Bus Inf Syst Eng 66(1):19–42 (2024)
https://doi.org/10.1007/s12599-023-00825-8

RESEARCH PAPER

Dynamic Circular Network-Based Federated Dual-View Learning for Multivariate Time Series Anomaly Detection

Weishan Zhang · Yuqian Wang · Leiming Chen · Yong Yuan · Xingjie Zeng · Liang Xu · Hongwei Zhao

Received: 24 December 2022 / Accepted: 14 June 2023 / Published online: 31 July 2023

Abstract Multivariate time-series data exhibit intricate correlations in both temporal and spatial dimensions. However, existing network architectures often overlook dependencies in the spatial dimension and struggle to strike a balance between long-term and short-term patterns when extracting features from the data. Furthermore, industries within the business community are hesitant to share their raw data, which hinders anomaly prediction accuracy and detection performance. To address these challenges, the authors propose a dynamic circular network-based federated dual-view learning approach. Experimental results from four open-source datasets demonstrate that the method outperforms existing methods in terms of accuracy, recall, and F1_score for anomaly detection.

Keywords Multivariate time series · Federated learning · Graph neural network · Anomaly detection · Deep learning

Accepted after three revisions by the editors of the special issue.

W. Zhang (✉) · Y. Wang · L. Chen · X. Zeng · H. Zhao
College of Computer Science and Technology, China University of Petroleum (East China), No.66 West Changjiang Road, Qingdao 266580, China
e-mail: zhangws@upc.edu.cn

Y. Yuan
School of Mathematics, Renmin University of China, No.59 Zhongguancun Street, Beijing 100872, China

L. Xu
College of Computer Science and Communication Engineering, Beijing University of Science and Technology, No.30 College Road, Beijing 100083, China

1 Introduction

Smart city and smart manufacturing processes generate large amounts of multivariate time-series data that capture information about the states of corresponding equipment. There have been efforts to utilize this data and integrate intelligent mechanisms to predict anomalies for the purpose of preventive maintenance. Mehdiyev et al. (2020) suggested a multi-stage deep learning approach that utilizes a classification strategy to predict process events and address undesired deviations from the required workflow. Wu et al. (2020) developed an LSTM-Gauss-NBayes method for anomalous event detection, which extracts regular time series information using an LSTM block and detects outliers using the Gaussian Naive Bayes model. However, these approaches only consider temporal dimensional features of multivariate time-series data, while ignoring their spatial dependencies.

To address this limitation, Graph Neural Networks (GNNs) have been introduced. Jiang et al. (2019) designed a GCN (Kipf and Welling 2017)-based anomaly detection model that characterizes different entity attributes and the structure of associations between attributes as graphs, enabling the detection of abnormal behaviors of users and malicious threat groups. Deng and Hooi (2021) introduced GAT (Velickovic et al. 2018) to model the spatial structure of multivariate time-series data in their GDN approach. They also proposed a graph deviation scoring method for anomaly detection. However, the use of GNNs requires defining a reasonable edge structure for spatial information aggregation between different variables. It remains challenging to apply GNNs to model multivariate time-series data without an initial spatial structure.

To address this challenge, Deng and Hooi (2021) proposed a spatial structure modeling method based on

similarities of node embedding vectors. However, the embedding vectors in their method are randomly generated and do not take into account the characteristics of each sensor. Wu et al. (2020b) introduced a directed graph modeling method that only calculates dependencies between subsets of nodes to alleviate computational and memory bottleneck problems. However, this technique cannot generate different edge structures for different moments in time. For instance, in the production process of semiconductor materials, the relation between temperature and gas volume sensors varies with time, which requires different weights for the spatial structure corresponding to the data at different moments in the model training process. To construct a more robust edge structure, a technique with greater stability is required.

Additionally, some studies (Mehdiyev et al. 2020; Wu et al. 2020; Jiang et al. 2019; Deng and Hooi 2021) focus solely on learning either temporal or spatial features, while multivariate time-series data from sensors often exhibit complex dependencies in both dimensions. Therefore, it is necessary to consider dependencies in both dimensions to extract features effectively. Oberdorf et al. (2022) proposed a multi-headed deep neural network that combines CNN, GNN, LSTM, and MLP networks to predict end-to-end enterprise process network monitoring. Yu et al. (2018) introduced a "sandwich" structured spatio-temporal convolutional model with a spatial graph convolutional layer sandwiched between two temporally gated convolutional layers, which can model both temporal and spatial dimensions of multivariate time-series data. However, there is a significant difference in extracting temporal information with different lengths of sliding windows. For this reason, a new network architecture is required that can efficiently extract temporal features and flexibly aggregate spatial characteristics.

The increasing concerns about data privacy and security (Baumann et al. 2019) have led to businesses being unwilling to share their raw data directly, and isolated data islands (Liang et al. 2020) have become a common problem for anomaly detection. Traditional centralized learning is limited by these data silos, leading to poor model performance. To address this, federated learning (McMahan et al. 2017) has been proposed as a promising paradigm that enables collaborative modeling between different clients while protecting data privacy.

This paper proposes a novel approach based on Dynamic Circular Network and Federated Dual-View Learning (FDVL-DCN) for anomaly detection. To make the spatial structure learned from multivariate time-series data more consistent with realistic scenarios, we propose a novel edge structure modeling approach that integrates three aspect information, **C**hange of inter-sensor correlations, **D**ecline of the strength of time-series data influence, and each sensor's **F**eatures, to learn the relationships among sensors and encode them as edges in the graph. This approach is named as CDF-based edge structure modeling. To better predict future behavior using multivariate time-series data, we extract features from temporal and spatial dimensions. A circular network architecture can be used to address conflicts between long-term and short-term predictions by adjusting the neural network based on input data size. To protect data privacy while improving anomaly detection performance, we introduce the idea of federated dual-view learning, which integrates the LOSS intervals recorded during the training of positive and negative view models to find thresholds and perform anomaly detection to obtain a higher F1 value.

In summary, the main contributions of the paper are as follows:

- We propose a more general and interpretable method for modeling spatial structure, which enables the flexible application of GNN to multivariate time-series data without initialized spatial structure.
- We design a new anomaly detection method, which can protect data privacy. It extracts feature information from multivariate time series data by integrating features of both temporal and spatial dimensions and considers the effect of sliding window length.
- The proposed approach is evaluated using four open-source datasets, and the experimental results show that FDVL-DCN outperforms baseline anomaly detection methods. Further investigations and analysis demonstrate the generality and effectiveness of the proposed method as a big data service solution for anomaly detection.

The structure of this article is as follows: Sect. 2 provides a literature review on technologies related to graph neural networks, spatio-temporal feature extraction, anomaly detection, and federated learning. In Sect. 3, we introduce our proposed method in detail, which is composed of five distinct parts: CDF-based Edge Structure Modeling Approach, Dynamic Circular Network Architecture, Architecture Design of FDVL-DCN, Anomaly Evaluation Mechanism, and FDVL-DCN Algorithm as a Software Service. In Sect. 4, experiments are conducted to verify the effectiveness of the CDF-based Edge Structure Modeling Approach, the effectiveness and generality of the Dynamic Circular Network Architecture, the prediction performance of DCN, the comparison of anomaly detection performance between DCN and FDVL-DCN with baseline methods, the evaluation of model performance in federated and non-federated scenarios, and the quality measurement of FDVL-DCN big data as service. Finally, Sect. 5 presents the conclusion of this article.

2 Related Work

2.1 Graph Neural Network

In recent years, Graph Neural Networks (GNNs) have become a successful approach for modeling patterns in graph-structured data. GNNs assume that a node's state depends on its neighbors' conditions and capture high-level representations of nodes by passing information from their neighbors to the node itself. Various kinds of GNNs have been developed through graph convolution (Kipf and Welling 2017), graph attention (Velickovic et al. 2018), message passing (Gilmer et al. 2017), and information propagation (Klicpera et al. 2019). GNNs and related variants have shown success in time-dependent problems, such as video analysis Zhong et al. (2019) and recommendation systems (Schlichtkrull et al. 2018; Lim et al. 2020).

However, these methods require initialized edge structures, making it challenging to apply GNN to multivariate time series data without an initial spatial structure.

2.2 Spatio-Temporal Feature Extraction

Spatio-temporal neural networks have shown success in various fields, and for extracting spatio-temporal features of multivariate time series data, GNN can capture spatial dependencies between nodes, while deep learning approaches such as convolutional neural networks (Wang et al. 2019) can capture temporal dependencies. This type of neural network was initially proposed to solve the problem of traffic prediction (Zhao et al. 2020; Chen et al. 2020) and skeleton-based action recognition (Shi et al. 2019; Yan et al. 2018), and its effectiveness has been proved for multivariate time series forecasting problems (Cao et al. 2020).

Spatio-temporal neural networks improve the extraction capability of multivariate time series data by considering both temporal and spatial dimensions. However, the existing architecture needs to balance the effectiveness of both long-term and short-term forecasts.

2.3 Anomaly Detection

Anomaly detection aims to detect unusual samples which deviate from the majority of the data, traditional methods for anomaly detection include linear-model-based approaches (Shyu et al. 2003), distance-based strategies (Hautamäki et al. 2004; Sugiyama and Borgwardt 2013), Proximity-based approaches (Goldstein and Dengel 2012; Liu et al. 2008), probabilistic based approaches (Li et al. 2022), but they may not work well on multivariate time series due to their inability to capture spatial-temporal dependencies. Deep learning methods have improved high-dimensional datasets anomaly detection, including generative adversarial networks used in Liu et al. (2020), disentangled representations in variational autoencoders utilized in Burgess et al. (2018), hypersphere mapping by a neural network in Ruff et al. (2018), an enhanced algorithm for intelligent anomaly detection in imbalance learning proposed in Zhou et al. (2022), an imbalanced detection method based on DTW (Dynamic Time Warping) oversampling and improved KNN in Langfu et al. (2023), a group anomaly detection method based on clustering algorithms, KNN, and pattern mining frameworks in Belhadi et al. (2021), and a comprehensive anomaly detection algorithm that utilizes decomposition methods, deep neural networks, and evolutionary computation to address anomalies in the Internet of Everything in Djenouri et al. (2021).

However, these methods only capture dependencies on a single dimension.

2.4 Federated Learning

Federated learning (McMahan et al. 2017) is a decentralized architecture that uses a central server to train a shared global model from decentralized data spread across many different clients. It has been used in various domains due to increasing data privacy concerns. Zhang et al. (2022) proposed an efficient semi-asynchronous federated learning framework for predicting the power generation of photovoltaic power plants, while Zhang et al. (2021) designed a dynamic fusion method to improve the communication efficiency and model performance of federated learning. Zhou et al. (2022b) developed a 2-dimensional federated learning framework to address insufficient training data and insecure data sharing in Cyber-Physical-Social Systems, while Zhou et al. (2021) proposed a multi-layer heterogeneous model selection and aggregation scheme for 6 G supported vehicular networks.

To improve federated learning for anomaly detection, one approach is to train positive view and negative view models separately.

3 Conceptual Framework

The FDVL-DCN framework focuses on multivariate time series anomaly detection and multi-client collaborative modeling while protecting data privacy. Figure 1 shows each client's general architecture of the DCN. We use $X_{train} = \{x_{t1}, \quad x_{t2} \quad \ldots \quad x_{tw}\}$ and $X_{tar} = \{x_{tw+1}\}$ to represent the characteristics of N variables at historical W time steps and $W + 1$ moments respectively. Unlike traditional unsupervised anomaly detection methods, local

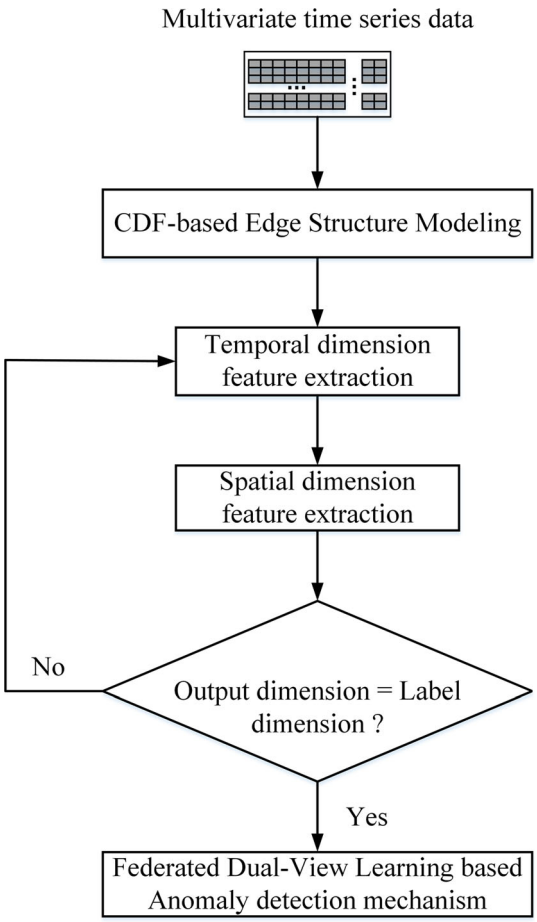


Fig. 1 Flowchart of the FDVL-DCN on each client node

models are trained separately using X_{train} and X_{tar} from normal and abnormal data, and the positive view and negative view model parameters are transmitted to the server node for federated learning-based model fusion and distribution. Test data represented as $X_{test} = \{s_{t1}, \quad s_{t2} \quad \ldots \quad s_{tw}\}$ with corresponding labels $X_{label} = \{label_{tw+1}\}$ are used to evaluate the predicted $W+1$ time step data with an anomaly detection mechanism, and the final output is a binary label indicating whether the data of each predicted time step is anomalous, where 1 represents abnormal and 0 represents normal. The evaluation metrics include *Precision*, *Recall*, and $F1\_score$. Details of the FDVL-DCN framework are elaborated upon in the following.

3.1 CDF-Based Edge Structure Modeling Approach

As the correlation between sensor characteristic values can approximately reflect the spatial dependence relationships between sensors, we first abstract different variables of multivariate time-series data in the spatial dimension into different nodes in a graph, and abstract the numerical values of each node in the temporal dimension as the current node's characteristic value. Then, we model the spatial structure of multivariate time-series data by considering three factors: **C**hange of inter-sensor dependencies, **D**ecline of time-series data impact strength, and each sensor's **F**eatures. The detailed process is as follows.

Firstly, we consider the inherent **F**eatures of different sensors and define the features of m variables on the time series t as:

$$T = \{t_1, \quad t_2 \quad \ldots \quad t_m\}, \tag{1}$$

and then normalize each variable in Equation 1 separately:

$$t_i = \frac{t_i}{MAX(t_i)} \qquad i \in (1, m). \tag{2}$$

Next, to address the issue of time-series data impact strength **D**ecline, we generate $len(t_i)$ sets of weight coefficients corresponding to different time points within the interval [0.7, 1] using the *linspace* function based on the length of the time series, and denote them as ts_weight:

$$ts\_weight = linspace(0.7, \ 1, \ len(t_i)) \qquad i \in (1, m). \tag{3}$$

The process of calculating the dependencies among different variables can be expressed as:

$$re = |T| \ * \ |T^{\top}| \ * \ ts\_weight, \tag{4}$$

where T comes from Eq. 1 normalized by Eq. 2, and ts_weisht comes from Eq. 3, expanding the weight matrix re, which is the result of the calculation of Eq. 4, can be expressed as:

$$re = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ r_{m1} & r_{m2} & \cdots & r_{mm} \end{bmatrix}. \tag{5}$$

The element r_{mm} in Eq. 5 represents the correlation coefficient between node m and node n, we filter out neighboring nodes that are weakly correlated with the current node based on the set threshold. The final adjacent nodes of each node are shown in Eq. 6. Finally, to address the issue of **C**hange of inter-sensor dependencies, we apply the above algorithm at different time points to model the spatial structure of multivariate time series data.

$$re = \begin{bmatrix} r_{1a} & r_{1b} & \cdots & r_{1c} \\ r_{2d} & r_{2e} & \cdots & r_{2f} \\ \cdots & \cdots & \cdots & \cdots \\ r_{mg} & r_{mh} & \cdots & r_{mi} \end{bmatrix} \qquad (a,b,c,d,e,f,g,h,i \in (1,m)) \tag{6}$$

Algorithm 1 describes the process of the CDF-based edge structure modeling, where X represents the multivariate time

series data input to the model at time t, $start_{edge}$ and end_{edge} are parameters used to define the starting and ending points of the edge, and $Edge_{index}$ is the spatial structure relation matrix corresponding to X. The function $splice(X, z)$ merges data X in z dimensions. The time complexity of this algorithm is $\mathcal{O}(n \log n)$, and the space complexity is $\mathcal{O}(n^2)$, where n represents the number of sensors.

Algorithm 1 CDF-based Edge Structure Modeling Approach.

Input: X
Output: $Edge_{index}$
$Edge_{index} \leftarrow$ Initialize()
$X \leftarrow$ splice $(X, (dim\_temporal))$
$X \leftarrow Row$ Normalization()
$ts\_weight \leftarrow$ linspace(0.7, 1, $len(t_i)$)
$E \leftarrow (|X| \cdot |X^{\top}| \cdot ts\_weight)$
$Edge_{init} \leftarrow \mathbf{e^{ij}} / \sum_{\mathbf{k}=1}^{\mathbf{m}} \mathbf{e^{ik}} \quad (i, j) \in (1, m)$
for $node$ in range($len(Edge_{init})$) **do**
  $Edge_{init}[node] \leftarrow$ filter $(Edge_{init}[node], threshold)$
end for
for $start_{node}$ in range $(size(Edge_{init})[0]$ **do**
  $start_{edge} \leftarrow start_{node}$
  for end_{node} in $Edge_{init}[Start_{node}]$ **do**
    $end_{edge} \leftarrow end_{node}$
    $Edge_{index}[0] \leftarrow$ Add$(start_{edge})$
    $Edge_{index}[1] \leftarrow$ Add(end_{edge})
  end for
end for
return $Edge_{index}$

In the next section, we will design the dynamic circular network for information extraction.

3.2 Dynamic Circular Network Architecture

Traditional prediction methods often consider the features of multivariate time series data only in the temporal or spatial dimension, but single-dimensional feature extraction often fails to achieve the desired predictive effect. At the same time, different lengths of sliding windows have a significant impact on the predictive performance of traditional methods. When the sliding step length is too long, the proportion of early features that are forgotten is more significant, making it difficult for the model to effectively capture these features. When the sliding step length is too short, there may not be enough information within the sliding window to form effective patterns for predicting future behavior. The selection of the sliding step length usually depends on the size of the dataset. For larger datasets, different lengths of sliding steps can be tested independently and the best one can be chosen. However, for smaller datasets, traditional methods must adopt shorter sliding steps to process the data. Our proposed DCN can simultaneously consider the feature dependencies of both temporal and spatial dimensions, while also addressing the conflict between long-term and short-term predictions. Even if a large sliding step is set for smaller datasets, DCN can cyclically extract spatio-temporal information using smaller steps. Figure 2 presents the overall architecture of DCN. Based on the length of input data in the temporal dimension($len(temporal\_len)$), *STNBlock* dynamically combines the temporal feature extraction module, spatial feature extraction module, Linear layer, Batch-Normalization (BN) layer, ReLU layer, and Dropout layer. DCN consists of several *STNBlock*, as shown in Algorithm 2. The details of the temporal and spatial feature extraction modules are described below.

Algorithm 2 Dynamic circular network architecture.

$DCN, STNBlock \leftarrow$ Initialize()
while $len(temporal\_len)$! = 1 **do**
  $STNBlock$.Add$(Bi-LSTM)$
  $STNBlock$.Add$(Soft-Attention(temporal_len-2))$
  $STNBlock$.Add$(Linear)$
  $STNBlock$.Add$(BN(temporal_len-2))$
  $STNBlock$.Add$(ReLU)$
  $STNBlock$.Add$(Bi-GCN(temporal_len-2))$
  $STNBlock$.Add$(ReLU)$
  $STNBlock$.Add$(Dropout)$
  $STNBlock$.Add$(Bi-GCN(temporal_len-2))$
  $STNBlock$.Add$(ReLU)$
  $STNBlock$.Add$(Dropout)$
  $STNBlock$.Add$(Bi-LSTM)$
  $STNBlock$.Add$(Soft-Attention(temporal_len-4))$
  $STNBlock$.Add$(Linear)$
  $STNBlock$.Add$(BN(temporal_len-4))$
  $STNBlock$.Add$(ReLU)$
  $temporal\_len = temporal\_len$ - 4
  DCN.Add$(STNBlock)$
end while
return DCN

3.2.1 Temporal Feature Extraction Module

In temporal feature extraction module, we first use Bi-LSTM to extract the temporal-dimensional features of the multivariate time series data. The specific extraction process is shown in Eq. 7:

$$\begin{aligned} i_t &= \text{sigmoid}\left(W_{\text{ii}}x_t + \text{b}_{\text{ii}} + \text{W}_{\text{hi}}h_{(\text{t}-1)} + \text{b}_{\text{hi}}\right), \\ f_t &= \text{sigmoid}\left(W_{\text{if}}x_t + \text{b}_{\text{if}} + \text{W}_{\text{hf}}h_{(\text{t}-1)} + \text{b}_{\text{hf}}\right), \\ g_t &= \tanh\left(W_{\text{ig}}x_t + \text{b}_{\text{ig}} + \text{W}_{\text{hg}}h_{(\text{t}-1)} + \text{b}_{\text{hg}}\right), \\ o_t &= \text{sigmoid}\left(W_{\text{io}}x_t + \text{b}_{\text{io}} + \text{W}_{\text{ho}}h_{(\text{t}-1)} + \text{b}_{\text{ho}}\right), \\ c_t &= \text{f}_t * \text{c}_{(\text{t}-1)} + \text{i}_t * \text{g}_t, \\ h_t &= o_t * \tanh(c_t), \end{aligned} \tag{7}$$

where x_t represents the input data at time t, W_{xx} and b_{xx} are trainable weight matrices, i_t, f_t, g_t, and o_t represent the input gate state, forget gate state, input value, and output gate state, respectively, while c_t represents the cell state at

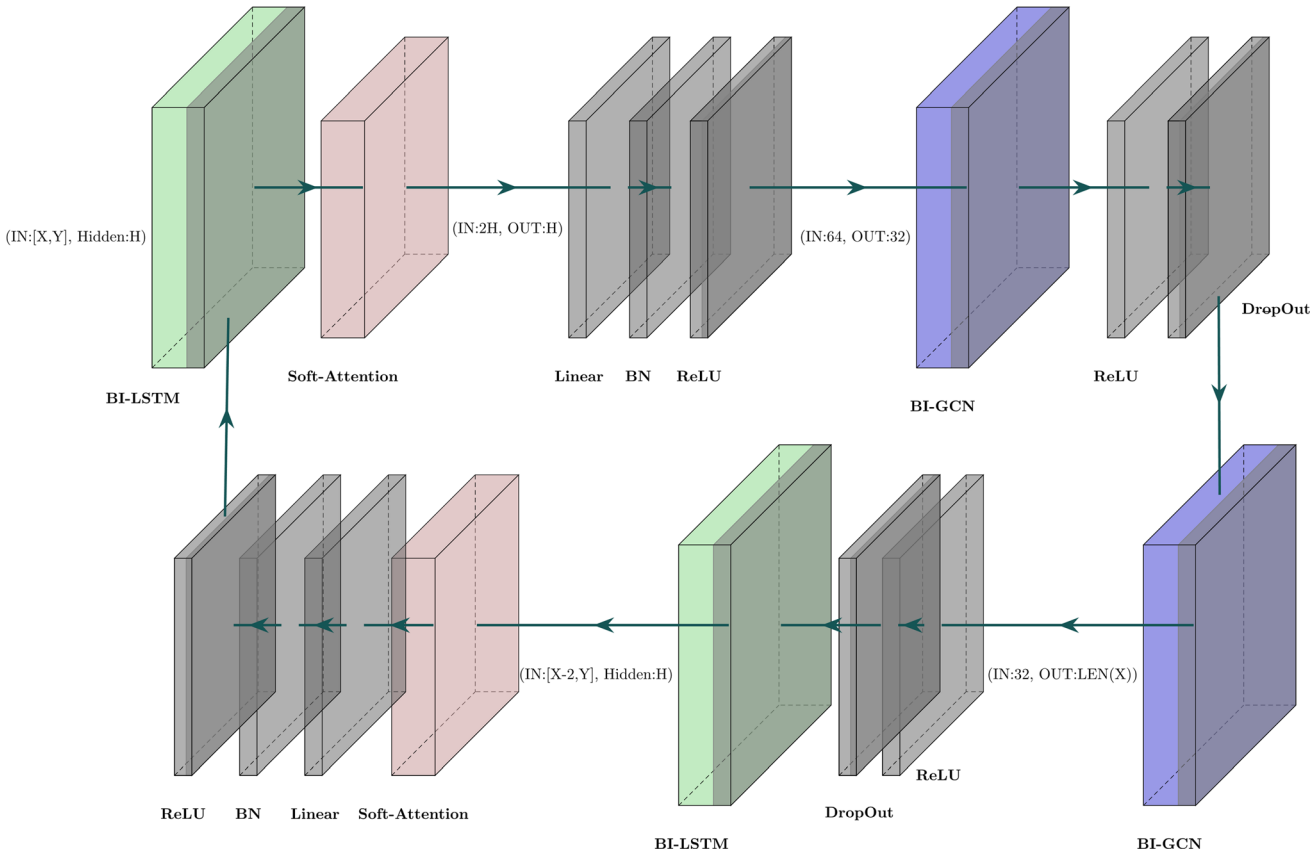


Fig. 2 Dynamic circular network architecture

time t, and h_t represents the output of the model at time t, which is a combination of input information from previous time steps up to t.

We define h as the output consisting of h_t at time period t, which can be represented as:

$$h = \{h_{t-1}, \quad h_{t-2} \quad \cdots \quad h_t\}. \tag{8}$$

Then, we apply the attention mechanism to further optimize the information h extracted in Equation 8, and the learning process can be summarized as:

$$\begin{aligned} &Set\ r^{(l)} \sim Bernoulli(p), \\ &\tilde{h} = r^{(l)} * h, \end{aligned} \tag{9}$$

$$\begin{aligned} &Set\ att_q = \tilde{h},\ att_k = att_v = h, \\ &\alpha = \left(\left(att_q * att_k{}^{\mathsf{T}}\right)/sqrt\left(\dim(att_q)\right)\right), \\ &\tilde{att} = att_v * \alpha. \end{aligned} \tag{10}$$

In Eq. 9, $Bernoulli(p)$ is a Bernoulli distribution with probability p, representing the dropout operation on the output of the LSTM. Equation 10 represents the soft attention operation. We refer to the combination of Bi-LSTM and Soft-Attention as Bi-AttLSTM. Through these operations, the features of the multivariate time-series data at time period t can be represented by $\tilde{att}$ in the temporal dimension.

3.2.2 Spatial Feature Extraction Module

Graphs are often used to represent data with "many-to-many" logical relationships and model complex relationships among different entities. Typically, Graph Neural Networks (GNN) assume that each node's state is influenced by its neighbors. By aggregating information from each node's neighbors, the model captures the spatial dependence of each node. In Eq. 11, we define the features of n nodes in the specified temporal dimension as X:

$$X = \left\{\tilde{x}_1, \quad \tilde{x}_2 \quad \cdots \quad \tilde{x}_n\right\}^{\mathsf{T}}, \tag{11}$$

when the model is trained, each input usually consists of *batch* sets of X. We stitch it in the temporal dimension, as in Eq. 12:

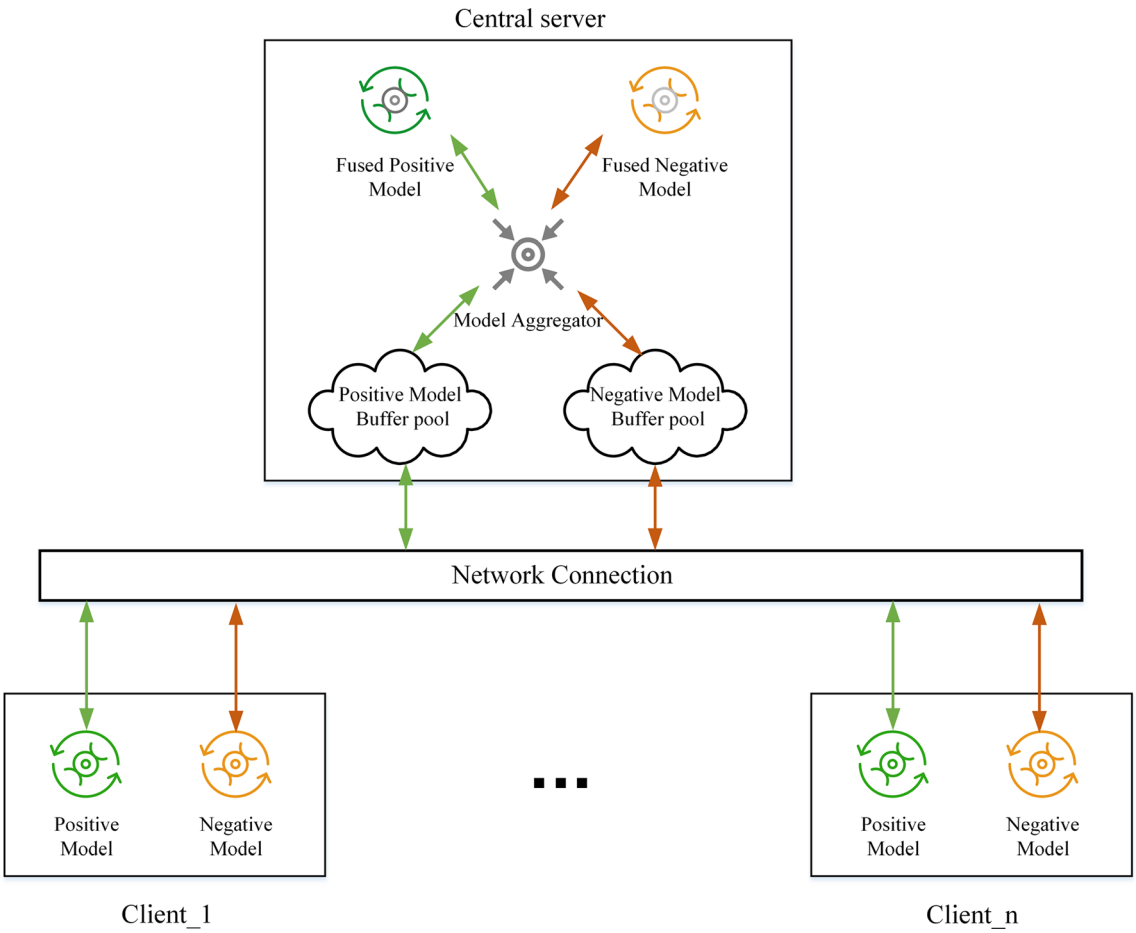


Fig. 3 Federated positive-view and negative-view model fusion strategy

$$x_i = \sum_{1}^{batch} \oplus \tilde{x}_i \qquad i \in (1, n), \tag{12}$$

where $\oplus$ denotes concatenation, and the final input data to the model can be expressed as X_{input}, as in Eq. 13:

$$X_{input} = \{x_1, \quad x_2 \quad \cdots \quad x_n\}^{\mathsf{T}}. \tag{13}$$

Classical convolutional networks are only applicable to ordered or dimensionally fixed data and cannot be applied to graph-structured data. To extend the convolution operation to graph-structured data, we must convert it from the spatial domain to the spectral domain for processing, as per graph theory and the convolution theorem. In Eq. 14, U and λ represent the eigenvectors and eigenvalues of the Laplacian matrix, respectively, corresponding to the learned graph structure:

$$\begin{aligned} U &= \left(\vec{u_1}, \quad \vec{u_2} \quad \cdots \quad \vec{u_n}\right), \\ \lambda &= (\lambda_1, \quad \lambda_2 \quad \cdots \quad \lambda_n). \end{aligned} \tag{14}$$

The essence of the Fourier inverse transform is to represent any function as a linear combination of several orthogonal basis functions. As the eigenvectors of the Laplace matrix are n linearly independent and orthogonal vectors in an n-dimensional space, they can form a set of bases. Therefore, the graph Fourier inverse transform uses the eigenvectors of the Laplace matrix as basis functions. The signal on the graph can be represented by Eq. 15, which completes the transition from the spatial domain to the spectral domain for signals on the graph.

$$X_{input} = \hat{x}(\lambda_1)\ \vec{u_1} \ + \ \hat{x}(\lambda_2)\ \vec{u_2} \ + \ \cdots \ + \hat{x}(\lambda_n)\ \vec{u_n} \tag{15}$$

the expansion of Eq. 15 can be expressed as:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \vec{u_1} & \vec{u_2} & \cdots & \vec{u_n} \end{pmatrix} \begin{pmatrix} \hat{x}(\lambda_1) \\ \hat{x}(\lambda_2) \\ \vdots \\ \hat{x}(\lambda_n) \end{pmatrix}, \tag{16}$$

where $\hat{x}$ is the representation of X_{input} in the spectral domain, for the convenience of writing, the graph Fourier inverse transform (Eq. 16) is abbreviated as:

$$X_{input} = U\hat{x}. \tag{17}$$

Inverse Eq. 17 to obtain the graph Fourier positive transform (Eq. 18) as follows:

$$\hat{x} = U^{\intercal} X_{input}. \tag{18}$$

Let the convolution kernel be represented as g in the spatial domain and $\hat{g}$ in the spectral domain, then the convolution operation in the spectral domain can be defined as:

$$\Gamma(x) \otimes \Gamma(g) = \hat{x} \otimes \hat{g} = U^{\intercal} x \otimes U^{\intercal} g, \tag{19}$$

in Eq. 19, $\otimes$ is the *hadamard* product, and Γ is the graph Fourier positive transform. The convolution result in the spectral domain is inverted to obtain the convolution result in the spatial domain. The convolution of x with the convolution kernel g is denoted by $x *_G g$. Equation 20 shows the complete convolution operation in the spatial domain,

$$x *_G g = \Gamma^{-1}(\Gamma(x) \otimes \Gamma(g)) = U(U^{\intercal} x \otimes U^{\intercal} g) \tag{20}$$

where Γ^{-1} is the graph Fourier inverse transform, by the above operation, the multivariate time-series data can be convolved in the spatial dimension.

3.3 Architecture Design of FDVL-DCN

Federated learning is comprised of two main components: a central server and clients. The central server initiates a machine learning job, such as specifying the model architecture, and coordinates the federated learning process. Meanwhile, clients use their local data and computational resources to train the specified local models. The structure of the federated positive view and negative view model fusion mechanism is illustrated in Fig. 3, which follows a specific workflow.

To start, a job creator initializes the model training job, setting the fusion policy and number of iteration rounds, and configures an initial wait time for a client to return model updates. After successful initialization, the central server exposes its IP address and port number for the running federated job service to the client. Each participating client accesses the resources of the central server via IP address and port number to download the job (*Job*), the initial model structure (*ModelArchitecture*), and the sliding window's temporal dimension step (*Temporal*). Additionally, the multivariate time-series data collected by local sensors are preprocessed by the data preprocessor in the client (*OriginalFeatures*), which includes tasks such as dimensionality reduction and missing value processing.

The client part of Algorithm 3 describes in details the model training and model interaction process.

Algorithm 3 Federated Dual-View Learning for Dynamic Circular Network.

Client Part
Input: $Job, ModelArchitecture, Temporal, OriginalFeatures$
Output: $Loss$
$FedStep \leftarrow 0$
while $FedStep < LocalEpoch$ **do**
  $EdgeIndex \leftarrow$ CDF-SpatialStructureModeling($OriginalFeatures$)
  while $Len(Temporal) > 1$ **do**
    ConstructDCNArchitecture
  end while
  $Loss, PositiveModel, NegativeModel, TrainingTime \leftarrow$
train($EdgeIndex, ModelArchitecture$)
  send($TrainingTime$)
  if $Loss(PositiveModel)$ or $Loss(NegativeModel) <$ previous version **then**
    uploadRequest($PositiveModel$ or $NegativeModel$)
    waitingForFusion()
    ($PositiveModel$ or $NegativeModel$) $\leftarrow$ receiveGlobalModel()
  else
    noUploadRequest($PositiveModel$ or $NegativeModel$)
  end if
  $FedStep++$
end while
return $Loss$

Server Part
Input: $TrainingTime$
while true **do**
  $WaitingTime \leftarrow$ update($TrainingTime$)
  $ClientModel \leftarrow$ AccessModelBufferPool()
  $S_t \leftarrow$ (completed training set of m clients)
  if expired($WaitingTime$) $== true$ **then**
    $(ClientModel\ ,\ t+1) \leftarrow \sum_{k \in S_t} (FedAvg(ClientModel, t))$
    for each client $k \in S_t$ **in parallel do**
      dispatch($(ClientModel\ ,\ t+1)$)
    end for
  end if
end while

Firstly, the cleaned data is passed through the CDF-based edge structure modeling module to construct the initial spatial structure (*EdgeIndex*) of the multivariate time-series data.

Secondly, each client builds the DCN architecture based on the length of the sliding window it receives from the server, then trains the positive view (*PositiveModel*) and negative view (*NegativeModel*) model separately through the model trainer. After completing the specified training rounds, the model trainer uploads the training time (*TrainingTime*) to the central server.

After each round of training, each client evaluates the model trained in the current round and the previous version of the model by comparing the magnitude of the Loss value during the training process. If the model trained in the current round performs better (*Loss*), the client sends a model upload request to the central server and waits for the global model sent down by the server to update the local model. Otherwise, the client will request not to upload the model update for this round. Clients that do not complete the specified number of iteration rounds within the current wait time are not allowed to participate in the current round of aggregation.

The server part of Algorithm 3 describes in details the process of model processing. Firstly, the aggregation scheduler updates the waiting time (*WaitingTime*) based on the training time (*TrainingTime*) received from participating clients. Secondly, after the set waiting time, the central server accesses the positive view and negative view model in the buffer pool and completes the weight aggregation of the positive view and negative view model using the FedAvg (McMahan et al. 2017) algorithm. Finally, the aggregation scheduler on the central server sends the aggregated global model back to each client for a new round of training.

The above process is iterated until the specified training rounds are completed or the model reaches certain recognition accuracy.

3.4 Anomaly Evaluation Mechanism

The goal of our work is to predict whether the state values are abnormal. The predicted and actual values of the *m-dimensional* multivariate time-series data at the moment t are given in Equation 21:

$$\begin{aligned} y\_predict &= \{y'_{t1}, \quad y'_{t2} \quad \cdots \quad y'_{tm}\}, \\ y\_true &= \{y_{t1}, \quad y_{t2}, \quad \cdots \quad y_{tm}\}. \end{aligned} \tag{21}$$

To evaluate the error between the predicted and actual values, we introduce the MSE function, as in Eq. 22:

$$\frac{1}{m}\sum_{i=1}^{m}\left(y_{ti} - y'_{ti}\right)^2. \tag{22}$$

When applying federated learning to supervised anomaly detection, it is a pressing issue to enable all participants to find the critical threshold that suits their own devices for distinguishing normal and anomalous data. Therefore, during the training of the federated positive view and negative view models, we recorded the loss intervals corresponding to the positive view and negative view model, respectively. The loss interval corresponding to the positive view model was obtained by training all normal data. The loss interval corresponding to the negative view model was obtained by training all abnormal data. We consider that there is a threshold within the merged set of these two intervals such that the loss greater than this threshold is the category to which the right endpoint of the merged interval belongs, and the loss less than this threshold is the category to which the left endpoint of the merged interval belongs (in our training process, there is no case that one interval contains another interval or the two intervals overlap). Therefore, after we merge the two loss intervals, we take 10-50 values of equal distance within the merged interval as the reference threshold. Then the test set is input into the federated positive view model, and the corresponding F1 values under each threshold are recorded separately. Finally, the corresponding threshold with the highest F1 value is set as the best threshold. The significance of training federated negative view model in this process is to obtain the LOSS range of abnormal data, so as to combine with the LOSS range of federated positive view, and find the best segmentation point to distinguish normal and abnormal data.

3.5 FDVL-DCN Algorithm As a Software Service

To simplify the use of FDVL-DCN, we encapsulate the main modules as software services using the concept of big data as a service. The dependencies among various modules are depicted in Fig. 4 utilizing a package diagram, where FL stands for Federated Learning, FDVL stands for Federated Dual-View Learning, and DCN stands for Dynamic Circular Network. Specifically, the spatial structure modeling service and DCN service help to initialize suitable models, while the FDVL service improves model performance while maintaining data privacy. The anomaly detection service can help to avoid anomalies and reduce losses, and the specific application scenarios are in the domain service. This design ensures efficient data analysis and enables users to make prompt decisions using FDVL-DCN.

4 Evaluations

This section introduces the experimental details, including data sources, experimental setup, and evaluation metrics. Then, we evaluate our model with the baseline model. Moreover, we assess the generality of our model and the function of each component in our model.

4.1 Datasets

The proposed model was evaluated on four datasets: a revised version of the KDD'99 dataset, NSL-KDD,[1] the Safe Water Treatment dataset, SWaT,[2] the Curiosity Mars Rover spacecraft telemetry dataset, MSL,[3] and the Semiconductor Microelectronics Manufacturing dataset, Wafer.[4] A summary of the dataset statistics is provided in Table 1. More information about datasets is as follows.

NSL-KDD The dataset was used in the 1999 KDD competition and includes 43 features. The first 9 columns represent essential features of TCP connections, while columns 10 to 22 contain content features of these connections. Columns 23 to 31 present time-based network traffic statistics based on a 2-second time window, and columns 32 to 41 show host-based network traffic statistics. The last two columns indicate if the traffic is normal or an

[1] https://www.unb.ca/cic/datasets/nsl.html (accessed 21 Dec 2022).

[2] https://drive.google.com/drive/folders/1ABZKdclka3e2NXBSxS9z2YF59p7g2Y5I?usp=sharing (accessed 26 April 2022).

[3] https://github.com/d-ailin/GDN/tree/main/data/msl (accessed 15 Mar 2022).

[4] https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (accessed 23 April 2022).

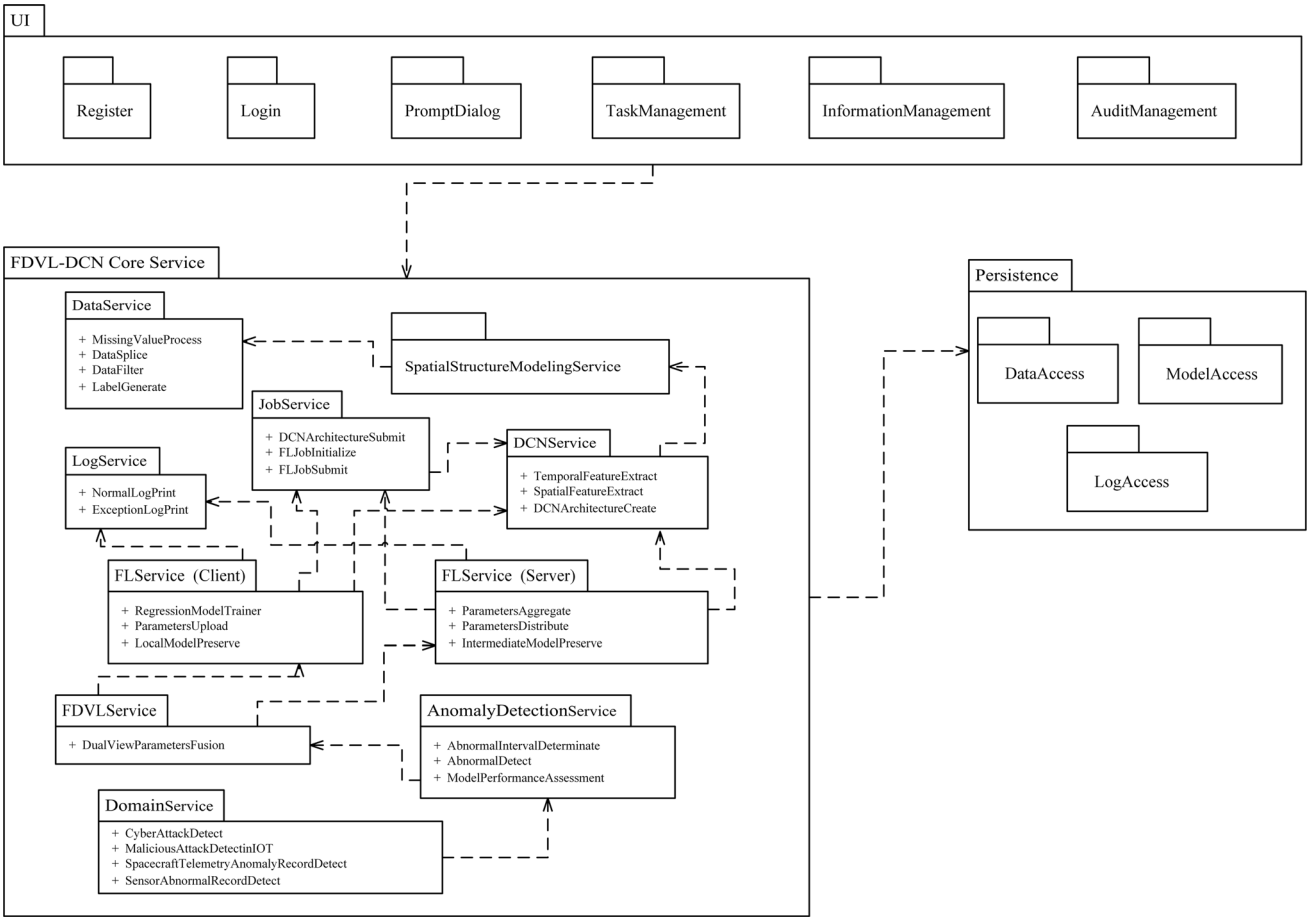


Fig. 4 The dependency relationships between different software services in the FDVL-DCN algorithm

Table 1 Statistics of the four datasets used in experiments

Datasets	Features	Train	Test	Anomaly ratio (%)
NSL-KDD	41	67343	22544	23.1
SWaT	51	496800	449919	12.1
MSL	27	1566	2050	29.8
Wafer	152	6164	1000	9.7

attack and provide a score indicating the severity of the traffic input.

SWaT The Singapore Public Utilities Commission developed the Water Treatment Testbed dataset, which represents a scaled-down version of a modern cyber-physical system. It combines digital and physical elements to monitor and control system behavior, making it ideal for critical areas such as power plants and the Internet of Things (IoT) that require protection against malicious attacks. The dataset contains 51 features, with one sensor data generated every second. The training set covers the period from December 22, 2015, at 16:00, to December 28, 2015, at 10:00, while the test set spans from December 28, 2015, at 10:00, to January 2, 2016, at 15:00.

MSL The MSL dataset contains real spacecraft telemetry and anomalies recorded by the Curiosity Mars rover. All data has been anonymized based on timestamps, with all telemetry values pre-scaled between (-1, 1) using minimum/maximum values in the test. Additionally, channel IDs have been anonymized, with the first letter indicating the channel type (e.g., P denotes power, R denotes radiation). Despite these measures, the dataset is still useful for research purposes.

Wafer The Wafer dataset comprises measurement data collected by various sensors during the processing of silicon wafers for semiconductor manufacturing. Domain experts analyze the data in real-time and assign a label of either normal or abnormal at each moment. The dataset includes 7164 records with 152 features.

4.2 Evaluation Metrics

Data imbalance is a common issue in anomaly detection because equipment typically operates correctly. To

comprehensively evaluate the model, we employ *Precision*, *Recall*, and *F1_Score* metrics, as shown in Equation 23, to assess the performance of our method and the baseline model:

$$\begin{aligned} precision &= \frac{TP}{TP+FP}, \\ recall &= \frac{TP}{TP+FN}, \\ F_1 &= \frac{2TP}{2TP+FP+FN}, \end{aligned} \tag{23}$$

where TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) are the number of true positives, true negatives, false positives, and false negatives separately. In practical application scenarios, a higher precision rate helps to operate more efficiently, as a low percentage of false positives helps to reduce the overall system inspection cost. On the other hand, a higher recall rate indicates that abnormalities can be captured more efficiently, which allows one to respond and handle anomalies before they occur.

4.3 Experimental Setup

We implemented our approach in PyTorch version 1.8.1 and CUDA version 11.1 and experimented with different tasks based on the GFL Hu et al. (2020) and a self-written federated learning framework, and the software finally packaged with Docker. The specific details are shown in Table 2.

The original data is processed by sliding window and sliding step. The features in the sliding window are extracted in spatial and temporal dimensions, and the next row of data in the sliding window is used as the label. Then, the sliding window and label are sequentially shifted back into the sliding step. The extracted features and labels are trained with mean square error (MSE) and SGD optimizer in the federated scenario. The data is split so that each client's data satisfies non-independent identical distribution (Non-IID). Table 3 shows the proportion of positive and negative labels of the dataset in each client node.

Table 3 Proportion of positive and negative labels in different client nodes

Client	Datasets			
	NSL-KDD	SWaT	MSL	Wafer
Client1	3.9:1	7.3:1	2.0:1	9.3:1
Client2	3.1:1	5.1:1	1.5:1	7.5:1
Client3	2.6:1	4.8:1	1.2:1	6.9:1

4.4 Baselines

We compare the performance of our proposed method with ten popular anomaly detection methods, as follows.

- *GDN* This method uses graph neural networks to model multivariate time-series data and proposes a graph deviation scoring mechanism to detect anomalies. Deng and Hooi (2021)
- *PCA* Principal Component Analysis finds a low-dimensional projection that captures most of the variance in the data. The anomaly score is the reconstruction error of this projection. Shyu et al. (2003)
- *KNN* K Nearest Neighbors uses each point's distance to its kth nearest neighbor as an anomaly score. Hautamäki et al. (2004)
- *Sampling* This method takes a small set of samples from a given set of objects, followed by measuring the outliers of each object by the distance from the object to its nearest neighbor in the sample set. Sugiyama and Borgwardt (2013)
- *HBOS* HBOS computes each feature dimension as a separate histogram, using an improved discrete plain Bayesian probability model to detect anomalies. Goldstein and Dengel (2012)
- *IForest* Isolation Forest constructs an *itree* set for a given dataset and marks the instances on the *itree* with shorter average path lengths as exceptions. Liu et al. (2008)

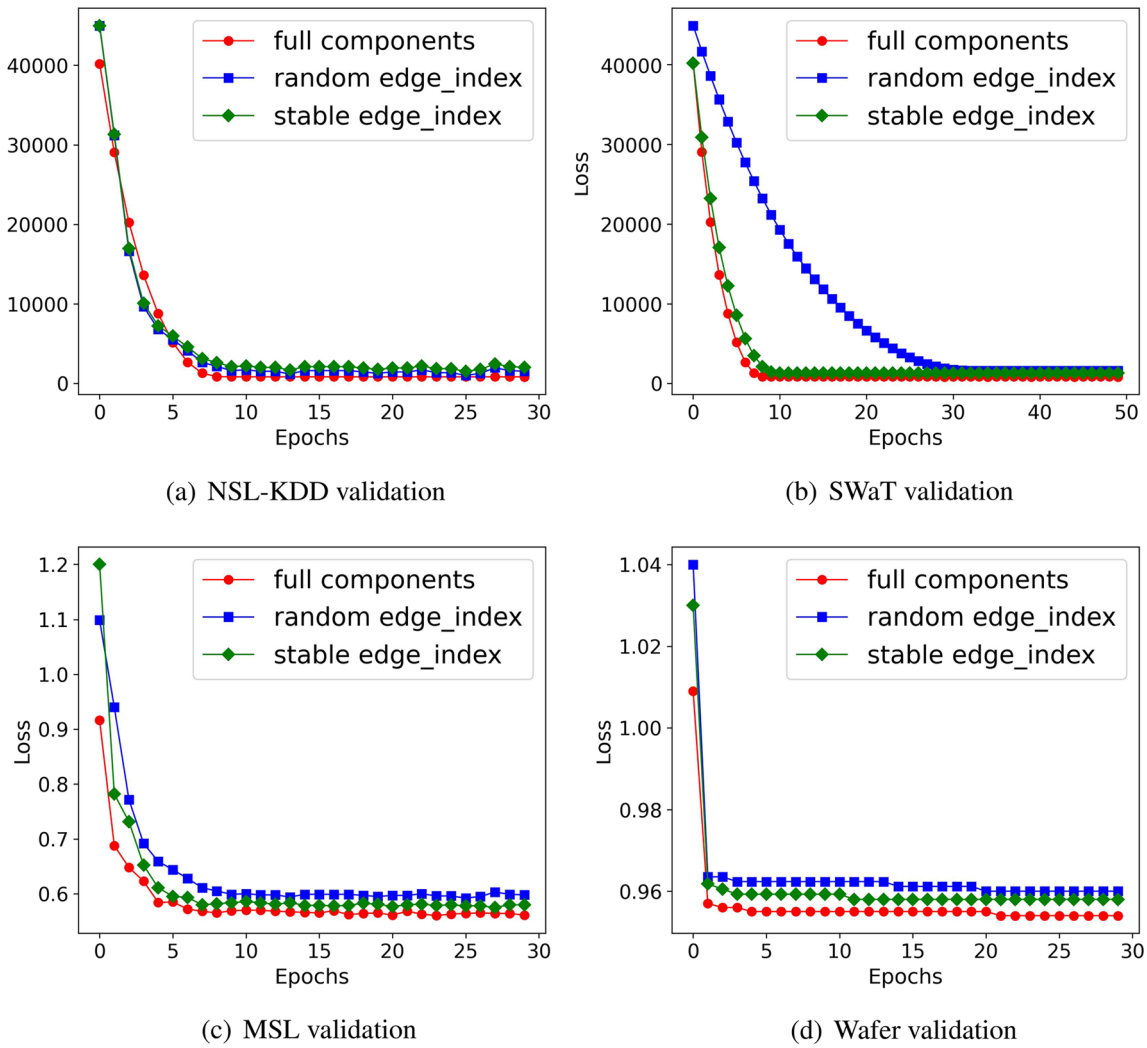
Table 2 Details of the hardware and software parameters used in the experiment

Parameters	Details
Server	Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz, GeForce RTX 3080
Client	Intel(R) Core(TM) i7-10700 CPU @ 3.80GHz, GeForce RTX 3090
Learning rate	1×10^{-2}
The number of training rounds	100
Batch sizes	NSL-KDD(40), SWaT(250), MSL(15), Wafer(80)
Topk	NSL-KDD(200), SWaT(300), MSL(100), Wafer(1000)
Sliding steps	NSL-KDD(8), SWaT(1), MSL(1), Wafer(4)

Table 4 Performance comparison of DCN under different edge structure modeling methods

Method	Datasets											
	NSL-KDD			SWaT			MSL			Wafer		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Normal DCN	**1.000**	**0.615**	**0.766**	**0.613**	**0.949**	**0.744**	**1.000**	**0.775**	**0.873**	**1.000**	**0.902**	**0.949**
Randomly initialized graph structure	1.000	0.591	0.712	0.588	0.926	0.701	0.501	0.503	0.498	1.000	0.375	0.545
Stable and invariant graph structure	1.000	0.588	0.711	0.601	0.933	0.729	0.497	0.504	0.501	1.000	0.429	0.600

Bold font represents the superiority of the proposed DCN compared to traditional methods in terms of three evaluation metrics across four datasets

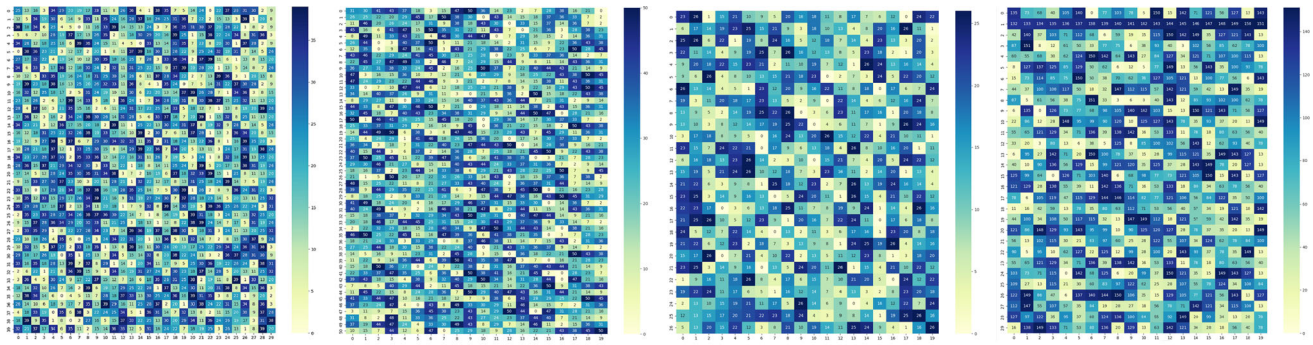


(a) NSL-KDD validation (b) SWaT validation

(c) MSL validation (d) Wafer validation

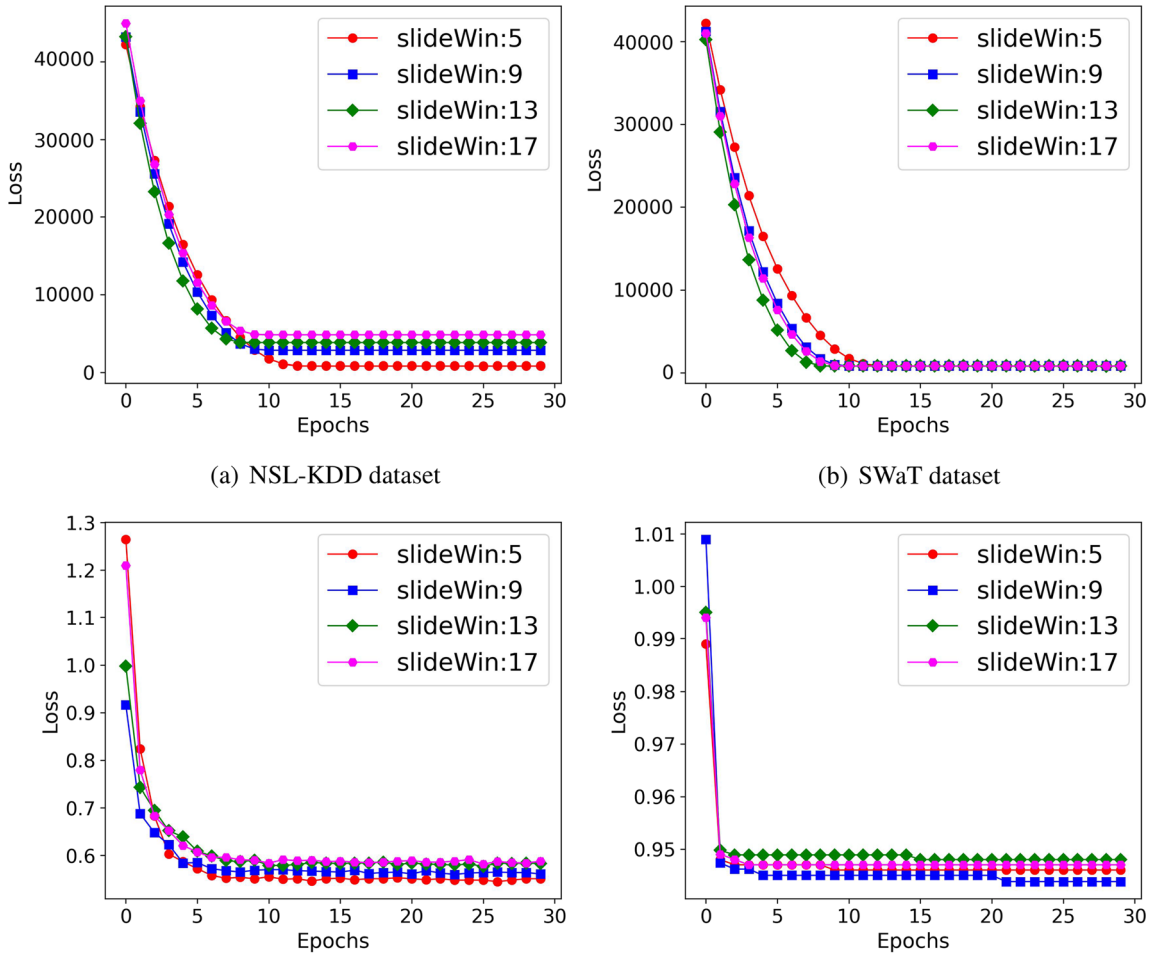
Fig. 5 Loss curves of different edge structure modeling methods on NSL-KDD, SWaT, MSL, and Wafer

- *ECOD* ECOD uses empirical cumulative distribution functions for outlier detection. Li et al. (2022)
- *MO_GAAL* This method uses multiple generators and discriminators to learn the data distribution and detect anomalies, respectively (Liu et al. 2020).
- *Beta-VAE* Beta-VAE facilitates understanding the entangled representation by controlling the increase in potential posterior coding power during training combined with better reconstruction fidelity. Burgess et al. (2018)
- *DeepSVDD* This method trains a neural network to map most data into a hypersphere with center c and radius R as the minimum volume. Examples that do not belong



(a) NSL-KDD (b) SWaT (c) MSL (d) Wafer

Fig. 6 Results of CDF-based edge structure modeling for some time periods in the four datasets



(a) NSL-KDD dataset (b) SWaT dataset

(c) MSL dataset (d) Wafer dataset

Fig. 7 Loss curves of different sliding windows on NSL-KDD, SWaT, MSL, and Wafer

Table 5 Performance comparison of DCN with different sliding windows

Length	Datasets											
	NSL-KDD			SWaT			MSL			Wafer		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
slideWin:5	1.000	0.634	0.766	0.620	0.965	0.757	1.000	0.774	0.871	1.000	0.889	0.941
slideWin:9	1.000	0.598	0.749	0.613	0.949	0.744	1.000	0.775	0.873	1.000	0.902	0.949
slideWin:13	1.000	0.579	0.742	0.612	0.949	0.744	1.000	0.775	0.873	1.000	0.943	0.958
slideWin:17	1.000	0.633	0.776	0.603	0.966	0.743	1.000	0.771	0.870	1.000	0.889	0.941
slideWin:21	1.000	0.593	0.716	0.581	0.956	0.723	1.000	0.756	0.853	1.000	0.879	0.933

Table 6 Percentage decrease in LOSS for dynamic circular network architecture compared to normal architecture

Architecture	Datasets			
	NSL-KDD (%)	SWaT (%)	MSL (%)	Wafer (%)
slideWin:3 (No circulation)	2.3	96.7	25.7	4.1
slideWin:9 (No circulation)	2.3	98.0	25.6	4.1

to this hypersphere are marked as anomalies. Ruff et al. (2018)

4.5 Experiment Results

4.5.1 Validation of CDF-Based Edge Structure Modeling

To evaluate the effectiveness of our CDF-based edge structure modeling approach, we made modifications to DCN in the following ways.

- Replacing the CDF-based edge structure modeling module with randomly initialized graph structures.
- Replacing the CDF-based edge structure modeling module with stable and invariant graph structures(Edge structure modeling method in Deng and Hooi (2021)).

The randomly initialized graph structure is equivalent to randomly generating different neighbor nodes for each sensor node at different moments. However, not all of these neighbor nodes are associated with the current sensor node, so in this case, the sensor nodes learn a large fraction of noise weights, and the presence of this noise guides the model to train in a direction that deviates from the objective function, which in turn produces large deviations in the current sensor's prediction of future moment behavior. Although the stable and invariant graph structure learns the dependencies of different sensor nodes at the initial moment, as described in Sect. 1, the method is not very different from the random initialized graph structure when faced with the topology that changes over time. Our proposed CDF-based edge structure modeling approach can model the inter-sensor topology at different moments. A correct and appropriate graph structure enables the GNN to play a positive role in the model training process, which helps the model predict the behavior at future moments with low LOSS values. As seen in Table 4 and Fig. 5, the CDF-based edge structure modeling approach dominates the anomaly detection evaluation metrics for all four datasets compared to the two methods mentioned above, while the models using this approach have faster convergence and smaller LOSS values, which validates the necessity of this component.

Figure 6 displays the edge structures learned by our proposed CDF-based edge structure modeling method for NSL-KDD, SWaT, MSL, and Wafer datasets at different time nodes. The computation time for modeling the edge structure on an NVIDIA Geforce RTX 3090 device is 0.4 s for NSL-KDD, 0.45 s for SWaT, 0.02 s for MSL, and 3 s for Wafer when the sliding window length is 9. Each vertical number in the figure corresponds to a sensor, and each horizontal column represents the *topk* sensors that are associated with the current sensor.

4.5.2 Generalization Capabilities of the Dynamic Circular Network Architecture

To verify the generalization capabilities of the dynamic circular network architecture proposed in the above

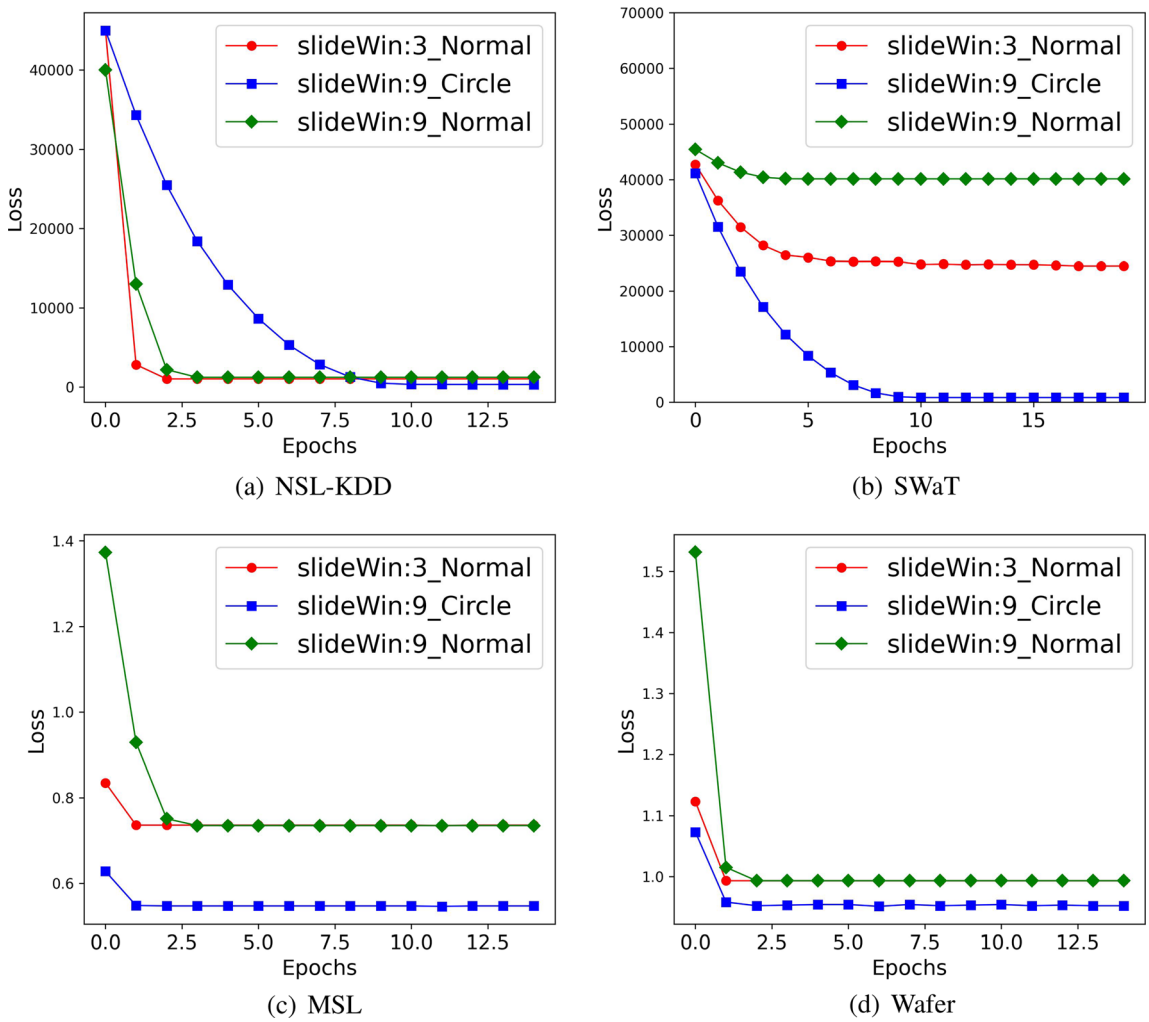
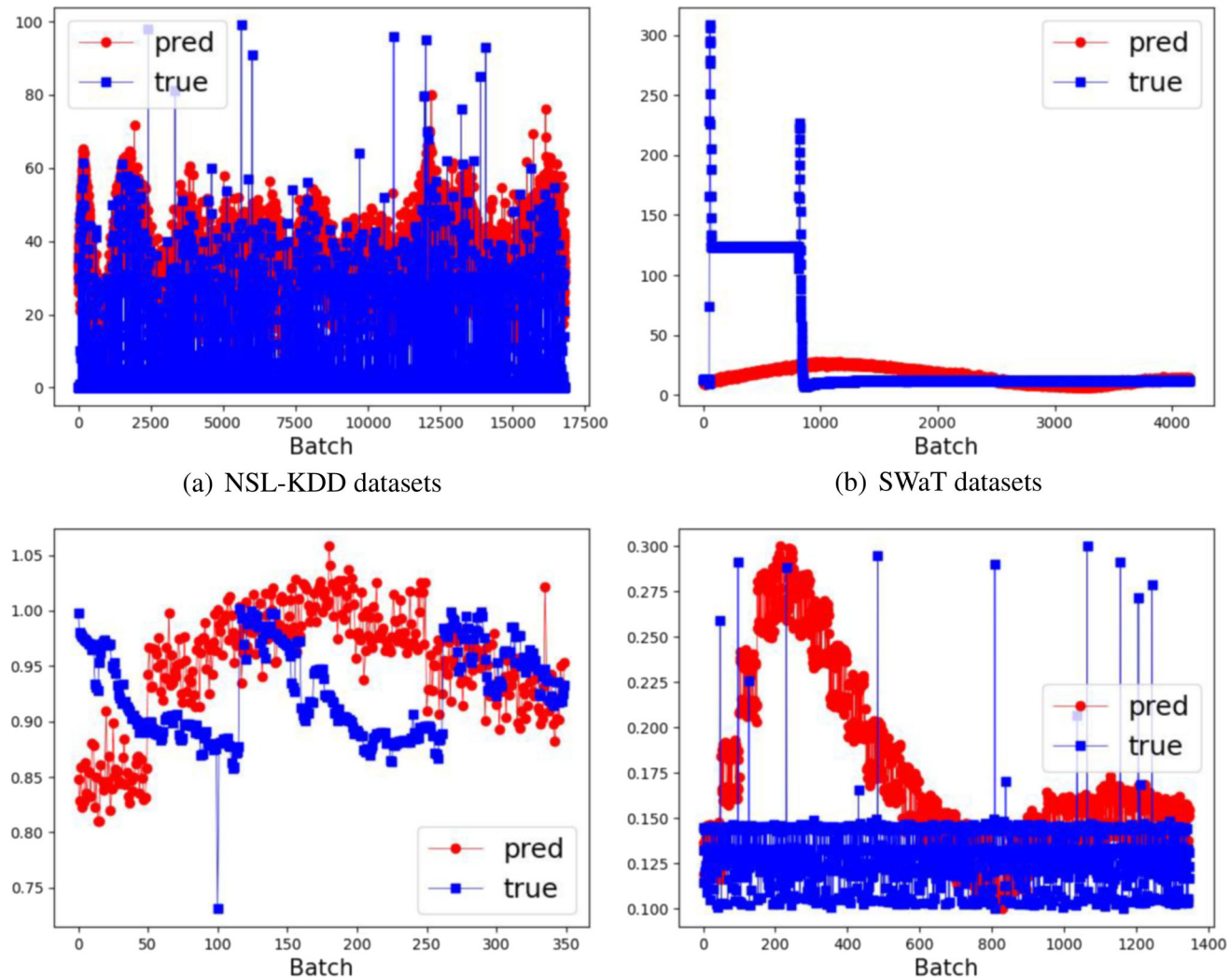


Fig. 8 Loss curves of different model architecture on NSL-KDD, SWaT, MSL, and Wafer

method, we used different lengths of sliding windows to extract the spatio-temporal features of the multivariate time-series data. The sampling frequencies are set to 5, 9, 13, 17 and 21. Figure 7 shows the convergence speed and trend of the four datasets at different sampling frequencies. This verifies the effectiveness and generality of the dynamic circular network architecture. The final anomaly detection results at different frequencies are shown in Table 5, which shows that the differences in the evaluation metrics on the four data sets are not significant at the first four sampling frequencies. However, when the sampling frequency is 21, the three evaluation metrics on the four data sets are reduced, but the effect is still better than most baseline methods.

4.5.3 Effectiveness of the Dynamic Circular Network Architecture

To validate the effectiveness of our proposed dynamic circular network architecture, we chose the network architecture without the cyclic idea for comparison experiments, where sliding step lengths were set to 3 and 9, respectively. As stated in Sect. 3.2, smaller sliding steps are insufficient to extract effective behavioral patterns. Bigger sliding steps ignore the behavioral features at earlier moments. The two traditional network architectures have higher loss values than our proposed DCN on the four datasets, with the most significant difference in loss on the swat dataset. The specific decline rates are shown in Table 6, which further validates the effectiveness of the dynamic circular network architecture. Figure 8 illustrates the experimental results.



(a) NSL-KDD datasets (b) SWaT datasets

(c) MSL datasets (d) Wafer datasets

Fig. 9 Comparison of actual and predicted values

Table 7 Impact of sliding steps on assessment metrics

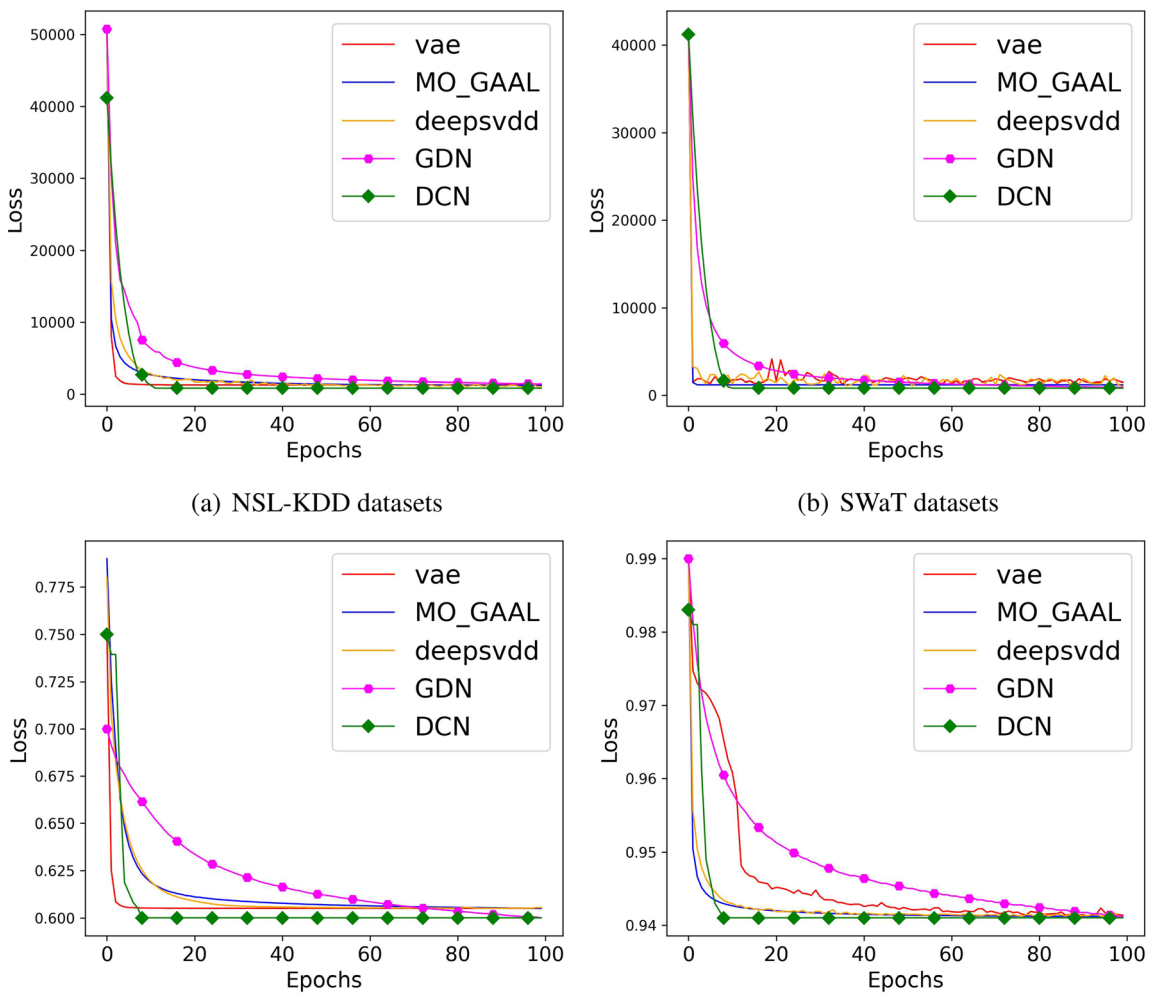
Length	Datasets											
	NSL-KDD			SWaT			MSL			Wafer		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
slideStride:1	0.992	0.572	0.725	0.603	0.952	0.738	1.000	0.736	0.853	1.000	0.889	0.940
slideStride:5	1.000	0.598	0.749	0.613	0.949	0.744	1.000	0.754	0.859	1.000	0.878	0.938
slideStride:8	1.000	**0.661**	**0.796**	0.597	1.000	0.737	1.000	0.732	0.850	1.000	**0.891**	**0.960**
slideStride:10	1.000	0.625	0.769	**0.667**	1.000	**0.800**	1.000	0.726	0.848	1.000	0.874	0.920

Bold font highlights the extreme values of different evaluation metrics for each dataset at different sliding window sizes (multiple occurrences of the same extreme value for different evaluation metrics within a dataset are not shown in bold)

Table 8 Anomaly detection results of different models on NSL-KDD, SWaT, MSL, and Wafer

Method	Datasets											
	NSL-KDD			SWaT			MSL			Wafer		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
PCA	0.726	0.520	0.610	0.153	0.906	0.261	0.781	0.800	0.867	0.037	0.031	0.034
KNN	0.805	0.494	0.612	0.235	0.683	0.392	0.843	0.956	0.831	0.588	0.783	0.695
Sampling	0.673	0.122	0.207	0.678	0.701	0.690	0.852	0.726	0.801	0.508	0.901	0.634
HBOS	0.858	0.546	0.671	0.313	0.742	0.441	0.781	0.976	0.870	0.103	0.113	0.108
ECOD	0.870	0.562	0.687	0.587	0.709	0.642	0.837	0.311	0.453	0.339	0.412	0.372
MO_GAAL	0.153	0.906	0.261	0.153	0.896	0.251	0.773	0.972	0.841	0.293	0.311	0.302
IForest	0.731	0.534	0.621	0.297	0.772	0.394	0.827	0.733	0.801	0.548	0.803	0.676
Beta-VAE	0.776	0.570	0.660	0.153	0.906	0.261	0.781	0.868	0.871	0.304	0.323	0.312
DeepSVDD	0.791	0.313	0.449	0.379	0.495	0.261	0.781	0.969	0.861	0.386	0.629	0.478
GDN	0.570	1.000	0.725	0.956	0.623	0.740	0.813	0.976	**0.873**	0.128	0.589	0.210
DCN	**1.000**	0.598	0.749	0.613	**0.949**	**0.744**	**1.000**	0.775	**0.873**	**1.000**	**0.902**	**0.949**
FDVL-DCN	**1.000**	0.661	**0.796**	**0.837**	0.828	**0.833**	**1.000**	0.583	0.737	0.964	0.739	0.809

Bold font is used to emphasize the superior performance of the proposed DCN and FDVL-DCN compared to the other ten baseline algorithms in terms of different evaluation metrics across four datasets



(a) NSL-KDD datasets (b) SWaT datasets

(c) MSL datasets (d) Wafer datasets

Fig. 10 Loss curve under different models on NSL-KDD, SWaT, MSL, and Wafer

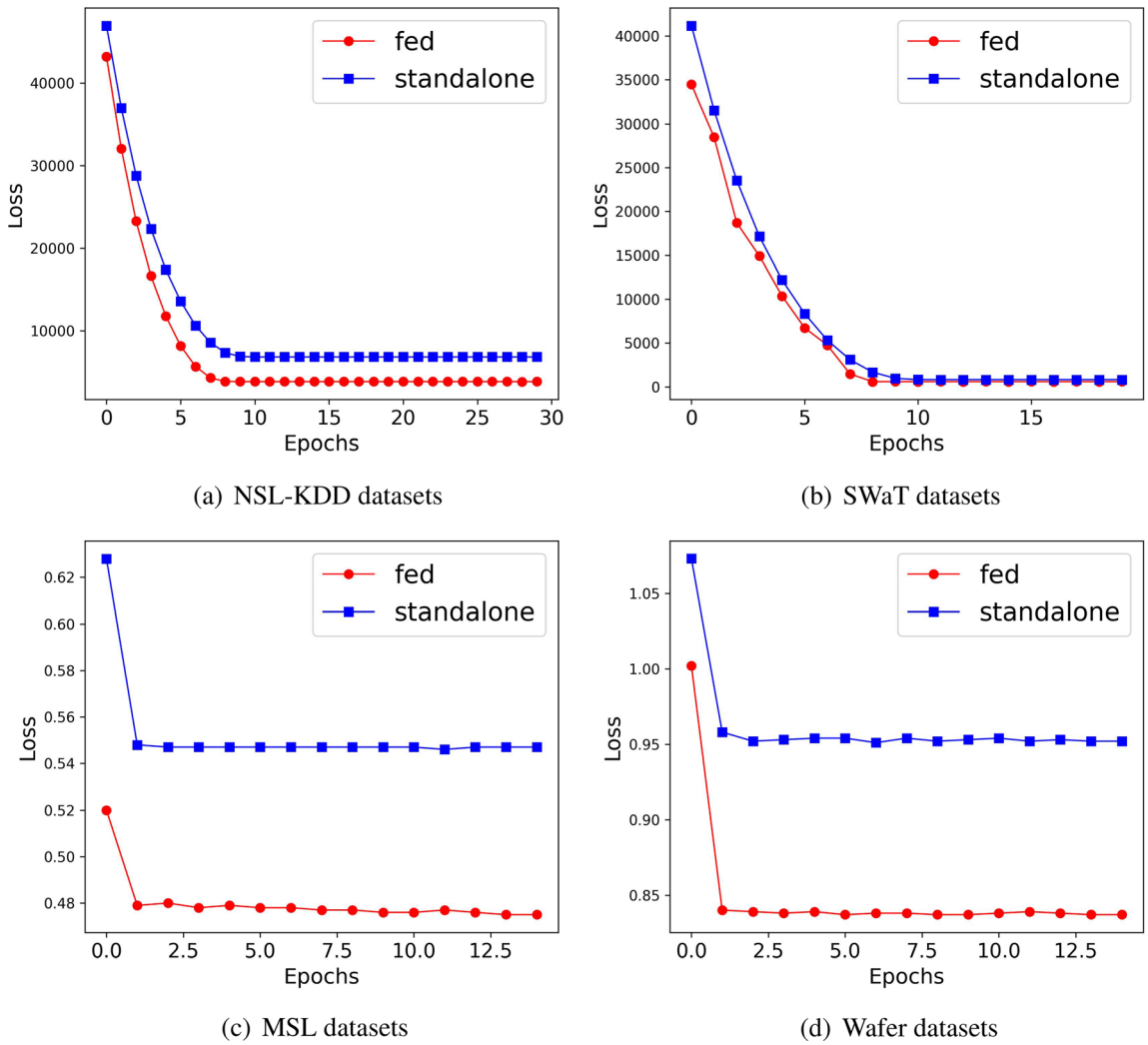


Fig. 11 Loss curve under different scenarios

Table 9 Experimental details of the federated dual-view learning and federated single-view learning

Datasets	Details				
	batch	topk	slide_stride	local_eopch	number of fusions
NSL-KDD	40	200	4	100	3
SWaT	240	100	1	100	3
MSL	15	30	1	20	3
Wafer	80	1000	4	20	3

Table 10 The ratio of increase in time consumption of federated dual-view learning compared to federated single-view learning(in seconds)

Methods	Datasets			
	NSL-KDD	SWaT	MSL	Wafer
FDVL	520 (↑4%)	116210 (↑2.5%)	80 (↑7.5%)	1740 (↑1.6%)
FDVL	500	113410	74	1713

Table 11 The improvement ratio of anomaly detection F1 values of federated dual-view learning compared to federated single-view learning

Methods	Datasets			
	NSL-KDD	SWaT	MSL	Wafer
FDVL (F1)	0.766 (↑7.4%)	0.833 (↑23.8%)	0.737 (↑17.2%)	0.809 (↑61.8%)
FPVL (F1)	0.713	0.673	0.629	0.5

Table 12 Percentage improvement in F1 values for DCN and FDVL-DCN compared to conventional methods

Method	Datasets							
	NSL-KDD		SWaT		MSL		Wafer	
	DCN (%)	FDVL-DCN (%)	DCN (%)	FDVL-DCN (%)	DCN (%)	FDVL-DCN (%)	DCN (%)	FDVL-DCN (%)
PCA	↑22.8	↑30.5	↑185.1	↑219.2	↑0.7	↓15.0	↑2691.2	↑2279.4
KNN	↑22.39	↑30.1	↑89.8	↑112.5	↑5.1	↓11.3	↑36.5	↑16.4
Sampling	↑261.8	↑284.5	↑7.8	↑20.7	↑9.0	↓8.0	↑49.7	↑27.6
HBOS	↑11.6	↑18.6	↑68.7	↑88.9	↑0.3	↓15.3	↑778.7	↑649.1
ECOD	↑9.0	↑15.9	↑15.9	↑29.8	↑92.7	↑62.7	↑155.1	↑117.5
MO_GAAL	↑187.0	↑205.0	↑196.4	↑231.9	↑3.8	↓12.4	↑214.2	↑167.9
IForest	↑20.6	↑28.2	↑88.9	↑111.4	↑9.0	↓8.0	↑40.4	↑19.7
Beta-VAE	↑13.5	↑20.6	↑185.1	↑219.2	↑0.2	↓15.4	↑204.2	↑159.3
DeepSVDD	↑66.8	↑77.3	↑185.1	↑219.2	↑1.4	↓14.4	↑98.5	↑69.2
GDN	↑3.3	↑9.8	↑0.5	↑12.6	↑0.0	↓15.6	↑351.9	↑285.2

4.5.4 Validation of Model Prediction Accuracy

To test the prediction effectiveness of the proposed method for multivariate time-series data, we plotted the comparison curves of predicted and actual values of all *batch* of a sensor in the training process for each of the four datasets. The details are shown in Fig. 9. We can see that the predicted and actual values are approximated as the *batch* gradually increases, which indicates that the prediction effect of our proposed method is gradually enhanced and can help us in the next step of anomaly detection.

4.5.5 Impact of Sliding Steps on Assessment Metrics

To evaluate the impact of sliding steps on the evaluation metrics, we tested our method using sliding steps of lengths 1, 5, 8 and 10. The final anomaly detection results for different sliding steps are shown in Table 7. It is observed that for the MSL dataset, there were no significant differences between the different slide step lengths with regards to the evaluation metrics. For NSL-KDD and Wafer datasets, the best performance was achieved at slide step length 8, while SWaT performed best at slide step length 10. Thus, varying the sliding step size had some influence on our method's performance.

4.5.6 Comparisons with Baseline Methods

The baseline methods include Linear Model, Proximity-based, Outlier Ensembles, Probabilistic, and Neural Networks. PCA cannot intervene in the process by parametric methods, and the principal elements derived in the case of non-Gaussian distribution may not be optimal. KNN algorithm has low prediction accuracy for rare categories when dealing with sample imbalance problems. HBOS algorithm is challenging to handle high-dimensional issues, and the algorithm requires a priori conditions that features are independent of each other. The IForest algorithm is not good at dealing with relatively local sparse points and is also unsuitable for high-dimensional data. Sampling causes the classifier to lose important information about the unsampled classes because the sample set is less than the original set of samples. ECOD algorithm is less effective in datasets with data bias. Beta-VAE, MO_GAAL, and DeepSVDD all reconstruct time series on a point-by-point basis without capturing spatial dimension dependencies, limiting the model's capability and detection performance. GDN does not capture the dependence of multivariate time-series data in the temporal dimension. At the same time, the graph structure information presented by this method cannot change dynamically with the input data, which limits the persuasiveness of the model to some

Table 13 Software Quality Metrics

Quality requirements	Quality characteristics	Quality sub-characteristics	Direct metrics	Metric description
Software will run on multiple platforms and operating systems currently in use by users	Portability	Hardware independency	Hardware dependency	None
		Software independency	Software dependency	Runs on Windows, CentOS, Ubuntu, etc.
		Ease of installation	The ease of installing software successfully	This software can be installed by simply starting the installation program through the command line. Configuring the language environment and installing related dependency packages will take approximately an hour.
		Compatibility	Environmental changes	The number of environment variables that must be modified after software installation is 0.
Software will be reliable	Reliability	Defectlessness	Test coverage	Statement coverage: 100% Branch condition coverage: 100% Path coverage: 100%
			Review coverage	All code in the software has been reviewed.
		Availability	Percentage of software available	99.9%
Software will be functionally robust	Functionality	Completeness	Test coverage	Branch condition coverage: 100%
		Reusability	Partial function modification	Modifying a single function point will not affect the regular operation of other function points.
Software will be easy to use	Usability	Comprehensibility	The understandability and clarity of the system prompt information	Software can help users accurately understand the current real status of the system and guide them for further operations. New users typically require one hour to learn the software's functionality.
		Ease of learning	Availability of online assistance	The software has an online help manual. New users typically require half an hour to learn the basic functionality of operating the software.
		Ease of operation	Ease of use of the software for users	The software interface is simple to operate and has shortcut keys, without too many complicated steps.

extent. The limitations of the above approaches are that they either do not consider feature-level correlations or do not explicitly address them. Our model not only captures dependencies in the temporal dimension but also deals with correlations in the spatial dimension in a coordinated manner. It is experimentally verified that the above joint treatment is effective in modeling dependencies on multivariate time series.

Table 8 shows the anomaly detection results of DCN and other baseline methods on datasets NSL-KDD, SWaT, MSL, and Wafer for the three evaluation metrics. In the MSL dataset, the F1 values of some baseline machine learning methods are close to our proposed method. As seen from Table 1, the MSL dataset has the highest percentage of anomalies in the four datasets, so performing anomaly detection on this dataset is close to executing a common binary classification problem. Some traditional machine learning methods can also show better results. On the remaining three datasets, our method outperforms the baseline model. Thus, our process improves the evaluation metrics for anomaly detection compared to the baseline machine learning approach. That is, the time consumed is necessary. Although GDN does not explicitly model the dependence of multivariate time-series data in the temporal dimension, it treats the sensor data in the temporal dimension as sensor features and projects these features to high-dimensional processing and captures the spatial dependence using GAT, which can also model the features of multivariate time-series data in the temporal dimension to some extent. In both NSL-KDD and SWaT datasets, DCN and GDN significantly improve over the traditional models because the traditional methods focus only on the dependence of multivariate time-series data in the temporal dimension. Also, GDN models the dependence of multivariate time-series data in the temporal dimension in an implicit way. Therefore, the improvement effect of DCN over GDN on these two datasets is not as significant as the traditional approach. However, DCN has a significant improvement over GDN on the Wafer dataset. This is because Wafer is more unbalanced and has higher

dimensionality than NSL-KDD, SWaT and MSL, as shown in Table 1. Therefore, DCN can achieve accurate anomalous event detection and localization even in highly unbalanced and high-dimensional attack scenarios while demonstrating the effectiveness of capturing spatio-temporal features of multivariate time-series data in an explicit manner. Finally, DCN can also better balance the two remaining evaluation metrics while maintaining higher F1 values.

We also evaluated our model in the federated scenario by dividing the dataset multiple times for experimental validation. The results show that our model's performance improves on NSL-KDD and SWaT datasets. However, on MSL and Wafer datasets, the three evaluation metrics of FDVL-DCN are generally lower than in the non-federated scenarios, the reasons are:

1. There are more or fewer differences between the manually divided dataset and the actual case.
2. Uneven distribution of anomalies in MSL and Wafer datasets.
3. MSL and Wafer have less anomalous data than NSL-KDD and SWaT, and it is difficult for the trained model to learn the information of anomalies.

Overall, our proposed method exhibits higher performance compared to the baseline models.

4.5.7 Loss Curves of Different Models

In order to compare the trend of LOSS curves generated by different methods in a clear and explicit manner, we use Eq. 24 to normalize the LOSS values of all methods, mapping them to the same interval,

$$Y = a + \frac{b-a}{X_{max} - X_{min}}(X - X_{min}) \tag{24}$$

where X is the list of loss function values before mapping, Y is the list of loss functions after mapping. X_{max} is the maximum value in this set of data X, X_{min} is the minimum value in this set of data X. As shown in Fig. 10, our proposed method exhibits a faster convergence rate than other baseline methods while maintaining a stable convergence state after reaching convergence. This normalization approach enables us to effectively compare the performance trends of different models.

4.5.8 Comparison of LOSS Curves in Federated and Non-Federated Scenarios

To compare with the LOSS curves in the non-federated scenario, we selected the model after the first round of fusion in the federated scenario and tested it on the training set to obtain the corresponding LOSS curves. The comparison results are shown in Fig. 11. After fusing model parameters from different nodes, it can be seen that the new model can predict the behavior of future moments with minor LOSS, improving the model's performance.

4.5.9 Time Consumption Evaluations

We evaluated the time consumption of Federated Dual-View Learning(FDVL) and Federated Positive-View Learning(FPVL), the detailed parameters of the training are shown in Table 9, and the comparison results are shown in Table 10. It can be seen that compared to FPVL, the training time increased by 3.9% on average. Federated Negative-View Learning(FNVL) plays only an auxiliary role in FDVL, and the number of negative samples is much smaller than the number of positive samples in the real scenario, so the quality of the negative-view model is also much lower than the quality of the positive-view model. All we need is the range of the anomaly interval corresponding to the negative-view model. So it is undesirable to train an FNVL alone, and it is not meaningful to compare the time consumption of FNVL and FDVL alone. In contrast, compared to the FPVL, FDVL increases the time on the four datasets by 4%, 2.5%, 7.5%, and 1.6%, respectively. The F1 values increases by 7.4%, 23.8%, 17.2%, and 61.8% respectively, as shown in Table 11 (Calculation of the above time growth rate percentage and F1 value growth rate percentage: (FDVL − FPVL) / FPVL). In general, the increased time for training is within a manageable range, but it helps us to detect anomalies more effectively.

4.5.10 Quality Metrics FDVL-DCN Big Data as Service

The software service corresponding to non-federated scenarios (DCN) and federated scenarios (FDVL-DCN) significantly improves anomaly detection evaluation metrics compared to traditional methods. Table 12 shows the specific percentage improvement in F1 values. Additionally, the software service was evaluated regarding functionality, reliability, ease of usage, efficiency, maintainability, and portability, as detailed in Table 13. From these two tables, it can be verified that the software service corresponding to FDVL-DCN improves the model's performance and can help make more accurate decisions.

5 Conclusions and Future Work

This paper proposes a dynamic circular network architecture and a general modeling approach for multivariate time-series data spatial structures while combining both with federated dual-view learning for anomaly detection

and finally encapsulating the main components in FDVL-DCN as a software service to improve usability. Comparative experiments of ten machine learning models and multi-angle model parameters show that FDVL-DCN has better anomaly detection performance while protecting the privacy of each participant. The multifaceted software quality metrics show that the software service corresponding to FDVL-DCN possess some robustness. In the future, we can improve the federated fusion algorithm, the aggregation method of spatial weights in BI-GCN, and the anomaly detection algorithm to obtain further improvements.

References

Baumann A, Haupt J, Gebert F, Lessmann S (2019) The price of privacy. Bus Inf Syst Eng 61(4):413–431

Belhadi A, Djenouri Y, Srivastava G, Cano A, Lin JCW (2021) Hybrid group anomaly detection for sequence data: application to trajectory data analytics. IEEE Trans Intell Transp Syst 23(7):9346–9357

Burgess CP, Higgins I, Pal A, Matthey L, Watters N, Desjardins G, Lerchner A (2018) Understanding disentangling in β-vae. arXiv:1804.03599

Cao D, Wang Y, Duan J, Zhang C, Zhu X, Huang C, Tong Y, Xu B, Bai J, Tong J, Zhang Q (2020) Spectral temporal graph neural network for multivariate time-series forecasting. In: Neurips 2020, December 6-12, virtual

Chen W, Chen L, Xie Y, Cao W, Gao Y, Feng X (2020) Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In: Proceedings of the AAAI conference on artificial intelligence 34:3529–3536

Deng A, Hooi B (2021) Graph neural network-based anomaly detection in multivariate time series. In: Proceedings of the AAAI conference on artificial intelligence 35:4027–4035

Djenouri Y, Djenouri D, Belhadi A, Srivastava G, Lin JCW (2021) Emergent deep learning for anomaly detection in internet of everything. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2021.3134932

Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. In: Precup D, Teh YW (eds) ICML 2017, Sydney, NSW, Australia, 6-11 August, PMLR, vol 70, pp 1263–1272

Goldstein M, Dengel A (2012) Histogram-based outlier score (hbos): a fast unsupervised anomaly detection algorithm. KI-2012: poster and demo track 9

Hautamäki V, Kärkkäinen I, Fränti P (2004) Outlier detection using k-nearest neighbour graph. In: ICPR 2004, Cambridge, UK, August 23-26, IEEE Computer Society, pp 430–433

Hu Y, Xia W, Xiao J, Wu C (2020) GFL: a decentralized federated learning framework based on blockchain. arXiv:2010.10996

Jiang J, Chen J, Gu T, Choo KR, Liu C, Yu M, Huang W, Mohapatra P (2019) Anomaly detection with graph convolutional networks for insider threat and fraud detection. In: MILCOM 2019, Norfolk, VA, USA, November 12-14, IEEE, pp 109–114

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: ICLR 2017, Toulon, France, April 24-26, Conference Track Proceedings, OpenReview.net

Klicpera J, Bojchevski A, Günnemann S (2019) Predict then propagate: graph neural networks meet personalized pagerank. In: ICLR 2019, New Orleans, LA, USA, May 6-9, OpenReview.net

Langfu C, Zhang Q, Yan S, Liman Y, Yixuan W, Junle W, Chenggang B (2023) A method for satellite time series anomaly detection based on fast-DTW and improved-KNN. Chin J Aeronaut 36(2):149–159

Li Z, Zhao Y, Hu X, Botta N, Ionescu C, Chen GH (2022) ECOD: unsupervised outlier detection using empirical cumulative distribution functions. arXiv:2201.00382

Liang Y, Guo Y, Gong Y, Luo C, Zhan J, Huang Y (2020) Flbench: a benchmark suite for federated learning. Benchcouncil international federated intelligent computing and block chain conferences. Springer, Heidelberg, pp 166–176

Lim N, Hooi B, Ng SK, Wang X, Goh YL, Weng R, Varadarajan J (2020) Stp-udgat: Spatial-temporal-preference user dimensional graph attention network for next poi recommendation. In: Proceedings of the 29th ACM international conference on information & knowledge management, pp 845–854

Liu FT, Ting KM, Zhou Z (2008) Isolation forest. In: (ICDM 2008), December 15-19, 2008, Pisa, Italy, IEEE Computer Society, pp 413–422

Liu Y, Li Z, Zhou C, Jiang Y, Sun J, Wang M, He X (2020) Generative adversarial active learning for unsupervised outlier detection. IEEE Trans Knowl Data Eng 32(8):1517–1528

McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: AISTATS 2017, 20-22 April, PMLR, proceedings of machine learning research, vol 54, pp 1273–1282

Mehdiyev N, Evermann J, Fettke P (2020) A novel business process prediction model using a deep learning method. Bus Inf Syst Eng 62(2):143–157

Oberdorf F, Schaschek M, Weinzierl S, Stein N, Matzner M, Flath CM (2022) Predictive end-to-end enterprise process network monitoring. Bus Inf Syst Eng pp 1–16

Ruff L, Görnitz N, Deecke L, Siddiqui SA, Vandermeulen RA, Binder A, Müller E, Kloft M (2018) Deep one-class classification. In: ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, PMLR, Proceedings of Machine Learning Research, vol 80, pp 4390–4399

Schlichtkrull MS, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M (2018) Modeling relational data with graph convolutional networks. In: Gangemi A, Navigli R, Vidal M, Hitzler P, Troncy R, Hollink L, Tordai A, Alam M (eds) ESWC 2018, Heraklion, June 3-7, Proceedings, Springer, LNCS, vol 10843, pp 593–607

Shi L, Zhang Y, Cheng J, Lu H (2019) Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In: CVPR 2019, Long Beach, CA, USA, June 16-20, Computer Vision Foundation / IEEE, pp 12,026–12,035

Shyu ML, Chen SC, Sarinnapakorn K, Chang L (2003) A novel anomaly detection scheme based on principal component classifier. Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering, Tech. rep

Sugiyama M, Borgwardt KM (2013) Rapid distance-based outlier detection via sampling. In: Advances in neural information processing systems 26: 27th annual conference on neural information processing systems 2013. proceedings of a meeting held December 5-8, Lake Tahoe, Nevada, US, pp 467–475

Velickovic P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: ICLR 2018, Vancouver, BC, Canada, April 30-may 3, Conference Track Proceedings, OpenReview.net

Wang J, Ma Y, Huang Z, Xue R, Zhao R (2019) Performance analysis and enhancement of deep convolutional neural network. Bus Inf Syst Eng 61(3):311–326

Wu D, Jiang Z, Xie X, Wei X, Yu W, Li R (2020) LSTM learning with Bayesian and gaussian processing for anomaly detection in industrial IoT. IEEE Trans Ind Inf 16(8):5244–5253

Wu Z, Pan S, Long G, Jiang J, Chang X, Zhang C (2020b) Connecting the dots: Multivariate time series forecasting with graph neural networks. In: Gupta R, Liu Y, Tang J, Prakash BA (eds) KDD '20: The 26th ACM SIGKDD conference on knowledge discovery and data mining, August 23-27, ACM, pp 753–763

Yan S, Xiong Y, Lin D (2018) Spatial temporal graph convolutional networks for skeleton-based action recognition. In: (AAAI-18), (IAAI-18), (EAAI-18), New Orleans, Louisiana, USA, February 2-7, AAAI Press, pp 7444–7452

Yu B, Yin H, Zhu Z (2018) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: IJCAI 2018, July 13-19, Stockholm, Sweden, ijcai.org, pp 3634–3640

Zhang W, Zhou T, Lu Q, Wang X, Zhu C, Sun H, Wang Z, Lo SK, Wang F (2021) Dynamic-fusion-based federated learning for COVID-19 detection. IEEE Internet Things J 8(21):15,884-15,891

Zhang W, Chen X, He K, Chen L, Xu L, Wang X, Yang S (2022) Semi-asynchronous personalized federated learning for short-term photovoltaic power forecasting. Digit Commun Netw. https://doi.org/10.1016/j.dcan.2022.03.022

Zhao L, Song Y, Zhang C, Liu Y, Wang P, Lin T, Deng M, Li H (2020) T-GCN: atemporal graph convolutional network for traffic prediction. IEEE Trans Intell Transp Syst 21(9):3848–3858

Zhong J, Li N, Kong W, Liu S, Li TH, Li G (2019) Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In: CVPR 2019, long beach, CA, USA, June 16-20, 2019, Computer Vision Foundation/IEEE, pp 1237–1246

Zhou X, Liang W, She J, Yan Z, Kevin I, Wang K (2021) Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles. IEEE Trans Veh Technol 70(6):5308–5317

Zhou X, Hu Y, Wu J, Liang W, Ma J, Jin Q (2022) Distribution bias aware collaborative generative adversarial network for imbalanced deep learning in industrial IoT. IEEE Trans Ind Inform 19(1):570–580

Zhou X, Liang W, Ma J, Yan Z, Kevin I, Wang K (2022) 2d federated learning for personalized human activity recognition in cyber-physical-social systems. IEEE Trans Netw Sci Eng. https://doi.org/10.1109/TNSE.2022.3144699