.

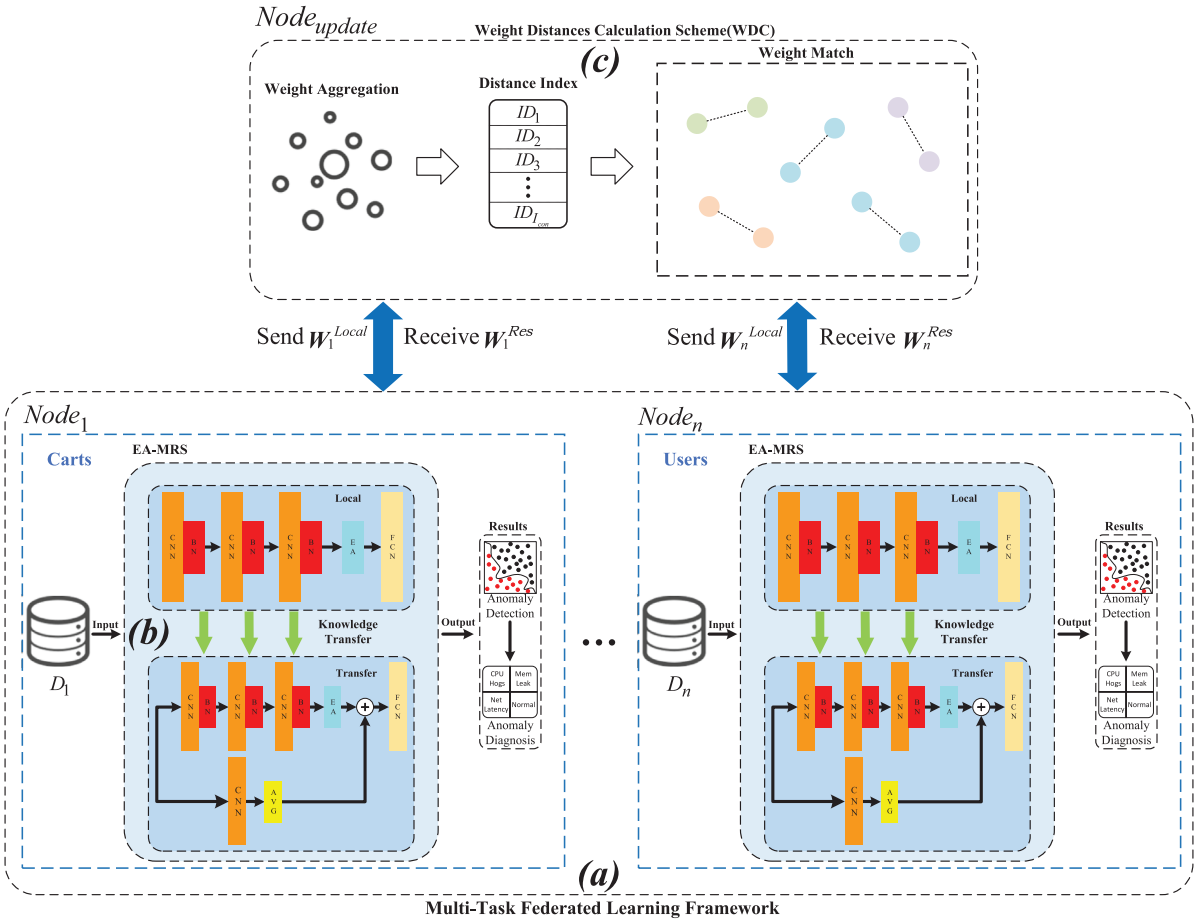


Fig. 4. SADMC-MT-FF-FL overall architecture. (**a**) denotes the Multi-Task Federated Learning framework, (**b**) denotes feature extractor based on EA-MRS (External Attention Mechanism and Multi-channel Residual Structure), and (**c**) denotes the LGF-PKT (Local–Global Feature-based Parallel Knowledge Transfer) framework.

This is due to the highly distributed nature of microservices, each running different tasks and exhibiting data heterogeneity. To address this, we employ multi-task federated learning for weight aggregation and updating. Each microservice node deploys EA-MRS feature extractors for both local and global feature extraction. Following the completion of model weight updating, anomaly classification occurs. Each model corresponds to a specific anomaly detection or classification task. Multi-task federated learning is utilized to address multiple microservices' anomaly detection and classification needs. Simultaneously, the approach leverages the similarity and differences between tasks to execute local–global model updating. The *Transfer* model receives the returned feature weights and promptly begins training. It outputs data prediction label values, where the prediction value represents the distribution probability of the data. Subsequently, the system determines whether the data is anomalous (for anomaly detection tasks) or falls into a specific anomaly category (for anomaly classification tasks) based on the return value of the Softmax (for anomaly detection tasks) or the assigned anomaly category (for anomaly classification tasks). This tailored approach accommodates the diverse tasks and data characteristics within the microservices architecture.

In accordance with the standard setup of federated learning [44], we assume the presence of t microservice distributed computing nodes, where $0 < t \leq I_{con}(I_{con}$ denotes the number of connected distributed nodes), (X^t, y^t) denotes the training data of the tth node, X^t is generated by the local system monitoring system in real time, and y^t denotes the truth value label or the output vector. The objective of multi-task federated learning is to minimize the cross-entropy on each distributed computing node. In the context of anomaly detection, this task can be viewed as a binary classification problem, where the model's goal is to categorize data as either "normal" or "abnormal". Anomaly classification is treated as a multi-classification task. In this case, the model not only needs to identify the presence of abnormality but also discern the specific type of abnormality.

For both anomaly detection and classification tasks, cross-entropy is employed as the loss function L to be minimized. The cross-entropy loss function is defined as follows:

$$L\left(y^t, \hat{y}^t\right) = -\frac{1}{m}\sum_{i=1}^{m} y^t log\left(\hat{y}^t\right) \tag{1}$$

where m is the number of input vectors, where $y^t, \hat{y}^t$ are the truth label and prediction probability of the tth node, respectively. The node performing the update task receives the trained feature weights from the local models of other computing nodes, and then performs the global weight update and sends back the feature weights to help further train the local model.

3.3. Feature extractor based on external attention mechanism and multi-channel residual structure (EA-MRS)

To effectively identify complex system anomaly patterns and features during the runtime of microservices and to promptly detect and respond to anomalies critical for the stability and reliability of the system, we introduce the EA-MRS. Deployment of EA-MRS on each computational node consists of a *Transfer* model and a *Local* model on the basis of a 1D-CNN. The *Local* model is tasked with local feature extraction, while the *Transfer* model undergoes further training through weight updates. Initially, the *Local* model is trained using the data from the current node, and the weights associated with the features of that

node are transmitted to the weight aggregation node. This triggers the weights distance computation scheme, updating the weights. Subsequently, parallel knowledge transfer occurs, with the matching feature weights sent back to the *Transfer* model for additional training. In the *Transfer* model, the Multi-channel Residual Structure is introduced to enhance its capabilities. Techniques from the computer vision domain are applied to time-series feature extraction. The External Attention Mechanism(EA) enables the model to selectively focus on specific parts of the input, rather than treating all information uniformly. The Multi-channel Residual Structure aims to preserve specific local features, thereby enhancing the *Transfer* model's performance in microservices system anomaly detection and classification. To address privacy concerns and mitigate data leakage, we refrain from globally updating the input and output of hidden layers. Instead, we perform local operations on the input and output of intermediate layers, only updating the feature weights of each layer. This approach effectively safeguards against data privacy leakage in microservices system anomaly detection and classification.

3.3.1. Local model

The *Local* comprises three convolutional layers, an adaptive maximum pooling layer, a fully connected layer, and the External Attention Mechanism. Each convolutional layer is composed of a one-dimensional CNN module, a batch normalization module, and a ReLU activation function, defined as follows:

$$F_{con} = f_{relu}\left(f_{bn}\left(W_{conv} \otimes x + b_{conv}\right)\right) \tag{2}$$

Where W_{conv} and b_{conv} are the weight and bias matrices of the CNN, respectively. $\otimes$ denote the convolution operation. f_{bn} and f_{relu} denote the batch normalization layer and the ReLU activation function, respectively. $F_{con} \in R^{N\times d}$ denotes the output feature result.

Let $X_{bn} = x_1, x_2, \ldots, x_m$ be denoted as the input to the batch normalization layer, where x_i and m denote the ith instance and batch size, respectively:

$$f_{bn}\left(X_{bn}\right) = \left(\alpha\frac{x_1-\mu}{\delta-\epsilon} + \beta, \alpha\frac{x_2-\mu}{\delta-\epsilon} + \beta, \ldots, \alpha\frac{x_m-\mu}{\delta-\epsilon} + \beta\right) \tag{3}$$

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4}$$

$$\delta = \sqrt{\sum_{i=1}^{m}\left(x_i - \mu\right)^2} \tag{5}$$

Where $\alpha \in R+$ and $\beta \in R$ are the parameters to be learned during training.

External Attention [45] is a mechanism in machine learning models that improves performance on a given task by selectively focusing on certain parts of the input data or features. This mechanism allows the model to focus on relevant information while ignoring irrelevant or redundant information. EA module is a lightweight external attention approach, which is an architecture containing only two linear layers only, mainly applied in the field of machine vision, and its core idea is mainly to enhance the expressive power of the model by combining internal and external information sources. We move it to our approach by combining the information from other nodes of the microservices architecture with the internal information from local nodes to enhance the expressive power of the model. We add this structure of residuals after updating the weights is able to help the *Transfer* model to retain a certain amount of feature information of the local computational nodes, and then the feature information learned by weighting the new weights can get a better performance. An attention graph is computed by calculating the affinity between the self-query vector and the self-key vector, and then a new feature graph is generated by weighting the self-value vector with that attention graph. The formula for the External Attention Mechanism can be expressed as follows:

$$A = softmax\left(FF^T\right) \tag{6}$$

$$F_{out} = AF \tag{7}$$

Where $F \in R^{N\times d}$ denotes feature map, where N is the vector length and d is the feature dimension.

3.3.2. Transfer model

The *Transfer* model is composed of three convolutional layers, an adaptive mean pooling layer, the External Attention Mechanism, and the Multi-channel Residual Structure. Building on this inspiration [46], we design the Multi-channel Residual Structure that includes a one-dimensional convolutional layer and an average pooling layer. This structure is designed to preserve specific local features. The exact calculations are as follows:

$$F_{con} = f_{relu}\left(f_{bn}\left(W_{conv} \otimes x + b_{conv}\right)\right) \tag{8}$$

$$F_{ea} = softmax\left(F_{con}F_{con}^T\right)F_{con} \tag{9}$$

$$F_{avg} = f_{Avgpooling}\left(W_{conv} \otimes x + b_{conv}\right) \tag{10}$$

$$O = concatenate(F_{ea}, F_{avg}) \tag{11}$$

$$y_{pre} = Softmax(O) \tag{12}$$

$$y^l = argMax(y_{pre}) \tag{13}$$

Where $F_{con} \in R^{N\times d}, F_{avg} \in R^{N\times d}$ denote the convolutional layer results and the Multi-channel Residual Structure results respectively. $F_{ea} \in R^{N\times d}$ denotes the feature map obtained through the External Attention Mechanism. O represents the final output that undergoes the Softmax function to obtain the predicted value. Subsequently, the argMax function is applied to derive the detection or classification result.

3.4. Local–global feature-based parallel knowledge transfer(LGF-PKT)

Algorithm 2 Local–Global Feature-based Parallel Knowledge Transfer Framework

Input: The set of feature weights for each node $W_{I_{con}}$;
Output: The set of most similar weights for each weight W_t^L

$Node_{update}$ receives all local weights $W_{I_{con}}$
procedure Weight Distance Calculation Scheme(WDC)
  for each Node t **in parallel do**
    $d = WDC\left(w_t\right)$ ▷ Calculate the similarity of the $i-$th node with other nodes
    ID $= sort(d)$ ▷ The **ID** list is obtained by sorting in descending order according to similarity
    index $= topK($**ID**$)$ ▷ Get the most similar node index
  end for
end procedure
procedure Parallel Knowledge Transfer(PKT)
  for each Node t **in parallel do**
    $W_t^{Transfer} \leftarrow \mathbf{W}^L \leftarrow \mathbf{W}_{\mathbf{I_{index}}}$
  end for
end procedure

The framework leverages both local and global features to update the knowledge of individual nodes through Weight Distance Calculation scheme. This approach aims to enhance the performance of local model anomaly detection and classification, adapting to the anomaly diversity and dynamics inherent in distributed architectures within the microservices environment. The gradual improvement in overall performance is achieved by incorporating insights from both local and global contexts. Local features typically pertain to those collected within a single microservice instance, while global features encompass those that span multiple microservice instances or the entire system. The parallelization approach employed in the framework proves beneficial in handling large volumes of data and effectively capitalizes on the heterogeneity and data diversity inherent in the distributed architecture of microservices.

In the context of federated learning frameworks, existing approaches often utilize average weight scheme to aggregate weights and update the model. However, these approaches ignore the differences between data distributions on individual nodes, a factor particularly noticeable in mutually independent microservices systems. Moreover, anomalies in microservices systems exhibit varied definitions and diversities. Different services may report distinct types of exceptions, and the definition of exceptions can vary depending on the business logic of the system. To address these challenges, we propose a parallel knowledge migration framework based on local–global features for local-to-global model updating, utilizing the Weights Distance Calculation Scheme. This approach acknowledges the nuances in data distributions and anomaly definitions across microservices, thereby enhancing the effectiveness of the federated learning model.

3.4.1. Weight distance calculation scheme

Let FL_{iter} denote the maximum number of cycles of federated learning, set to 4 in this experiment. As observed through experimentation, the overall system reaches stability after three federated learning cycles. Let $W_i^{Local,k}$ and $W_i^{Transfer,k}$ be the kth previously trained weight uploaded to the server and the weight sent from the server after weight matching, $k = 1, 2, \ldots, FL_{iter}$. The weights of their hidden layers are denoted by $W_i^{Local_{hidden},k} \subset W_i^{Local,k}, W_i^{Transfer_{hidden},k} \subset W_i^{Transfer,k}$ respectively.

Specifically, $W_i^{Local_{hidden},k}$ consists of Conv1, Conv2 and Conv3, with weights $W_i^{Local_1,k}$, $W_i^{Local_2,k}$ and $W_i^{Local_3,k}$, respectively, where $W_i^{Local_{hidden},k} = W_i^{Local_1,k}, W_i^{Local_2,k}, W_i^{Local_3,k}$. In the kth federated learning phase, nodes $T_i, i = 1, 2, \ldots, I_{con}$ upload $W_i^{Local_{hidden},k}$ to the weight distance calculation node. The node stores the uploaded weights in the set of weights defined in Eq. (14).

$$\mathbf{W} = \left[W_1^{Local_{hidden},k}, W_2^{Local_{hidden},k}, \ldots, W_{I_{con}}^{Local_{hidden},k}\right] \tag{14}$$

The server then computes the set of weight distances d,d defined by **W**:

$$d = \begin{bmatrix} d_1 \\ d_2 \\ \cdots \\ d_{I_{con}} \end{bmatrix} = \begin{bmatrix} d_{1,2} & \cdots & d_{1,I_{con}} \\ d_{2,1} & \cdots & d_{2,I_{con}} \\ \cdots & \cdots & \cdots \\ d_{I_{con},1} & \cdots & d_{I_{con},I_{con}-1} \end{bmatrix} \tag{15}$$

Where $d_{i,j}\left(i, j \in 1, \ldots, I_{con}, i \neq j\right)$ is the weight distance between $W_i^{Local_{hidden},k}$ and $W_j^{Local_{hidden},k}$ obtained by the combination of Euclidean and Cosine distances, as defined in Eq. (18):

$$d_{i,j}^{eu} = \sqrt{\sum_{n=1}^{3} \left\| W_i^{Local_n,k} - W_j^{Local_n,k} \right\|^2} \tag{16}$$

$$d_{i,j}^{cos} = \frac{\sum_{n=1}^{3} W_i^{Local_{hidden},k} \bullet W_i^{Local_{hidden},k}}{\sqrt{\sum_{n=1}^{3} W_i^{Local_{hidden},k^2}} \bullet \sqrt{\sum_{n=1}^{3} W_i^{Local_{hidden},k^2}}} \tag{17}$$

$$d_{i,j} = \frac{d_{i,j}^{eu}}{d_{i,j}^{cos}} \tag{18}$$

Our proposed Weight Distance Calculation Scheme differs from the updating method of clustering algorithms, such as K-Means. Unlike clustering algorithms that require setting the number of clusters, WDC is an iterative updating method. It does not require specifying the number of clusters beforehand. Instead, it obtains a similarity index table through a distance combining formula, and this index table is continually updated as the federated learning cycle progresses. This feature renders the Weight Distance Calculation Scheme well-suited for the microservices architecture. It effectively enhances the overall generalization ability and robustness of the local model without the need for a predetermined cluster count.

Table 1
Details of all datasets.

	SS	TT	SWaT	SMD	SKAB
Train	22 576	27 140	7000	70 843	12 712
Test	9680	11 632	3000	30 361	5448
Abnormality Rate	12.2%	27.8%	29.2%	4.21%	35.1%

3.4.2. Parallel knowledge transfer

Parallelized knowledge transfer employs a distributed structure to gather and migrate feature weights, leveraging previously accumulated knowledge. By assimilating feature information from other nodes, it addresses the limitations of the local model's learning capacity, consequently improving the learning efficiency and generalization capability of the local model.

The node responsible for model updating collects the feature weight set of all nodes. Subsequently, through the Weight Distance Calculation Scheme, it obtains the weight distance set d of each node, d is sorted in descending order to generate the **ID** list. The **ID** list represents the ranked list of nodes with feature weights most similar to those of the current node. The higher the ranking, the greater the degree of similarity. Each iteration involves returning to the highest-ranked index using the index to retrieve the most similar feature weights. **ID** is defined in Eq. (19).

$$\mathbf{ID} = \left[ID_1, ID_2, \ldots, ID_{I_{con}}\right] \tag{19}$$

Where ID_i is the index of the T_i distance.

According to the **ID**, the set of weights can be obtained based on the union equation $\mathbf{W}^L$ from **W**, $\mathbf{W}^L$ is defined in Eq. (20):

$$\mathbf{W}^L = \left[W_1^{L,k}, W_2^{L,k}, \ldots, W_{I_{con}}^{L,k}\right] \tag{20}$$

$$= \left[\mathbf{W}\left(ID_1\right), \mathbf{W}\left(ID_2\right), \ldots, \mathbf{W}\left(ID_{I_{con}}\right)\right] \tag{21}$$

Where $W_i^{L,k}$ are the weights matching T_i at the kth federated learning cycle.

Upon completion of the Weight Distance Calculation Scheme, the updating node distributes the corresponding weights to all nodes. When the weights distance calculation scheme is completed, the update node distributes the corresponding weights to all the nodes. Once the I_i node receives $W_i^{L,k}$ from the update node, T_i loads these weights into $W_i^{Transfer_{hidden},k}$ at the beginning of the next federated learning cycle, as defined in Eq. (21).

$$W_i^{Transfer_{hidden},k+1} \leftarrow W_i^{L,k} \tag{22}$$

The feature weights extracted from the *Local* model are migrated to the *Transfer* model to execute a local–global parallel weight update. In particular, we initiate the process by uploading the feature weights from the *Local* model of all connected nodes to the node where the weight distance is calculated. Following the upload of feature weights from all connected nodes, the Weight Distance Calculation Scheme is activated. The matched feature weights are subsequently communicated back to the nodes for the local–global model update.

$$W_j^{Transfer} \leftarrow W_i^{Local} \tag{23}$$

4. Experiments

4.1. Datasets

We selected two microservice benchmarks, Sock-Shop and Train-Ticket, along with three public datasets (SMD, Swat, and SKAB) for evaluation. The anomaly rates and feature dimensions for these five datasets are presented in Table 1. The choice of these public datasets was deliberate as they differ in distribution, dimension, and proportion

Table 2
Sock-Shop dataset details.

	carts	catalogue	frontend	orders	payment	shipping	user
Dimension	35	35	35	35	36	35	36
Train	3276	3229	3448	3273	3080	3096	3174
Test	1405	1384	1479	1403	1320	1328	1361
Anomaly Type	3	3	3	3	3	3	3

of anomalies. Specifically, SMD exhibits a lower anomaly rate and a more discrete distribution of anomalies, Swat has a higher feature dimension, and SKAB has fewer features. This diversity underscores the heterogeneous nature of the selected datasets, effectively representing the varied definitions and diversities of anomalies in microservices systems. To simulate real microservices architecture environments, we injected three anomaly types across seven microservices. These anomalies are pervasive, reflecting the diverse nature of anomalies in real systems.

Sock-Shop(SS).[1] In our experiment, we deployed the Sock-Shop microservices benchmarking. To simulate a real running application, we injected three common exceptions: CPU Hog, Memory Leak, and Network Latency. We use the tool Pumba to simulate network failures and stress test Docker containers for anomaly injection. For CPU Hog, we use CPU resources to consume each service; for Memory Leak, we allocate memory continuously for each service; and for network latency, we enable flow control to delay network packets. Each exception lasts from 1 to 5 min, and we repeat each exception at least five times after the application has been running normally for 10 to 30 min. We collect data in real time every 5 s as configured by Prometheus, collecting both service-level and resource-level data. At the service level, we collect latency for each service. At the resource level, we collect container resource-related metrics, including CPU utilization, memory utilization, disk reads and writes, and network received and transmitted bytes. Kubernetes' elastic horizontal scaling mechanism focuses on automatically adjusting the number of application replicas based on the workload and container load. Our data can be downloaded through https://surfdrive.surf.nl/files/index.php/s/FjmBjVQnDRWqDmV. Detailed information of the collected data is shown in Table 2.

Train-Ticket(TT).[2] We continue to verify the generalization capability of SADMC-MT-FF-FL using Train-Ticket provided by Eadro [47]. Based on the monitoring data provided by Eadro, we selected six microservice datasets including food, food-map, assurance, order-other, train, inside payment. Eadro injected three common faults into Train-Ticket: CPU utilization, Network latency, and Network loss.

Server Machine Dataset(SMD).[3] It is a dataset that simulates a server machine and contains performance and anomaly data from server hardware and software. It typically includes time-series data on metrics such as CPU usage, memory usage, network traffic, disk usage, etc. SMD represents performance monitoring data in a data center or server environment. SMD has a low percentage of anomalies and a discrete distribution of anomalies, which validates the robustness and generalization of the modeling approach.

Secure Water Treatment(SWaT).[4] It is a dataset of Internet of Things (IoT) devices and industrial control systems (ICS) that model the environment of a water treatment plant. It includes data from sensors, actuators, and controllers of industrial control systems, as well as information related to the water treatment process.SWaT represents industrial control system data in the critical infrastructure domain, and SWaT also has the highest dimensionality of characterization and more continuous occurrence of anomalies than any other dataset.

[1] https://github.com/microservices-demo/microservices-demo
[2] https://github.com/FudanSELab/train-ticket
[3] https://github.com/NetManAIOps/OmniAnomaly
[4] https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_swat/

Table 3
Comparison of experimental setup and baseline methods.

Experiments		Methods	Evaluation Metrics
Ex1[4.4]	Anomaly Detection	LSTM-AD [23], DAGMM [24], OmniAnomaly [26], USAD [27], GDN [29], TranAD [28], TimesNet [31], Transfer FedAVG [32], FedTL [34], FedKD [36], MOON [39], Scaffold [40], Elastic [41], SADMC-MT-FF-FL	F1, AUC, MCC
Ex2[4.5]	Anomaly Classification	SGDClassifier [22], KNN [21], CNN, MLP, Local, Transfer FedAVG [32], FedTL [34], FedKD [36], SADMC-MT-FF-FL	Macro F1, Micro F1
Ex3[4.6]	Dynamic Performance Analysis	The effect of different nodes on the performance of the model.	Macro F1
Ex4[4.7]	Visualized Explanation	Visualize the results after downscaling the anomaly scores.	
Ex5[4.8]	Ablation Experiments	Results of three ablation experiments and a comparison of weight aggregation methods.	Macro F1, Micro F1

Skoltech Anomaly Benchmark(SKAB).[5] It is a dataset for researching and testing cybersecurity, anomaly detection, and industrial control system (ICS) security. The goal of SKAB is to simulate and reproduce industrial control system Cyber attacks and anomalous events. SKAB is used to study and test the problems of cyber attack detection, anomaly detection and industrial process optimization. SKAB has only 8 feature dimensions, which is much more difficult for model feature extraction.

4.2. Evaluation metrics

Based on previous work [28], the anomaly detection experiments employ F1,Area Under Curve(AUC) and Matthews Correlation Coefficient(MCC) [48] as evaluation metrics, while the anomaly classification experiments utilize Macro F1 and Micro F1 [49] as evaluation metrics.

4.3. Experiments settings

The experiments were conducted across five distinct groups, namely Anomaly Detection, Anomaly Classification, Dynamic Performance Analysis, Visualized Explanation, and Ablation Experiments, and the detailed description is shown in Table 3. In our setup, anomaly classification tasks were operationalized on seven microservice and six Train-Ticket nodes. Conversely, the nodes assigned to SMD, SWaT, and SKAB were tasked with anomaly detection. The node dedicated to weight updates played a pivotal role in aggregating, processing, and disseminating weights across the network. Each dataset functioned as a distributed computing node within our framework, integrating a total of 20 nodes, inclusive of those designated for updates. These comprised seven microservice nodes, six Train-Ticket nodes, four SMD nodes, one SWaT node, one SKAB node, and a singular node for weight update execution. Under the Anomaly Detection and Anomaly Classification experiments we do two sets of control experiments, respectively from the Local Model and the Federated Learning-based method. The purpose of this is to evaluate the resilience and adaptability of the local model in the EA-MRS framework. At the same time, the adaptability enhancement of SADMC-MT-FF-FL over other basic methods for

[5] https://github.com/waico/SKAB

Table 4
Hyperparameters.

Neural network parameters	
Local model	Transfer
EA attenuation	64
Number of conv layer	3
Kernel filters in each conv layer	7, 5, and 3
Activation function	ReLU
Optimizer	AdamW
Learning rate	0.001
Batch size	128
Federated learning parameters	
Number of clients	20
Local training epoch	100
Communication round	4

microservice system deployment is verified by the method based on federated learning.

DatasetsSetup: Each dataset functioned as a distributed computing node within our framework, integrating a total of 20 nodes, inclusive of those designated for updates. These comprised seven microservice nodes, six Train-Ticket nodes, four SMD nodes, one SWaT node, one SKAB node, and a singular node for weight update execution. The anomaly classification experiments were conducted on Train-Ticket and Sock-Shop, using 13 microservice datasets, each with a total of 3 types of anomaly samples plus 1 type of normal samples. For the anomaly detection experiments, we selected 10,000 samples from SWaT, in SMD we chose the first 4 subsets for the experiments, and for SKAB we used all the data under the valve1 file. Because SMD, SWaT and SKAB cannot do anomaly classification, we only performed anomaly detection.

ParameterSettings: To better capture features and according to previous works [37], the convolution kernel sizes are 7, 5, and 3, respectively. We set the parameter settings for both the batch normalization layer and the attention mechanism to default values, and the value of the External Attention module (EA) attenuation was set to 64, denoting a scaling factor used to adjust the range or magnitude of the attention distribution when calculating the attention score, which can help the model better focus on processing the most relevant parts of the input sequence, thereby improving its performance. The detailed Settings are shown in Table 4.

DataPreprocessing: We perform MinMax normalization to the five datasets.

ExperimentalEnvironments: For multi-task simulations are based on virtual nodes implemented in Pytorch 1.13 and Python 3.9, and all experiments were performed on a PC with an NVIDIA 3070 (8G) graphics card, an Intel(R) Core(TM) i5-12600KF CPU @ 3.69 GHz and 16 GRAM.

4.4. Anomaly detection

Anomaly detection experiments We conducted on three public datasets, and the experiments were divided into two groups. The first group of experiments was compared with the traditional centralized anomaly detection method, and the second group of experiments was compared with the anomaly detection method based on the federated learning framework.

4.4.1. Local model anomaly detection

Table 5 presents the results of centralized anomaly detection model experiments conducted on three public datasets. Experimental results show that TimesNet performs at the level of SOTA on SWaT and SKAB, but its performance on SMD is not as good as other methods. Although the performance of Transfer model is not optimal on SWaT and SKAB, its overall performance is still the best. Fig. 5 graphically illustrates the overall effectiveness of *Transfer* across the three datasets for anomaly detection, clearly demonstrating that *Transfer* outperforms all baseline methods in this domain.

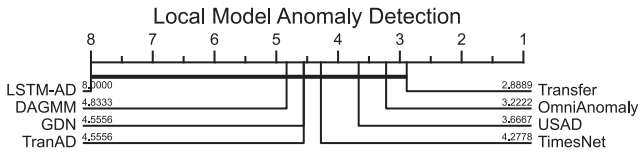


Fig. 5. Anomaly detection performance critical difference diagram. The overall performance of Transfer is more stable.

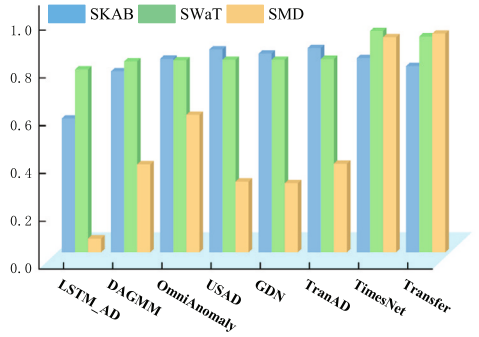


Fig. 6. Comparison results of *Transfer* model with EA-MRS structure with other anomaly detection models MCC.

Furthermore, as depicted in Fig. 6, *Transfer* also exhibits optimal performance in anomaly detection experiments. Specifically, *Transfer* displays robust discriminative ability in the SMD dataset. This is indicative of its competence in accurately classifying both positive and negative instances, which includes efficiently identifying true positive and true negative examples while minimizing the incidence of false positives and false negatives.

4.4.2. Federated learning based anomaly detection

Table 6 details the results of the distributed anomaly detection experiments conducted in our federated learning framework. The experimental results show that our method has more advantages than the other six federated learning frameworks. In SMD, F1 is 20.5% higher than the best comparison method, 6% higher than SWaT, and 0.8% higher than SKAB. Meanwhile, FedKD and Scaffold also perform well in distributed anomaly detection. The data from three publicly available datasets collectively demonstrate the higher robustness and generalization ability of SADMC-MT-FF-FL. Fig. 7 presents the MCC values for seven federated learning methods across these three datasets. Compared with the other six benchmark methods, our model achieves superior performance, further affirming the effectiveness of our proposed approach. The MCC is particularly valuable in anomaly detection and classification, offering robust evaluation in scenarios with an imbalance between positive and negative case categories. This is due to its comprehensive consideration of the four fundamental categorization outcomes. *Elastic* proposes a new elastic aggregation method to help the model converge faster. However, in the complex environment of anomaly detection, this aggregation method is not very suitable. *MOON* is a contrastive learning based federated learning framework, which aims to reduce the representation distance learned by the local model and the representation distance learned by the global model. In this way, the global features are better learned, but the local features are ignored. *Scaffold* introduces server-controlled variables and client-controlled variables to improve the case of user drift, but the performance is significantly degraded in the dynamic anomaly detection environment.

Table 5
The local model anomaly detection SWaT, SMD and SKAB results.

Dataset	*SWaT*				*SMD*				*SKAB*			
Metric	*Pre*	*Rec*	*F1*	*AUC*	*Pre*	*Rec*	*F1*	*AUC*	*Pre*	*Rec*	*F1*	*AUC*
LSTM-AD	0.777	0.010	0.021	0.505	0.501	0.457	0.564	0.501	0.561	0.625	0.587	0.616
DAGMM	0.959	0.695	0.806	0.844	0.999	0.844	0.846	0.911	0.705	**0.999**	0.827	0.819
OmniAnomaly	0.978	0.695	0.813	0.846	0.987	**0.998**	0.993	0.937	0.799	0.875	0.830	0.787
USAD	0.991	0.687	0.812	0.843	0.981	0.997	0.989	0.938	0.842	0.875	0.857	0.844
GDN	0.991	0.687	0.812	0.843	0.910	0.997	0.952	0.941	0.775	0.875	0.820	0.822
TranAD	**0.997**	0.687	0.814	0.844	0.907	0.997	0.950	0.930	0.748	0.875	0.804	0.810
TimesNet	0.921	**0.931**	**0.926**	**0.960**	0.879	0.815	0.846	0.900	**0.983**	0.777	**0.868**	**0.880**
Transfer	0.967	0.864	0.913	0.929	**0.999**	0.997	**0.998**	0.980	0.902	0.807	0.852	0.874

Table 6
Experimental results of the anomaly detection method based on federated learning on SWaT, SMD and SKAB.

Dataset	*SWaT*				*SMD*				*SKAB*			
Metric	*Pre*	*Rec*	*F1*	*AUC*	*Pre*	*Rec*	*F1*	*AUC*	*Pre*	*Rec*	*F1*	*AUC*
FedAVG	**0.994**	0.554	0.711	0.633	0.491	0.191	0.238	0.500	0.238	0.39	0.454	0.511
FedTL	0.985	0.619	0.761	0.781	0.706	0.230	0.324	0.666	0.324	0.254	0.345	0.503
FedKD	0.976	0.694	0.811	0.817	0.953	0.708	0.793	0.817	0.793	0.739	0.816	0.750
MOON	0.334	0.783	0.469	0.610	0.998	0.470	0.639	0.540	0.945	0.750	0.836	0.863
Scaffold	0.988	0.671	0.799	0.806	0.999	0.469	0.639	0.538	**0.933**	0.750	0.832	0.860
Elastic	0.189	**0.872**	0.311	0.571	0.955	0.342	0.503	0.516	0.963	0.356	0.519	0.536
SADMC-MT-FF-FL	0.902	0.848	**0.871**	**0.883**	**0.999**	**0.997**	**0.998**	**0.980**	0.911	**0.786**	**0.844**	**0.867**

Table 7
The local model anomaly classification Macro F1 and Micro F1 results.

	Dataset	*TT*						*SS*						
		assurance	food	food_map	inside	order_other	train	carts	catalogue	frontend	orders	payment	shipping	user
MacroF1	*SGD*	0.928	0.927	0.957	0.852	0.944	0.889	0.782	0.865	0.885	0.847	0.880	0.844	0.747
	KNN	0.727	0.700	0.833	0.660	0.707	0.715	0.776	0.878	0.884	**0.855**	**0.895**	0.839	0.802
	CNN	0.908	0.899	0.933	0.844	0.903	0.871	0.704	0.850	0.803	0.806	0.824	0.832	0.704
	MLP	0.952	0.910	0.956	0.820	0.931	0.874	0.777	0.861	**0.890**	0.847	0.871	0.780	0.664
	Local	**0.953**	0.930	0.960	0.886	0.940	0.916	0.830	0.860	0.885	0.854	0.870	0.854	0.815
	Transfer	0.951	**0.935**	**0.966**	**0.887**	**0.952**	**0.919**	**0.833**	**0.879**	0.879	0.852	0.868	**0.873**	**0.841**
MicroF1	*SGD*	0.923	0.916	0.953	0.833	0.946	0.883	0.782	0.857	0.885	0.854	0.881	0.840	0.742
	KNN	0.738	0.716	0.827	0.657	0.721	0.729	0.773	0.872	0.884	**0.864**	**0.896**	0.839	0.798
	CNN	0.910	0.900	0.934	0.845	0.903	0.876	0.701	0.845	0.805	0.819	0.827	0.827	0.692
	MLP	0.951	0.911	0.954	0.815	0.929	0.880	0.775	0.851	**0.889**	0.852	0.873	0.786	0.670
	Local	**0.953**	0.931	0.961	0.888	0.939	0.919	0.829	0.860	0.887	0.856	0.872	0.855	0.812
	Transfer	0.951	**0.936**	**0.967**	**0.889**	**0.951**	**0.922**	**0.835**	**0.879**	0.881	0.856	0.867	**0.871**	**0.842**

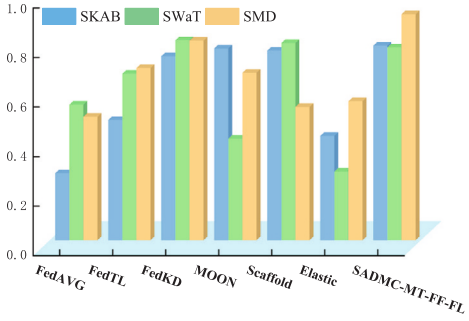


Fig. 7. Results of SADMC-MT-FF-FL compared to other federated learning methods MCC.

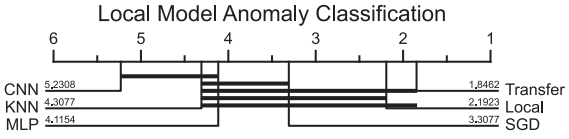


Fig. 8. Anomaly classification performance critical difference diagram.

4.5. Anomaly classification

Anomaly classification experiments were carried out on two microservice datasets, and the experiments were divided into two groups. The first set of experiments is compared with the traditional centralized method, and the second set of experiments is compared with the anomaly classification method based on the federated learning framework. The anomaly classification experiments aimed at identifying three distinct anomalies within the SS and TT datasets.

4.5.1. Local model anomaly classification

The structure of the *Transfer* model is EA-MRS. In the realm of machine learning, the KNN algorithm is widely utilized for both classification and regression tasks. Specifically, in classification tasks, KNN assigns a label to a new sample by analyzing the labels of its closest neighbors. Another notable algorithm is the SGDClassifier, which is underpinned by the stochastic gradient descent (SGD) optimizer. Primarily, it functions as a linear model for classification tasks. The results of the anomaly classification are systematically presented in Table 7. When compared with baseline methodologies, the *Transfer* model exhibits a marginal superiority in both macro F1 and micro F1 scores. However, the performance of the one-dimensional CNN model lags slightly behind other models. Fig. 8 provides further insights, where the Critical Difference(CD) diagram reveals that *Transfer* model with EA-MRS outperforms others, demonstrating exceptional efficacy in the local model anomaly classification experiment, alongside commendable performance from all six methods involved.

Table 8
Federated learning based anomaly classification Macro F1 and Micro F1 results.

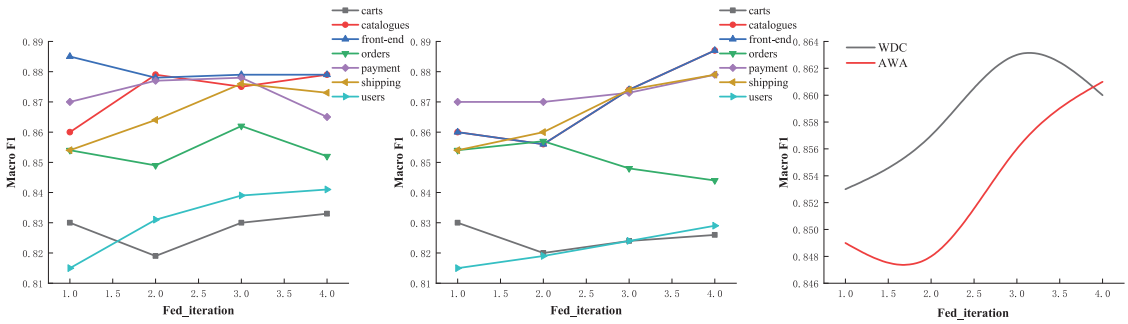
	Dataset	*TT*						*SS*						
		assurance	food	food_map	inside	order_other	train	carts	catalogue	frontend	orders	payment	shipping	user
MacroF1	*FedAVG*	0.674	0.678	0.634	0.594	0.660	0.658	0.700	0.828	0.831	0.757	0.808	0.779	0.686
	FedTL	0.685	0.010	0.392	0.001	0.352	0.535	0.428	0.302	0.563	0.191	0.154	0.297	0.287
	FedKD	0.606	0.632	0.628	0.459	0.525	0.526	0.671	0.799	0.812	0.737	0.776	0.747	0.641
	SADMC-MT-FF-FL	**0.951**	**0.931**	**0.963**	**0.888**	**0.945**	**0.918**	**0.828**	**0.873**	**0.880**	**0.854**	**0.873**	**0.867**	**0.832**
MicroF1	*FedAVG*	0.687	0.679	0.638	0.589	0.664	0.674	0.699	0.823	0.833	0.773	0.810	0.782	0.690
	FedTL	0.69	0.010	0.411	0.001	0.373	0.526	0.431	0.364	0.566	0.267	0.568	0.366	0.338
	FedKD	0.620	0.636	0.635	0.454	0.537	0.541	0.669	0.794	0.815	0.754	0.778	0.751	0.645
	SADMC-MT-FF-FL	**0.951**	**0.932**	**0.963**	**0.890**	**0.944**	**0.921**	**0.829**	**0.873**	**0.882**	**0.857**	**0.874**	**0.866**	**0.831**

Table 9
Macro F1 results of ablation experiments on seven microservices datasets.

	carts	catalogue	frontend	orders	payments	shipping	users	Avg
SADMC-MT-FF-FL	**0.828**	**0.873**	**0.880**	**0.854**	0.873	0.867	**0.832**	**0.858**
w/o LGF-PKT	0.825(−0.3%)	0.869(−0.4%)	0.869(−1.1%)	0.851(−0.3%)	0.873	0.867	0.822(−1.0%)	0.854(−0.4%)
w/o EA-MRS	0.646(−18.2%)	0.632(−24.1%)	0.692(−18.8%)	0.689(−16.5%)	0.755(−11.8%)	0.752(−11.5%)	0.661(−17.1%)	0.687(−17.1%)
w/o LGF-PKT,EA-MRS	0.594(−23.4%)	0.565(−30.8%)	0.683(−19.7%)	0.674(-18%)	0.719(−15.4%)	0.718(−14.9%)	0.606(−22.6%)	0.615(−24.3%)

Table 10
Micro F1 results of ablation experiments on seven microservices datasets.

	carts	catalogue	frontend	orders	payments	shipping	users	Avg
SADMC-MT-FF-FL	**0.829**	**0.873**	**0.882**	**0.857**	0.874	0.866	**0.831**	**0.859**
w/o LGF-PKT	0.825(−0.4%)	0.869(−0.4%)	0.869(−1.3%)	0.853(−0.4%)	0.875(+0.1%)	0.867(+0.1%)	0.820(−1.1%)	0.854(−0.5%)
w/o EA-MRS	0.632(−19.7%)	0.689(−18.4%)	0.752(-13%)	0.671(−18.6%)	0.693(−18.1%)	0.683(−18.3%)	0.671(-16%)	0.684(−17.5%)
w/o LGF-PKT,EA-MRS	0.565(−26.4%)	0.683(-19%)	0.674(−20.8%)	0.719(−13.8%)	0.718(−15.6%)	0.606(-26%)	0.604(−22.7%)	0.625(−23.4%)



(a) Weight Distance Calculation(WDC) (b) Average Weight Aggregation(AWA) (c) Comparison of overall optimization effect

Fig. 9. Comparison of the optimization effect of two weights scheme on individual microservices.

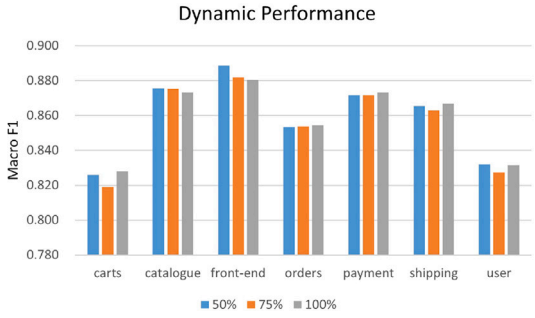


Fig. 10. Impact of changes in computing nodes on performance.

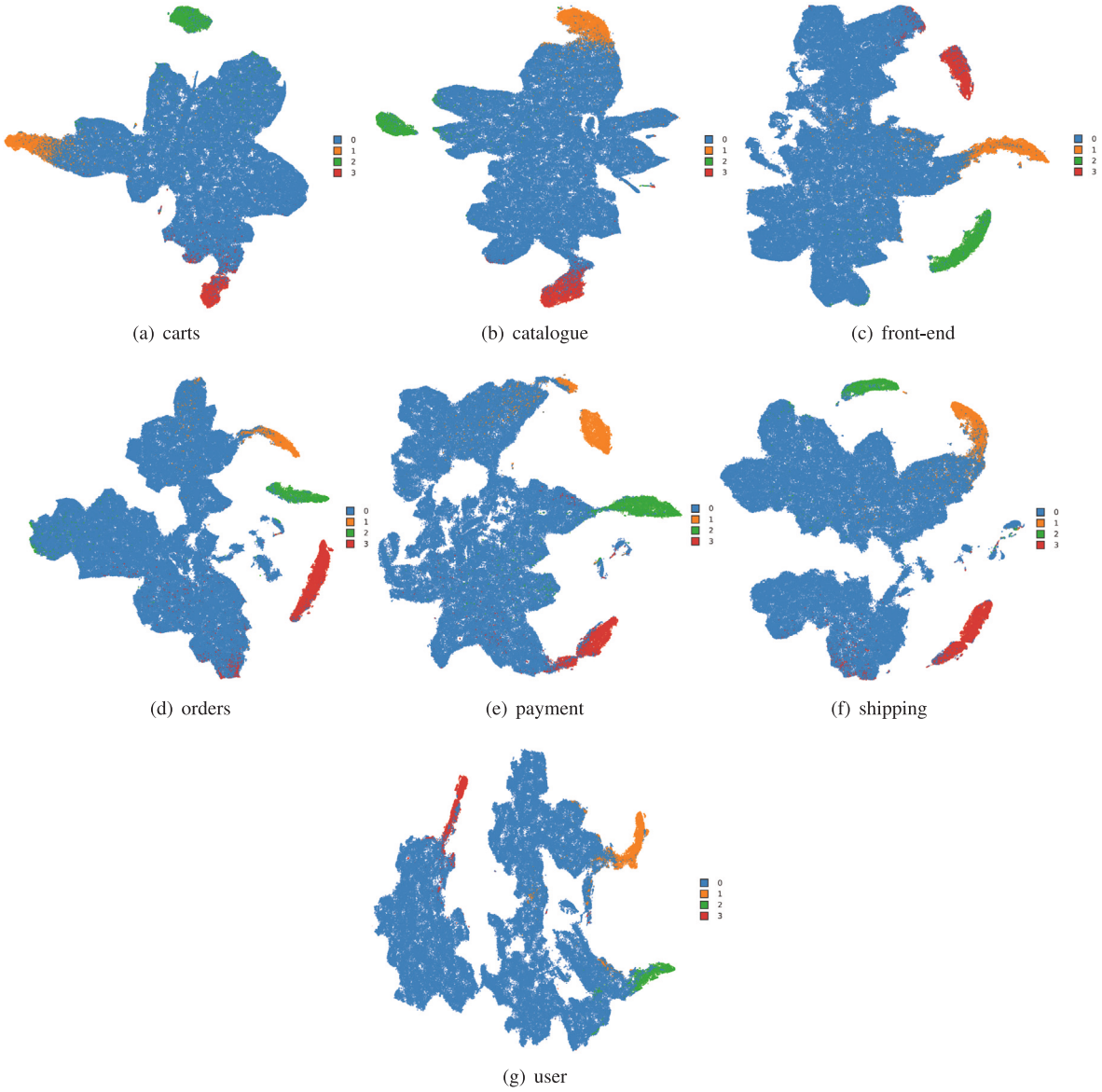
4.5.2. Federated learning based anomaly classification

In the distributed scenario, our research involved conducting experiments across four training cycles within the federated learning framework. Each cycle's results were averaged for precision, and the comprehensive outcomes for the 13 microservices are depicted in Table 8. An analysis of the three foundational methods reveals that our approach demonstrates superior performance across all 13 microservices.

The FedAVG employs a global machine learning framework, where distributed nodes collaborate and communicate. Here, mobile devices or nodes train on local data using their models and forward the *Local* model weights to a central weight aggregation node. This node averages the incoming model weights and redistributes the averaged parameters back to all nodes for anomaly classification. A caveat of this approach is that if weights are uniformly averaged, the resultant features might not represent local specifics adequately, potentially hindering rather than aiding feature extraction.

FedKD aims to mitigate model size and computational demands by condensing larger neural networks into more compact forms through distributed node collaboration. This method involves averaging received intermediate layer feature vectors at the weight aggregation nodes, which serve as features for the smaller neural network. Similar to FedAVG, FedKD also employs weighted averaging for feature weights, which may overlook the unique characteristics of local models.

Finally, FedTL utilizes a pre-trained model within the collaborative framework of distributed nodes. This method focuses on rapid iterative training on local devices, enhancing model performance and



(a) carts (b) catalogue (c) front-end

(d) orders (e) payment (f) shipping

(g) user

Fig. 11. Seven microservices visualizations, 0 for normal data, 1 for CPU Hogs, 2 for Mem Leak, and 3 for Net Latency.

generalization. However, FedTL's efficacy is largely contingent on the interplay between the pre-trained models and the nodes responsible for weight alignment, resulting in variable performance across different microservices.

4.6. Dynamic performance analysis

Our dynamic performance experiments on the Sock-Shop (SS) dataset are shown in Fig. 10, with a total of 20 nodes retaining 50%, 75% and 100% of the compute nodes, respectively. When training only 50% of the nodes, the average macro F1 of SADMC-MT-FF-FL is 0.859, 0.856 for 75% of the nodes, and 0.858 for all the nodes. Therefore, the experimental results demonstrate the effectiveness and scalability of SADMC-MT-FF-FL in dynamic microservice environments.

4.7. Visualized explanation

In our study, we employed t-SNE[6] downscaling analysis to scrutinize the final outcomes. Fig. 11 illustrates the result visualizations for two microservices, where '0' signifies normal data, '1' indicates CPU Hogs, '2' represents Mem Leak, and '3' denotes Net Latency. The visualization clearly demonstrates the efficacy of our method in distinguishing between these anomalies. Anomalies can be effectively detected and identified across seven microservices.

A notable observation from Fig. 11(g) is the clustering of all three types of anomalies (CPU Hogs, Mem Leak, and Net Latency). This clustering indicates a challenge in accurately characterizing the anomaly data specific to the User microservice. The confluence of these anomalies suggests a higher complexity or similarity in their patterns within this particular microservice, rendering it more challenging to distinguish them individually.

4.8. Ablation experiments

We validate the effectiveness of LGF-PKT framework and EA-MRS through three ablation experiments. The first set of experiments will remove LGF-PKT in favor of most of the existing methods [32–38] using the Weight Averaging Aggregation(AWA) Scheme to aggregate the weights and thus update the model; the second set of experiments removes the EA-MRS retaining the LGF-PKT and the *Local* model is used

[6] https://github.com/pavlin-policar/openTSNE

for the local model; and the third set of experiments retains only the Multi-task Federated Learning framework using the Weight Averaging Aggregation(AWA) Scheme.

From Tables 9 and 10, we can see that when we remove LGF-PKT, the performance has a more significant drop on the frontend, orders, payment and shipping datasets, with an average Macro F1 and Micro F1 drop of 0.4% and 0.5%. From the experimental results, LGF-PKT still has a significant performance improvement over most model updating methods, and local-to-global weight updating in a multi-task federated learning framework has better improvement for specific local models. In order to prove the effectiveness of the Weight Distance Calculation scheme, we analyze the stability of seven microservices Macro F1 over four federated learning cycles, and we also compare the average weight scheme in a comparative experiment, as shown in Fig. 9. In Fig. 9(c), it can be clearly seen that the overall optimality of the first three rounds of federated learning of WDC shows an upward trend and rises faster; while AWA as a whole, although also in an upward trend, is more appropriate in a dynamic and real-time distributed environment.

When we remove the EA-MRS, there is a significant decrease in the overall performance Macro F1 and Micro F1 by 17.1% and 17.5% on average. From the experiments, we can see that there is a significant decrease in the feature extraction ability of the system model and EA-MRS plays a significant role in anomaly feature extraction in the microservices architecture.

When we remove LGF-PKT and EA-MRS and keep only the multi-task federated learning wide leave, Macro F1 and Micro F1 decreased by 24.3% and 23.4% on average on the seven datasets, which effectively verifies the effectiveness of LGF-PKT and EA-MRS under the architecture.

5. Conclusion and future work

In this paper we propose the SADMC-MT-FF-FL(System Anomaly Detection and Multi-Classification based on Multi-Task Feature Fusion Federated Learning). The experimental results show that SADMC-MT-FF-FL outperforms all the Federated Learning methods in microservices system anomaly detection and classification; it also performs optimally in multi-task system anomaly detection experiments.

In addition, there are also some limitations in our method. (1) Challenges related to communication overhead, synchronization, and computational complexity as the deployment scale up. Data privacy protection is particularly important in sensitive tasks such as anomaly detection, and further research is needed to provide stronger privacy protection in federated learning, such as differential privacy technology and cryptographic methods, to protect the data privacy of participants. (2) In a distributed environment, non-IID data distribution may introduce bias and inconsistency, which affects the training process and leads to client drift, resulting in model performance degradation.

Our future work will be carried out from the following aspects. Firstly, the Local–Global Feature-based Parallel Knowledge Transfer Framework (LGF-PKT) will be optimized to discover the commonality between independent microservices. Other unimportant features will be filtered out to further improve the generalization ability of the model. We will also enhance the feature extraction of contextual information that can enrich the ability of anomaly detection and classification algorithms to correctly identify observations or sequences that do not match the expected behavior. Secondly, data resampling and sampling distributed nodes will be performed during distributed training, which can effectively reduce the bias and inconsistency caused by non-IID data distribution to improve the applicability and stability of SADMC-MT-FF-FL in different distributed environments, such as more complex IoT monitoring and natural disaster risk recognition. At the same time, more real cases or datasets will be collected for verification. On this basis, an anomaly root cause localization technology based on causal reasoning will be introduced, which can quickly locate the anomaly and generate an immediate response in the distributed environment. Through the anomaly root cause localization technology, it will be possible to more accurately locate the distributed node that caused the anomaly.

CRediT authorship contribution statement

Junfeng Hao: Writing – original draft, Software, Methodology, Conceptualization. **Peng Chen:** Writing – review & editing, Supervision, Methodology. **Juan Chen:** Validation, Software, Methodology. **Xi Li:** Software, Methodology, Data curation.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

This work has been partially supported by National Natural Science Foundation of China under Grant No. 62376043 and Science and Technology Program of Sichuan Province under Grant No. 2020JDRC0067, No. 2023JDRC0087, and No. 24NSFTD0025.

References

[1] P. Jamshidi, C. Pahl, N.C. Mendonça, J. Lewis, S. Tilkov, Microservices: The journey so far and challenges ahead, IEEE Softw. 35 (3) (2018) 24–35.
[2] A. Ikram, S. Chakraborty, S. Mitra, S. Saini, S. Bagchi, M. Kocaoglu, Root cause analysis of failures in microservices through causal discovery, Adv. Neural Inf. Process. Syst. 35 (2022) 31158–31170.
[3] R. Xin, P. Chen, Z. Zhao, Causalrca: Causal inference based precise fine-grained root cause localization for microservice applications, J. Syst. Softw. 203 (2023) 111724.
[4] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, Exploring microservices for enhancing internet QoS, Trans. Emerg. Telecommun. Technol. 29 (11) (2018) e3445.
[5] P. Chen, H. Liu, R. Xin, T. Carval, J. Zhao, Y. Xia, Z. Zhao, Effectively detecting operational anomalies in large-scale iot data infrastructures by using a gan-based predictive model, Comput. J. 65 (11) (2022) 2909–2925.
[6] R. Zhang, J. Chen, Y. Song, W. Shan, P. Chen, Y. Xia, An effective transformation-encoding-attention framework for multivariate time series anomaly detection in IoT environment, Mob. Netw. Appl. (2023) 1–13.
[7] Y. Song, R. Xin, P. Chen, R. Zhang, J. Chen, Z. Zhao, Autonomous selection of the fault classification models for diagnosing microservice applications, Future Gener. Comput. Syst. (2023).
[8] A.A. Cook, G. Mısırlı, Z. Fan, Anomaly detection for IoT time-series data: A survey, IEEE Internet Things J. 7 (7) (2019) 6481–6494.
[9] L. Mariani, C. Monni, M. Pezzé, O. Riganelli, R. Xin, Localizing faults in cloud systems, in: 2018 IEEE 11th International Conference on Software Testing, Verification and Validation, ICST, IEEE, 2018, pp. 262–273.
[10] R. Xin, H. Liu, P. Chen, Z. Zhao, Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework, J. Cloud Comput. 12 (1) (2023) 1–16.
[11] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, D. Ding, Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study, IEEE Trans. Softw. Eng. 47 (2) (2018) 243–260.
[12] R. Wang, W. Huang, X. Zhang, J. Wang, C. Ding, C. Shen, Federated contrastive prototype learning: An efficient collaborative fault diagnosis method with data privacy, Knowl.-Based Syst. 281 (2023) 111093.
[13] P. Chen, Y. Xia, S. Pang, J. Li, A probabilistic model for performance analysis of cloud infrastructures, Concurr. Comput.: Pract. Exper. 27 (17) (2015) 4784–4796.
[14] J. Chen, P. Chen, X. Niu, Z. Wu, L. Xiong, C. Shi, Task offloading in hybrid-decision-based multi-cloud computing network: a cooperative multi-agent deep reinforcement learning, J. Cloud Comput. 11 (1) (2022) 1–17.
[15] T. Long, P. Chen, Y. Xia, Y. Ma, X. Sun, J. Zhao, Y. Lyu, A deep deterministic policy gradient-based method for enforcing service fault-tolerance in MEC, Chin. J. Electron. 34 (2024) 1–11.
[16] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl.-Based Syst. 216 (2021) 106775.
[17] V. Mothukuri, R.M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Gener. Comput. Syst. 115 (2021) 619–640.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[19] R. Du, S. Xu, R. Zhang, L. Xu, H. Xia, A dynamic adaptive iterative clustered federated learning scheme, Knowl.-Based Syst. 276 (2023) 110741.
[20] V. Smith, C.-K. Chiang, M. Sanjabi, A.S. Talwalkar, Federated multi-task learning, Adv. Neural Inf. Process. Syst. 30 (2017).
[21] M. Teng, Anomaly detection on time series, in: 2010 IEEE International Conference on Progress in Informatics and Computing, vol. 1, IEEE, 2010, pp. 603–608.
[22] F. Kabir, S. Siddique, M.R.A. Kotwal, M.N. Huda, Bangla text document categorization using stochastic gradient descent (SGD) classifier, in: 2015 International Conference on Cognitive Computing and Information Processing, CCIP, IEEE, 2015, pp. 1–4.
[23] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff, LSTM-based encoder-decoder for multi-sensor anomaly detection, 2016, arXiv preprint arXiv:1607.00148.
[24] B. Zong, Q. Song, M.R. Min, W. Cheng, C. Lumezanu, D. Cho, H. Chen, Deep autoencoding gaussian mixture model for unsupervised anomaly detection, in: International Conference on Learning Representations, 2018.
[25] Y. Song, R. Xin, P. Chen, R. Zhang, J. Chen, Z. Zhao, Identifying performance anomalies in fluctuating cloud environments: A robust correlative-GNN-based explainable approach, Future Gener. Comput. Syst. 145 (2023) 77–86.
[26] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2828–2837.
[27] J. Audibert, P. Michiardi, F. Guyard, S. Marti, M.A. Zuluaga, Usad: Unsupervised anomaly detection on multivariate time series, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3395–3404.
[28] S. Tuli, G. Casale, N.R. Jennings, TranAD: Deep transformer networks for anomaly detection in multivariate time series data, Proc. VLDB 15 (6) (2022) 1201–1214.
[29] A. Deng, B. Hooi, Graph neural network-based anomaly detection in multivariate time series, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 5, 2021, pp. 4027–4035.
[30] S. Qi, J. Chen, P. Chen, P. Wen, X. Niu, L. Xu, An efficient GAN-based predictive framework for multivariate time series anomaly prediction in cloud data centers, J. Supercomput. (2023) 1–26.
[31] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, TimesNet: Temporal 2D-variation modeling for general time series analysis, in: The Eleventh International Conference on Learning Representations, 2023.
[32] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of FedAvg on non-IID data, in: International Conference on Learning Representations, 2020.
[33] J. Xu, J. Lin, Y. Li, Z. Xu, MultiFed: A fast converging federated learning framework for services QoS prediction via cloud–edge collaboration mechanism, Knowl.-Based Syst. 268 (2023) 110463.
[34] H. Yang, H. He, W. Zhang, X. Cao, FedSteg: A federated transfer learning framework for secure image steganalysis, IEEE Trans. Netw. Sci. Eng. 8 (2) (2020) 1084–1094.
[35] L. Wan, J. Ning, Y. Li, C. Li, K. Li, Intelligent fault diagnosis via ring-based decentralized federated transfer learning, Knowl.-Based Syst. (2023) 111288.
[36] Z. Zhu, J. Hong, J. Zhou, Data-free knowledge distillation for heterogeneous federated learning, in: International Conference on Machine Learning, PMLR, 2021, pp. 12878–12889.
[37] C. He, M. Annavaram, S. Avestimehr, Group knowledge transfer: Federated learning of large cnns at the edge, Adv. Neural Inf. Process. Syst. 33 (2020) 14068–14080.
[38] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, J. Jiang, Multi-center federated learning: clients clustering for better personalization, World Wide Web 26 (1) (2023) 481–500.
[39] Q. Li, B. He, D. Song, Model-contrastive federated learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 10713–10722.
[40] S.P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, A.T. Suresh, Scaffold: Stochastic controlled averaging for federated learning, in: International Conference on Machine Learning, PMLR, 2020, pp. 5132–5143.
[41] D. Chen, J. Hu, V.J. Tan, X. Wei, E. Wu, Elastic aggregation for federated optimization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 12187–12197.
[42] R.A. Sater, A.B. Hamza, A federated learning approach to anomaly detection in smart buildings, ACM Trans. Internet Things 2 (4) (2021) 1–23.
[43] P. Qi, D. Chiaro, A. Guzzo, M. Ianni, G. Fortino, F. Piccialli, Model aggregation techniques in federated learning: A comprehensive survey, Future Gener. Comput. Syst. (2023).
[44] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H.B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1175–1191.
[45] M.-H. Guo, Z.-N. Liu, T.-J. Mu, S.-M. Hu, Beyond self-attention: External attention using two linear layers for visual tasks, IEEE Trans. Pattern Anal. Mach. Intell. 45 (5) (2022) 5436–5447.
[46] H. Zhu, J. Zhang, H. Cui, K. Wang, Q. Tang, TCRAN: Multivariate time series classification using residual channel attention networks with time correction, Appl. Soft Comput. 114 (2022) 108117.
[47] C. Lee, T. Yang, Z. Chen, Y. Su, M.R. Lyu, Eadro: An end-to-end troubleshooting framework for microservices on multi-source data, 2023, arXiv preprint arXiv:2302.05092.
[48] D. Chicco, G. Jurman, The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, BMC Genomics 21 (1) (2020) 1–13.
[49] M. Grandini, C. SpA, E. Bagli, G. Visani, Metrics for multi-class classification: An overview, stat 1050 (2020) 13.

Junfeng Hao is currently working towards a master degree at School of Computer and Software Engineering, Xihua University, Chengdu, China. His research areas include cloud computing, anomaly detection and time series analysis.

Peng Chen is currently a full professor of School of Computer and Software Engineering, Xihua University, Chengdu, China. His research interests include edge computing, cloud computing, anomaly detection and time series analysis.

Juan Chen is currently a lecturer of School of Computer and Software Engineering, Xihua University, Chengdu, China. Her research interests include edge computing, cloud computing and deep learning.

Xi Li is currently an associate professor of School of Computer and Software Engineering, Xihua University, Chengdu, China. Her research interests include network and information security, anomaly detection and time series analysis.