Computer Modeling in Engineering & Sciences
DOI:10.32604/cmes.2020.09404
Article

Tech Science Press

FP-STE: A Novel Node Failure Prediction Method Based on Spatio-Temporal Feature Extraction in Data Centers

Yang Yang[1,*], Jing Dong[1], Chao Fang[2], Ping Xie[3] and Na An[3]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
[2]Beijing Smartchip Microelectronics Technology Company Limited, Beijing, China
[3]The 54th Research Institute of CETC, Shijiazhuang, China
*Corresponding Author: Yang Yang. Email: yyang@bupt.edu.cn
Received: 10 December 2019; Accepted: 17 March 2020

Abstract: The development of cloud computing and virtualization technology has brought great challenges to the reliability of data center services. Data centers typically contain a large number of compute and storage nodes which may fail and affect the quality of service. Failure prediction is an important means of ensuring service availability. Predicting node failure in cloud-based data centers is challenging because the failure symptoms reflected have complex characteristics, and the distribution imbalance between the failure sample and the normal sample is widespread, resulting in inaccurate failure prediction. Targeting these challenges, this paper proposes a novel failure prediction method FP-STE (Failure Prediction based on Spatio-temporal Feature Extraction). Firstly, an improved recurrent neural network HW-GRU (Improved GRU based on HighWay network) and a convolutional neural network CNN are used to extract the temporal features and spatial features of multivariate data respectively to increase the discrimination of different types of failure symptoms which improves the accuracy of prediction. Then the intermediate results of the two models are added as features into SCS-XGBoost to predict the possibility and the precise type of node failure in the future. SCS-XGBoost is an ensemble learning model that is improved by the integrated strategy of oversampling and cost-sensitive learning. Experimental results based on real data sets confirm the effectiveness and superiority of FP-STE.

Keywords: Failure prediction; data center; features extraction; XGBoost; service availability

1 Introduction

With the introduction of virtualization technology, the phenomenon of large-scale service failures due to node failures frequently occurs in cloud data centers. According to incomplete statistics, the Google application engine has at least one downtime every quarter. Amazon has experienced two large-scale downtimes in 2011 and 2017, a large number of applications are affected by the disruption. The average time between the failures of IBM's ASCIW Hite is only about 40 hours. Google analyzed the running data within one year from dozens of sites deployed in different regions with nodes ranging from 1000 to

8000, indicated that the node failure rate is 2–3%, which means that one node will fail every 36 hours [1]. Research shows that node failure is one of the main causes of service downtime [2].

The current research on reliability assurance mainly focuses on fault detection and diagnosis which are both remedial measures taken after the failure. As a proactive reliability management and failure prevention mechanism, failure prediction technology can predict the failure tendency of nodes before the actual failure occurs, helps the system avoid costly service losses and additional cost of repairs. The result of prediction can also provide an important reference for resource mapping, virtual machine migration and fault isolation of cloud data center, ensuring the continuity and quality of service [3]. Therefore, this paper mainly studies the problem of node failure prediction.

At present, most existing node failure prediction methods belong to monitoring based failure prediction approaches [4]. The basic principle is to build a failure prediction model by applying machine learning techniques according to the characteristics of historical failure data (including node performance monitoring indicators and error logs), and then use the model to predict the likelihood of a node failing in the coming days [2]. However, the existing failure prediction model based on the classical machine learning algorithm can only realize binary prediction on whether a node fails, but cannot achieve accurate multiple prediction of failure types. After analysis, this article summarizes the challenges of building an accurate prediction model for node failure in data centers into three aspects:

Complicated Failure Causes: Due to the complexity of the large-scale data center, node failures could be caused by many different software or hardware issues, such as software bugs, OS crash, disk failure, service exception, etc. However, in the current researches, basic classification algorithms in machine learning are mostly used to construct failure prediction models that are more suitable for solving binary failure prediction problems of a single component such as disk, memory and CPU [5,6]. Therefore, it is difficult for these models to make an effective division of multiple failure types.

Complex Failure Symptoms and Rough Feature Extraction: The symptoms of failures are complex, but the feature extraction process of the existing methods is too rough and simple without mining the deep characteristics of failure symptoms. The node performance monitoring indicators we detected (such as memory usage, CPU load, disk read and write rates, network packet loss, etc.) belong to time series data [7]. Most of the failure prediction methods are not designed specifically for time-series data [8], so the temporal characteristics behind are ignored. What's more, the node of cloud data centers often has multiple copies to guarantee service reliability. The features of neighboring nodes in the cluster, especially in the same load balancing group often have a certain correlation [9]. The traditional prediction method also lacks an analysis of the spatial related characteristics. Regretfully, these ignored spatio-temporal correlation information is very helpful to improve the accuracy of prediction results.

Highly Imbalanced Data: Finally, the imbalance of samples can also affect the accuracy of the prediction results. In reality, the probability of failure is much lower than normal [10]. So the positive and negative samples in the training data set are unbalanced, which is a big challenge for the classification algorithm, because the classifier will favor the majority class, resulting in the high prediction accuracy of the majority class(normal class), and the low prediction accuracy of the minority class (failure class).

This paper aims to explore new approaches to solve the above problems. Specifically, the innovations and contributions of this article are summarized as follows:

- A novel node failure prediction method is proposed for data centers named FP-STE (Failure Prediction based on Spatio-temporal Feature Extraction), which realizes accurate prediction of multiple types of node failure for the first time.

- Considering the complexity and implicitness of failure symptoms, an improved GRU network HW-GRU and a CNN model are firstly used to extract the temporal and spatial characteristics of node features respectively which improves the accuracy of failure prediction.
- Aiming at the influence of sample imbalance on the prediction effect, a new ensemble learning model, SCS-XGBoost is proposed to realize accurate multi-class failure prediction which is improved by the integrated strategy of SMOTE sampling and cost-sensitive learning.

The rest of this article is organized as follows. Section 2 reviews some related works. Section 3 describes our node failure prediction method FP-STE. Simulation results and corresponding discussions are presented in Section 4. Finally, Section 5 summarizes this paper.

2 Related Work

Failure prediction using machine learning techniques has gained enormous attention in recent times, and a lot of research has been conducted in this area. According to the literature [3,4,9], failure prediction method can be classified into two categories: failure tracking based failure prediction and monitoring-based failure prediction.

For failure tracking based prediction approaches, the basic idea is to derive the spatiotemporal correlation rules of failure from previous failures that have occurred, so as to infer the upcoming failures. Failure prediction decision can be made by either estimating the probability distribution of a random variable for the time to the next failure [11–14] or building on the co-occurrence of failures [9,15,16]. The author in [11] developed a machine learning approach for predicting individual component time until failure which they reported as far more accurate than the traditional MTBF approach. But the drawback of their work was that their model has not been trained on a module with real-time failure. Hence, there is no assurance that this model will predict failure accurately. Mohammed et al. [12] proposed a failure prediction model based on ARIMA (1,1,1) time series and machine learning. The primary algorithms they considered are the support vector machine (SVM), random forest (RF), *k*-nearest neighbors (KNN), classification and regression trees (CART) and linear discriminant analysis (LDA). In recent years, deep learning methods have also been used to predict failure time series. Xu et al. [13] introduced a method based on Recursive Neural Network (RNN) to assess the health statuses of hard drives. Zhang et al. [14] presented a virtual network failure diagnosis method based on LSTM, which has early failure prediction capability. Fu et al. [15] and Yu et al. [16] mine the causal association between log events and generate event correlation graphs to represent event rules and predict failure events.

The biggest drawback of failure tracking based prediction approaches is that they can only provide the time when the failure occurs and cannot predict the failure cause type, so it is not suitable for the failure prediction of data center node with multiple failure causes.

In monitoring-based failure prediction approaches, failures are considered as deviations from normal behaviors and can be predicted via such techniques as function approximations, pattern recognition, and classifiers with the assumption that failure-prone behaviors can be identified by characteristic patterns of symptoms. For example, memory leaks can be caught by their symptoms such as abnormal memory usage, CPU load, disk I/O, or unusual function calls in the system. Li et al. [5] collected various performance data of the Apache Web server and established an autoregressive system model to predict the exhaustion of resources. On this basis, Hoffmann et al. [6] proposed the UBF prediction model which has higher accuracy in predicting the “remaining free memory”, but for the “Web server response time”, the method based on SVM is more efficient. Liang et al. [17] analyzed the log records of IBMs BlueGene/L and proposed a failure prediction method based on a custom nearest neighbor classifier that performs better than the other standard classification algorithms such as RIPPER and SVM. The IBM Research Institute published research results at KDD 2016 using RGF algorithms and migration learning

to predict hard disk failures [18]. Khan et al. [19] used an ensemble classifier to achieve hard drive failure prediction on a cloud infrastructure. Liu et al. [20] presented a switch failure prediction method in data center networks based on Random Forest. Sun et al. [21] proposed a deep-learning-based prediction scheme for system-level hardware failure prediction and design a loss function to train the model with extremely imbalanced samples effectively. Experimental results show the effectiveness of the model in predicting disk and memory failures.

There are two main disadvantages: On the one hand, these researches only considered single component failure prediction for example hard disk failure, while node failure can be triggered by any software or hardware issue, or a mixture of both, which requires analyzing more heterogeneous characteristics and brings greater challenges for feature extraction and classifier. On the other hand, most of the works above convert failure prediction into a classification problem. But the feature extraction parts of these methods are too simple, only considering the statistical characteristics of parameters. In addition, the single classifiers they used have limitations, especially when dealing with unbalanced samples and multiple classification problems. So, the existing monitoring-based failure prediction approaches cannot achieve accurate multi-class failure prediction of nodes.

To solve the above problems, this paper proposes a new failure prediction method using HW-GRU, CNN, and an improved ensemble learning model SCS-XGBoost to realize accurate multi-class failure prediction.

3 FP-STE: The Proposed Failure Prediction Method

3.1 Overview

According to the theories of monitoring-based failure prediction described in Section 2, failure prediction technology is to detect the wrong side effects (i.e., symptoms) to determine whether the system is about to fail, and then use the model to predict the time and type of failure. In other words, we need to build a model between symptoms and failures, use the model to simulate the internal functions of the system, calculate the system running trend to determine whether the system is about to generate failure.

This paper proposes a new node failure prediction method called FP-STE (Failure Prediction based on Spatio-temporal Feature Extraction). The basic principle of FP-STE is shown in Fig. 1 and described as follows.

Assuming that the current system time is t, the FP-STE is trained by using the historical monitoring data of the system during the T period before the time t, so that the FP-STE learns the characteristic patterns of symptoms and has the capability of failure prediction. Then, in the failure prediction stage, the operating parameters in the current observation window Δt_d are extracted and input into the three trained models to predict the status of the target node at the next moment. FP-STE mainly includes three important stages: data preparation, feature extraction and failure prediction.

Data Preparation: we collect all symptom information of the nodes from the monitor and network management system to constitute the training set and the test set. One sample in the training set is expressed as $\{X_{1\times M}, Y\}$, $\mathrm{X}_{1\times M}$ is the feature vector of the node including key performance indicators and alarm information, and M represents the number of features. Y represents node status label including the normal status class and a variety of predefined failure classes.

Feature Extraction: feature extraction is to get high-level abstract characteristics of complex failure symptoms, which can improve the accuracy of prediction. Deep learning technology can ensure the most effective information extraction and feature expression because of its multi-layer structure. FP-STE builds two independent learners HW-GRU and CNN to capture the temporal and spatial features respectively.

Failure Prediction: the intermediate results of CNN and HW-GRU are input into the improved ensemble learning model SCS-XGBoost which gives the final prediction results (normal or the type of

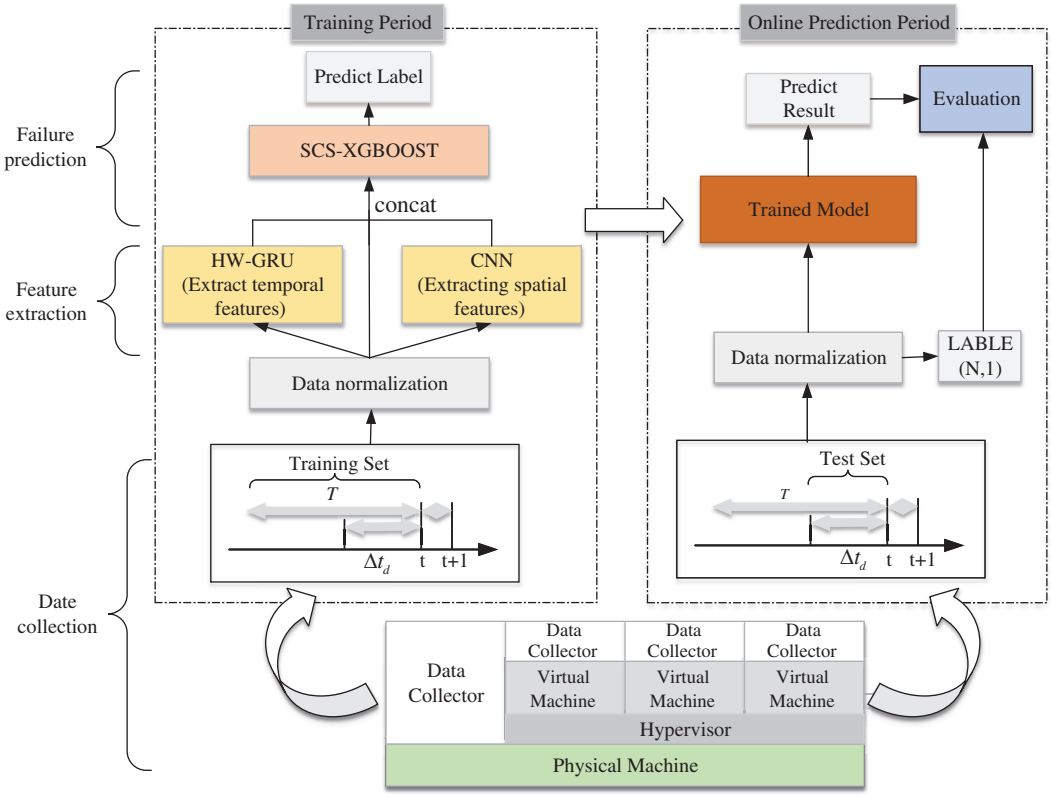


Figure 1: The overview of FP-STE

failure). SCS-XGBoost is an ensemble learning model that is improved by an integrated strategy of oversampling and cost-sensitive learning, which can improve the predictive effect on imbalance samples.

In the next sections, we will introduce the three important models in FPSTE: HW-GRU, CNN, SCS-XGBoost.

3.2 HW-GRU: An Improved Recurrent Neural Network Based on HighWay Network

Through long-term monitoring and analysis of the node's performance indicators (such as CPU usage, memory usage, I/O rate), it can be found that the performance indicators of normal running nodes show regular and stable changes. However, for nodes with a risk of failure, the performance indicators will show sudden or gradual abnormal fluctuations, which is a symptom of node failure. Therefore, node indicators have a certain correlation in the time dimension. In addition, for different types of failure, the nodes often experience a state evolution for a period of time when they reach the failure state, and the running performance parameters will show different fluctuation laws in the time dimension. We think that extracting the temporal characteristics of the analysis node's performance indicators will increase the discrimination of node states and make the prediction process more accurate.

GRU (Gated Recurrent Unit) is a time recursive neural network model that can memorize the influence of historical input information on subsequent output results and the specific information at specific points in time, which is very suitable for processing time-series data and extracting related information of adjacent features in time dimension [22]. As is shown in Fig. 2, the high-capacity deep network obtained by connecting GRU neurons layer by layer can be used to process complex sequence data, further improving the prediction accuracy. Therefore, considering the complexity of failure symptoms, this paper uses

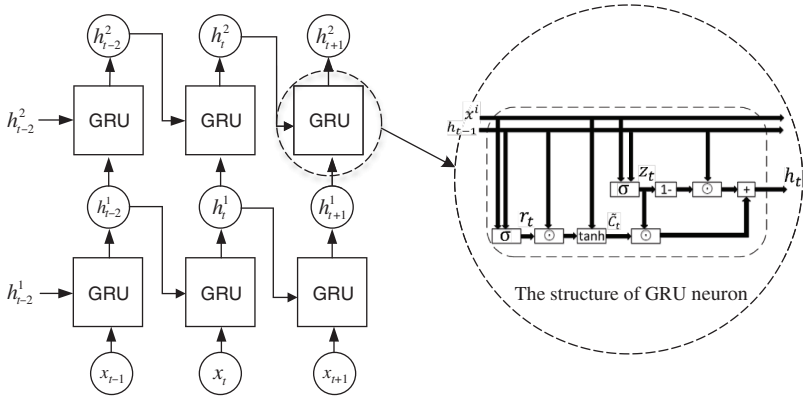


Figure 2: Double-layer GRU neural network model

multi-layer GRU deep networks to extract the temporal characteristics of performance indicators in the time window Δt_d before the failure occurs.

However, there are also some drawbacks to multi-layer GRU deep networks. First of all, the high level of abstraction will lose some important features that are important to distinguish the operating status and failure types of nodes, which is contrary to our original goal of obtaining more feature details. Secondly, as the GRU network deepens, the training efficiency of deep neural networks will decrease, which is not suitable for online fault prediction. Finally, the tanh activation function is prone to produce vanishing gradient problems, which leads to hovering at one point and unable to search for the optimal solution.

Aiming at the first two problems, this paper uses the Highway mechanism [23] to improve the multilayer GRU network and proposes a new network structure named HW-GRU. It can achieve cross-layer information transfer, that is, the output information at a certain time can be directly transmitted to the next layer without going through the neural network of the current hidden layer, so that the temporal features extracted from the shallow layer can be retained to improve the accuracy of failure prediction, and the convergence speed of model training can be increased. At the same time, this article also uses the Relu activation function to adjust the GRU neuron and set a small learning rate to prevent the GRU network from entering dead neurons. The structure of the HW-GRU neuron is shown in Fig. 3.

$$z_t = \sigma\ (W_{hz} * h_{t-1}^i + W_{xz} * x_t^i + b_z) \tag{1}$$

$$r_t = \sigma(W_{hr} * h_{t-1}^i + W_{xr} * x_t^i + b_r) \tag{2}$$

$$\widetilde{c_t^i} = \text{relu}(W_{hc} * (r_t \odot h_{t-1}^i) + W_{xc} * x_t^i + b_c) \tag{3}$$

$$h_t^i = (1 - z_t) * h_{t-1}^i + z_t * \widetilde{c_t^i} \tag{4}$$

$$d_t = \sigma(W_{hd} * h_{t-1}^i + W_{xd} * x_t^i + b_d) \tag{5}$$

$$h_t^{i+1} = (1 - d_t) * h_t^i + d_t * x_t^i \tag{6}$$

z_t is the update gate, r_t is the reset gate, $\widetilde{c_t^i}$ is the memory gate, d_t is cross-layer selection gate which determines how much of the output and input information of the previous layer is reserved to the next layer. x_t^i is the input vector at time t in the i^{th} layer, $\mathrm{h}_{\mathrm{t}-1}^{\mathrm{i}}$ h_{t-1}^i is the output information at time $t-1$ in the i^{th} layer, W is the weights, b is the offset, σ is sigmoid activation function, Relu is also activation function, $\odot$ is the Hadamard product, h_t^i is the output of at time step t in the i^{th} layer, h_t^{i+1} is the output at time t in the $(i+1)^{th}$ layer.

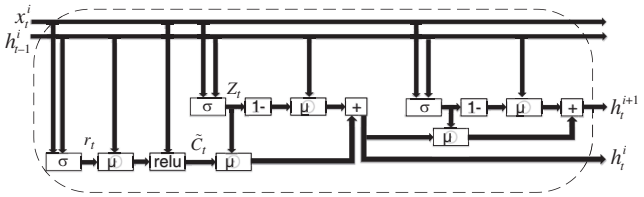


Figure 3: The structure of HW-GRU neuron

The multi-layer HW-GRU model designed for temporal feature extraction in this paper includes an input layer, three HW-GRU hidden layers, a full connection layer, and a Softmax layer. The improved multi-layer HW-GRU network can not only speed up the training speed and prevent the model from falling into the local optimum but also integrate the time features at different levels to obtain the most comprehensive feature extraction effect, which meets the requirements of failure characteristics extraction. The model architecture as well as the model parameter is shown in Fig. 4 and described as follows.

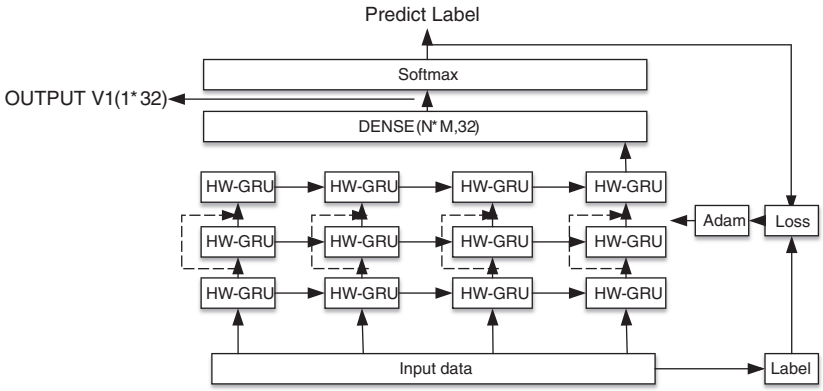


Figure 4: The structure of the multi-layer HW-GRU model

1) Input layer: The performance index and node status label of the target node are collected by the node monitor, and incomplete data are removed to obtain the original data. Using the sliding time window to recombine the original data to generate the input and output data of the model. Let the input of HW-GRU model be $X_{t_d} = [x_{t_1}, x_{t_2}, \cdots, x_{t_d}]_{\Delta t_d \times M}$, representing all state information of the node in the Δt_d period before time t_d. The output is of the model is Y_{t_d}, which represents the node state type at time t_d. M is the dimension of node features.

2) HW-GRU hidden layers: The HW-GRU hidden layers are used to extract and abstract the temporal feature of the input data layer by layer. Each hidden layer contains Δt_d HW-GRU neurons connected by the front and back moments, and the input of each neuron corresponds to the feature sequence of a moment. The output of the i^{th} hidden layer can be expressed as $H^i = [h^i_{t_1}, h^i_{t_2}, \cdots, h^i_{t_d}]$, then the output at the next layer (the $(i+1)^{th}$ layer) at the time t can be calculated by Eqs. (1)–(6).

3) Output layers: The output layer of the network consists of a fully connected neural network layer (also known as a dense layer) and a Softmax network layer. The fully connected layer is used for vector dimension transpose. Through the dense layer composed of 32 fully connected neurons, the two-dimensional vector output from the hidden layer is converted into a one-dimensional vector of 1×32. Softmax layer is a classifier that outputs the probability of different types of failure events at the next time. We take the output vector $V_1(1 \times 32)$ of last moment neurons in the third HW-GRU layer that remembers the characteristic information of all previous moments as the extracted temporal feature.

3.3 CNN: Convolutional Neural Network

In the spatial dimension, the research of the literature [9,15,16] shows that the state of the node is affected by the state of the adjacent node and shows a certain correlation. That is to say, when the performance parameter of a node is significantly different from other nodes, the node is likely to fail. In addition, when the value of a performance indicator of a node is abnormal, other indicators of the node will fluctuate, and different failure types often have significant effects on different performance indicators. For example, the packet loss rate will change greatly when porting IP address conflict, while the disk read and write rate will change significantly when the disk fails. The specificity, mutation and correlation of the performance parameters of these nodes in the spatial dimension may become the key to early symptom recognition of node failure. Therefore, we need a model to capture the correlation between different nodes' parameters, and take the extracted correlation features as the additional features of the classifier, so as to improve the accuracy of failure recognition.

CNN is a feedforward neural network, which extracts hidden local correlation features by layer-by-layer convolution and pooling of input data, and generates high-level features by layer-by-layer combination and abstraction [24]. As we all know, CNN has made great success in the field of image classification, it can effectively extract pixel information of two-dimensional images. Spatial feature extraction of failures requires the integration of attribute features of multiple nodes. The original data composed of feature vectors of multiple nodes also belongs to two-dimensional structured data, so the process of spatial feature extraction and abstraction we expected to be realized in this paper is very similar to the process of pixel information extraction of images. Therefore, this paper tries to use the CNN model for spatial correlate feature extraction of network node failure for the first time.

The CNN model designed for spatial feature extraction in this paper consists of an input layer, a hidden layer, and an output layer. The hidden layer comprises two convolution layers, one pooling layer, and three sets of fully connected dense layers. The output layer is composed of a layer of SoftMax. The model architecture as well as the model parameter is shown in Fig. 5. and described as follows.

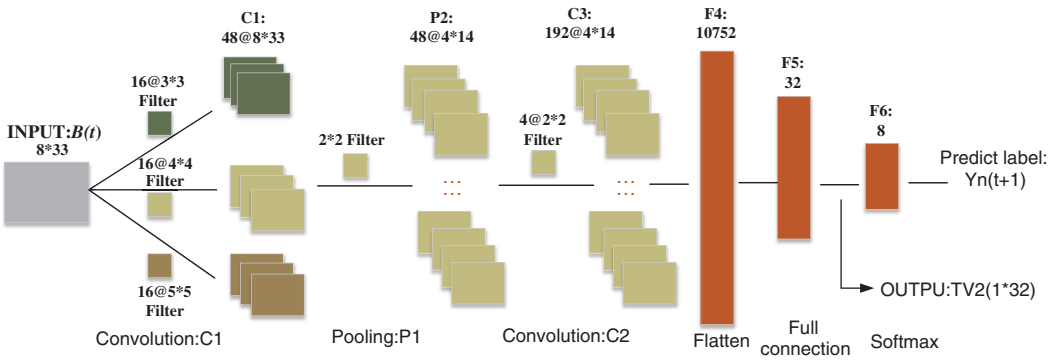


Figure 5: The structure of CNN model

1) Input tensor transformation: Firstly, we need to convert the feature vectors of multiple nodes associated with the target node into a two-dimensional feature tensor. Let the feature vector of node n at time t be $X_n(t) = [X_n^1(t), X_n^2(t), \cdots, X_n^M(t)]$. In order to comprehensively consider the correlation between nodes and features, this paper constructs spatial information of nodes at time t into spatial feature graph B:

$$B(t) = \begin{bmatrix} X_1(t) \\ X_2(t) \\ \vdots \\ X_{N+1}(t) \end{bmatrix} = \begin{bmatrix} X_1^1(t) & X_1^2(t) & \cdots & X_1^M(t) \\ X_2^1(t) & X_2^2(t) & \cdots & X_2^M(t) \\ \vdots & \vdots & & \vdots \\ X_{N+1}^1(t) & X_{N+1}^2(t) & \cdots & X_{N+1}^M(t) \end{bmatrix} \tag{7}$$

in which, M is the dimension of node performance parameters. N is the number of the closest nodes to the node n (especially in the same load balancing group). In Section 4, M is 33 and N is 7. The output of CNN is $Y_n(t+1)$ which is the label of node n at time $t+1$.

2) Convolution layer: The model contains two convolutional layers C1 and C2. The function of the convolutional layer is to use filters (convolution kernel) to slide in the input data and perform convolution weighting operation to complete feature extraction. Different scale filters can extract different local features. Large-scale filter may find the symmetric property, while small-scale filter could find particular characteristics. In the convolution layer C1, we use 48 filters of different scales to extract different local features of node attributes, including 3×3, 4×4 and 5×5. The number of each kind of filter is set to 16. In the convolution layer C2, we use four 2×2 filters to take the insightful information between nodes. In order to keep the dimension of feature maps extracted by different sizes of filters consistent, the padding method is "same". Moreover, we apply the "ReLu" activation function right after each convolutional layer.

3) Pooling layer: Pool layer is the lower sampling layer in CNN, which is used to reduce the dimension, shorten the training time and control over fitting. The most common pool types are Max pooling and mean pooling. This paper uses maximum pooling to retain more significant information. The size of convolution kernel of pooling layer P1 is set to 2×2.

4) Dense layers: The main function of the dense layer is to extract the distinguishing features by combining and sampling the features extracted from the convolutional layer, and finally achieve the purpose of classification. The CNN model designed in this paper has three dense layers: The first dense layer (flatten) is to transpose the feature maps obtained by convolution and merge them into a one-dimensional vector. The second dense layer is to further extract high-level features to obtain a 1×32 one-dimensional vector. The activation function of the first two layers is ReLu. The activation function of the last dense layer is softmax, which is used for failure classification and to transfer the prediction result into probabilities. Therefore, the output of the CNN is the probability of a certain type of failure.

5) Feature extraction: We extracted the output vector $V_2(1 \times 32)$ of the second dense layer as the abstract spatial features which will be used as the additional input of classifier.

3.4 SCS-XGBoost: An Ensemble Learning Model Improved by SMOTE Sampling and Cost-Sensitive Learning

In this paper, node failure prediction is equivalent to a multi-classification problem. In order to overcome the limitation of a single classifier in multi-type failure prediction, we choose the ensemble learning model XGBoost [25] has the classifier for failure prediction. Considering the influence of sample imbalance on the prediction results, this paper optimizes XGBoost by the integration strategy of SMOTE sampling and cost-sensitive learning [26]. The new model is renamed to SCS-XGBoost that can better adapt to the imbalance of samples and improve the precision of prediction.

1) The Data Level: SMOTE Oversampling

At the data level, we use the SMOTE algorithm [10] to oversample the minority classes. Specifically, for each sample x in minority training set $S_j(1 \le j \le K)$, the k-nearest neighbor algorithm is adopted to select the neighborhood sample set S_k, and then randomly select a neighbor sk to generate a new sample according to Eq. (8). Repeat the above operation a_j times for each sample in S_j, we can get the oversampling dataset

$S'_j = n_j\, a_j$. Where n_j is the sample number of the minority class j, a_j is the sampling rate of the class j, which is determined by the ratio of the sample number of this class to the total sample number.

$$x_{new} = s_k + rand(0,1) \times (s_k - x) \tag{8}$$

2) The Algorithm Level: Cost-Sensitive Learning

The traditional XGBoost algorithm assumes that the misclassification cost of all samples is the same. But for imbalanced classification problems such as fraud detection, intrusion detection, medical diagnostic classification, and failure prediction, the value of correctly identifying the minority classes (negative samples) is much higher. The cost-sensitive learning strategy is to distinguish the cost of different categories of samples when they are misclassified, thereby improving the prediction accuracy of the minority classes.

XGBoost has an important advantage is that it supports custom loss functions. So, at the algorithm level, we use the idea of cost-sensitive learning to improve the multi-class loss function *mlogloss* of XGBoost, and propose a new multi-class loss function $w-mlogloss$ based on weight offset. The main idea is that for normal data samples a lower weight α will be given to the model loss when classification result is wrong. At the same time, a higher weight β will be given to the model loss when failure samples are misclassified. So that the model pays more attention to the failure samples during optimization, and improve the accuracy of the prediction.

The loss function of the model is defined as Eq. (9). In which, N is the number of sample sets. K is the number of categories. $\hat{m}$ is the normal label. y_i is the actual label of the sample i. $\hat{y}_i$ is the prediction label of sample i, $0 < i \leq N$. $P(x)$ represents the probability that the prediction is x. $I(x)$ is the indication function. α, β are the weight coefficients, $0 < \alpha < \beta$. w_j is the score on the j^{th} leaf node in the tree f. T is the total number of leaf nodes in the tree f. λ and γ are custom parameters.

$$\begin{aligned} w-mlogloss = L(y_i,\hat{y}_i) = -\frac{1}{KN}\sum_{k\in K}\sum_{i=1}^{N}[\alpha^{I(y_i=\hat{m})} + \beta^{(1-I(y_i=\hat{m}))}][I(y_i=k)\log P(\hat{y}_i) \\ +(1-I(y_i=k))\log(1-\log P(\hat{y}_i)) + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2] \end{aligned} \tag{9}$$

The basic principle of SCS-XGBoost is to train k classification tree sets $F = \{f_1(x), f_2(x), \cdots, f_k(x)\}$. For a given training set $D = \{(X_n, Y_n)\}_{n=1}^{N}$, assign each input sample to different leaf nodes according to the attribute points. Each leaf node corresponds to one a real-time value *score*. When given a sample that needs to be predicted, the predicted result for that sample is the sum of the predicted scores for each tree. The SCS-XGBoost algorithm is shown in Algorithm 1.

4 Performance Evaluation Results

4.1 Experimental Environment and Data Set

To evaluate the proposed approach, we collect data from a cloud simulation platform in a lab environment, including 20 servers and 64 virtual machines. This paper uses the open source monitoring tool Ganglia to obtain the performance data (including CPU, Memory, Disk, Network and External State), and also extracts the alarm information and error log from the network management system. The collection period is 1 min. Tab. 1 shows the classification performance metrics we collected. We identified seven failure categories from the history log to mark the data, including abnormal fan speed, abnormal CPU temperature, CPU overload, memory leak, I/O exceptions, severe delays, and severe packet loss. We extracted approximately 222,500 data from June to December 2018, which is divided into a training set of 212,500, and a test set of 10,000. The positive and negative sample ratio is about 10:1.

Algorithm 1: The SCS-XGBoost algorithm

INPUT: Training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, the minority class dataset S_j ($S_j \in D$);
the resample ratio a_j; the number of iteration M; the regularization term Ω
a sample-weighted cost-sensitive loss function $L(y,f(x))$

OUTPUT: Classification tree $F_M(x)$

1: Random sample j subsets $\widehat{S}_j$ from S_j, $\left|\widehat{S}_j\right| = a_j|S_j|$

2: generate new sample S'_j=$SMOTE(\widehat{S}_j, a_j)$

3: update the training dataset $D' \leftarrow D + S'_j$;

4: get the new training dataset $D' = \{(x_1, y_1), (x_2, y_2), \cdots (x_{\hat{N}}, y_{\hat{N}})\}$

5: **for** $t = 1$ to M **do**

6: **for** $i = 1$ to $\hat{N}$ **do**

7: calculate the gradient on Ψ: $g_t(x_i) = -\partial_{f_{t-1}(x_i)} L(y_i,f(x_i))$

8: calculate the hessian on Ψ: $h_t(x_i) = -\partial^2_{f_{t-1}(x_i)} L(y_i,f(x_i))$

9: determine the tree structure $\{R_{jt}\}_{j=1}^{T}$ by maximizing the leaves score:

$$score = \max\left(score,\ \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma\right)$$

10: determine the optional leaf weight $\{w_{jt}\}_{j=1}^{T}$ given $\{R_{jt}\}_{j=1}^{T}$ by

$$w^*_{jt} = \arg\min_{w_j}\left(\sum_{i\in I_{jt}} (L(y_i,f_{t-1}(x_i) + w_j)) + \Omega(w_j)\right)$$

11: update $f_t(x) = f_{t-1}(x) + \sum_{j=1}^{T} w^*_{jt} I(x \in R_{jt})$

12: **end for**

13: $F_t(x) = F_{t-1}(x) + f_t(x)$

14: **end for**

15: **Return** $F_M(x)$

Table 1: Categorized performance metrics

Category	CPU	Memory	Disk	Network	External State
metrics	user percent idle percent IO wait hardware interrupts software interrupts system percent	total availability percent memory used memory active memory buffer memory cache memory slab memory swap memory	disk usage read count write count read byte write byte	sent byte receive byte sent packet receive packet error in/ error out delay packet loss	CPU temperature fans speed load balancing group count of normal alarm count of warning alarm count of critical alarm

Table 2: The training parameters of FP-STE

Model	HW-GRU	CNN	SCS-XGBoost
parameters	windows_size Δt_d: 20	N: 7	eta: 0.2
	epoch: 20	epoch: 15	max_depth: 4
	batch_size: 128	batch_size: 128	subsample: 1
	dropout: 0.5	dropout: 0.8	num_boost_round: 20000
	learning_rate: 0.1	learning_rate: 0.05	early_stop_round: 50

The experimental environment in this paper is a server with 8 G memory and 1.6 GHz CPU (Xeon e5-2603 v3). All the algorithms are written in Python 3.7 environment using Keras library and scikit-learn tools. The main training parameters of each component of FP- STE are set as shown in Tab. 2.

4.2 Performance Measures

In this section, we present some metrics to evaluate the performance of our proposed method. Precision, Recall, F1 and AUC are used to evaluate the prediction effect of a single category. Precision is the percentage of predicted failure events that are correctly labeled. Recall is the true failure percentage of all failure occurring in the environment. F1 represents the comprehensive performance in terms of correctness and accuracy. ROC (Receiver operating characteristic curve) is another tool for measuring the imbalanced data in classification, which is a comprehensive index reflecting the continuous variables of sensitivity and specificity [27]. One of the indicators for comparing different ROC curves is the area under the curve (AUC) which shows the average performance of the classifier for imbalanced and cost-sensitive problems. The details of these metrics are described in Eqs. (10)–(12).

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FP} \tag{11}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{12}$$

Since FP-STE is designed for multi-classification problems, we also use four Micro-averaging measures [28] to examine the quality of the overall classification, including Micro-Precision, Micro-Recall, Micro-F1, Micro-AUC. The calculation is similar to the formulas Eqs. (10)–(12), but is based on the cumulative True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) of all categories.

4.3 Experimental Results

In this section, we will evaluate the performance of our proposed method from three aspects: the comprehensive prediction effect, the effectiveness of feature extraction and the effectiveness of the improved algorithm. Because there are few prediction methods for node multi-type failure prediction in the academic field, we compare FP-STE with some algorithms used in single-type failure prediction, including SVM [12], Logistic Regression [18], LSTM [14], Random Forest [20].

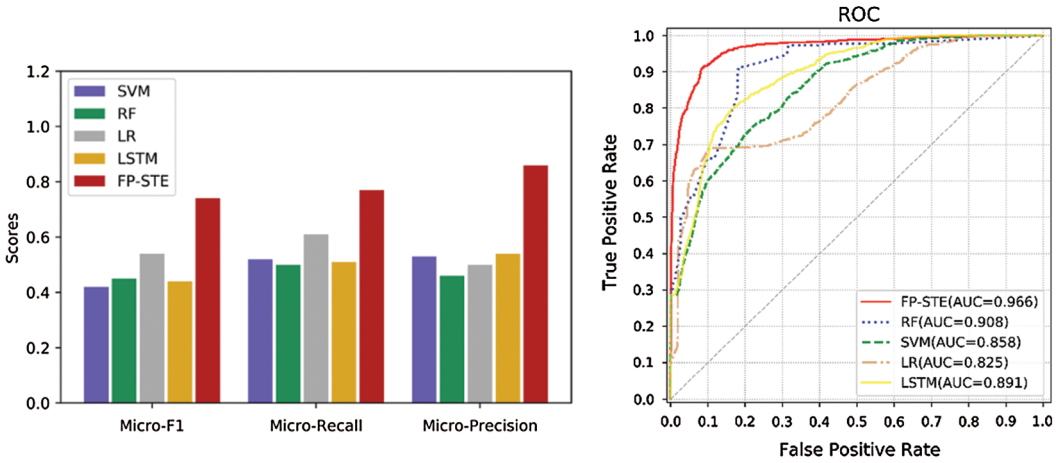
4.3.1 The Comprehensive Prediction Effect

The evaluation of the prediction effect is mainly divided into two parts: the overall prediction result and the prediction result in each category.

Firstly, we analyzed the overall prediction result of FP-STE. Tab. 3 and Fig. 6 show a comparison of FP-STE with other methods. It can be found out that among the comparison algorithms, LR has the highest Micro-Recall, which is 60%. LSTM has the highest Micro Precision, which is 54%. LR has the highest Micro-F1, which is 54%. In comparison, The Micro-Recall, Micro-Precision, and Micro-F1 of the FP-STE are 28%, 59%, and 37% higher than the best case of other algorithms. Therefore, it can be assured that the overall failure prediction effect has been greatly improved, and the number of missed and misjudged samples has been greatly reduced. Meanwhile, FP-STE has the highest AUC value 0.966, which is 5.8%, 10.8%, 14.1%, and 7.5% higher than other algorithms, indicating that the algorithm in this paper is more suitable for multi-class prediction of unbalanced samples. This is of great significance for failure prediction in real industrial scenarios.

Table 3: The overall results of FP-STE and comparative approaches

Metric	SVM	RF	LR	LSTM	FP-STE
Micro-Recall	52%	49%	60%	50%	77%
Micro-Precision	53%	46%	49%	54%	86%
Micro-F1	41%	44%	54%	42%	74%



(a) (b)

Figure 6: The overall results of FP-STE and comparative approaches (a) Micro-F1, Micro-Precision and Micro-Recall. (b) Micro-ROC

Secondly, we analyzed the prediction effect of FP-STE on each category. As shown in Fig. 7, FP-STE has basically the same prediction effect for failure categories and normal categories. Except for "cpu-T", "memory", and "fans", the F1 of all categories can reach over 70%, and the F1 of "I/O" and "delay" even reach up to 99% and 96%. However, the Precision, Recall and F1 of the failure classes obtained by other methods are significantly lower than the normal class, which indicates that FP-STE performs better in multi-failure identification compared with other methods.

Taking the LR algorithm with better comprehensive performance as an example, LR has a very good prediction effect on the two types of failure, such as "I/O" and "severe delay", where the F1 value is close to 1. But the prediction effect on CPU-related and memory-related failures are very inadequate. The F1 and Recall are both below 10%, indicating that the prediction results basically have no reference

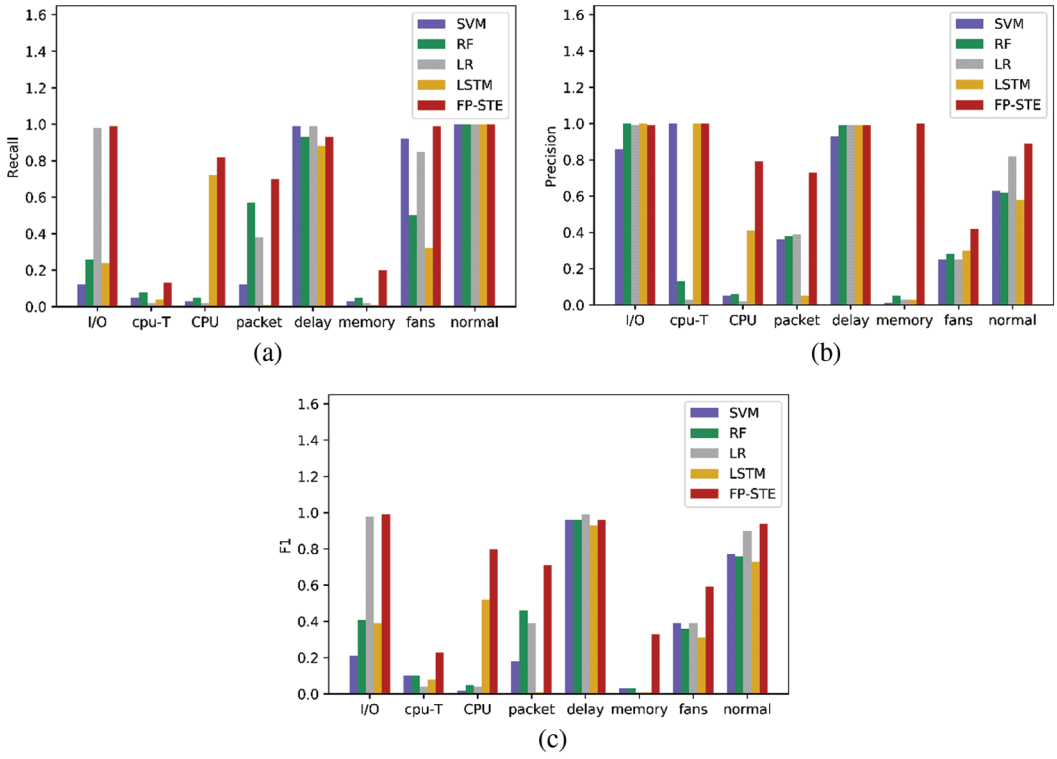


Figure 7: Precision, Recall and F1 of different failure classes. (a) Recall. (b) Precision. (c) F1

value. However, the F1 values of "severe delay", "I/O" and "normal" obtained by FP-STE has exceeded 90%. In terms of "CPU overload", "severe packet loss" and "fan" failures, the F1 values have also reached more than 60%. Even on the "CPU temperature" and "memory leak" which are unpredictable by all other methods, FP-STE can also provide a valuable F1 that is more than 30%, especially the Precision value can even reach 97% and 98%, respectively.

Other algorithms are similar to the LR algorithm, and they only have good prediction results for certain failure types, while FP-STE has higher stability and universality, showing significant advantages. But we also found the performance of FP-STE fluctuates a lot on extremely imbalanced failure classes, where "cpu-T", "memory", and "fans" achieve unsatisfactory results, which illustrates that there still exists improvement spaces for FP-STE.

4.3.2 The Effectiveness of HW-GRU and CNN

Next, we have analyzed the usefulness of feature extraction. FP-STE utilizes two base learners (HW-GRU and CNN) to incorporate the temporal and spatial features, respectively. So, we evaluated the usefulness of each type of features by applying these two models separately. The results are shown in Tab. 4.

It can be seen that if only using the original data without feature extraction, the Micro-F1 of SCS-XGBoost is 58% and the AUC is 0.935. If only the temporal features were added, the Micro-F1 would increase by 13.8% and the Micro-AUC would increase by 2.9%. If only the spatial features were added, the Micro-F1 would increase by 3.4% and the Micro-AUC would increase by 0.6%. The FP-STE method proposed in this paper uses both time and space features. Judging from experimental results, the performance improvement of FP-STE is the most obvious. The Micro-Recall increased by 26.2%, the Micro-Precision increased by 21.1%, the Micro-F1 increased by 27.6%, the Micro-AUC increased by 3.3%. In summary, the experimental results show that the temporal and spatial features extracted by

Table 4: The effectiveness of feature extraction

Metric	Original (SCS-XGBoost)	Tempotal only (HW-GRU + SCS- XGBoost)	Spatial only (CNN + SCS-XGBoost)	FP-STE
Micro-Recall	0.61	0.68	0.60	0.77
Micro-Precision	0.71	0.80	0.62	0.86
Micro-F1	0.58	0.66	0.60	0.74
Micro-AUC	0.935	0.962	0.941	0.966

HW-GRU and CNN are useful for node failure prediction, and the temporal features have more predictive power. Therefore, FP-STE can capture the spatio-temporal features behind the original data, and get the best prediction effect.

4.3.3 The effectiveness of the SCS-XGBoost

In addition, we have analyzed the performance improvement of the optimized algorithm SCS-XGBoost. As is shown in Fig. 8, our proposed method has a better classification effect on imbalanced data. In particular, compared with the traditional XGBoost algorithm, the improvement of the Micro-F1 brought about by the SMOTE + XGB algorithm which is only improved by the SMOTE method is about 12%, while the SCS-XGBoost achieved an improvement of about 22%. On the Micro-Recall indicator, the SMOTE + XGB has only achieved a 6% improvement, while SCS-XGBoost has improved by 15%. On the Micro-Precision indicator, the improvement ratio of the SCS-XGBoost algorithm is also higher than that of SMOTE+XGB algorithm by more than 10%. The experimental results show that SCS-XGBoost which adopts an integrated strategy of oversampling and cost-sensitive learning can better study the characteristics of minority and majority classes than other solutions that only use partial strategies. So, FP-STE is more suitable for actual node failure prediction scenarios.

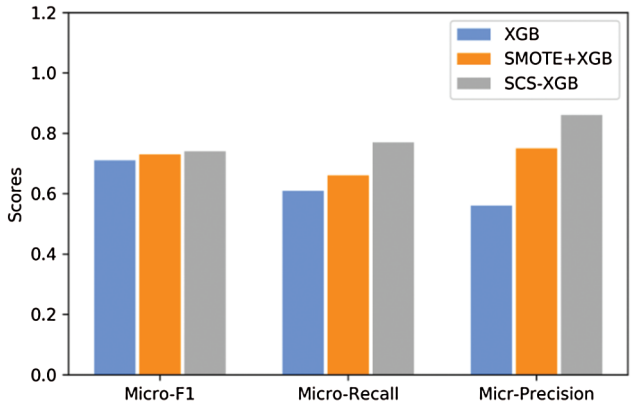


Figure 8: The comparation result of the effectiveness of the SCS-XGBoost

In summary, all the above experimental results confirm that FP-STE is an excellent failure prediction method and has a better prediction effect than the other methods. It is noted that in our data magnitude, the training duration of FP-STE can be controlled within 2 minutes, and the prediction time is controlled in the second level, so it provides more possibilities for its application in real production environments.

5 Conclusion

Data-driven failure prediction technology will play an increasingly important role in the intelligent management and system maintenance of the next generation networks. In order to improve the reliability of data center services, this paper proposes a node failure prediction method named FP-STE. This method uses HW-GRU and CNN to extract the features of time and space dimensions from different sources, and inputs the extracted features into the improved model SCS-XGBOOST to predict the failure tendency of the nodes. The experimental results show that the proposed method has higher prediction accuracy and is more conducive to multi-class failure prediction. However, there are still further improvements in FP-STE. The future research work can be carried out from the following aspects: this method only verifies the failure types that can be obtained in the laboratory environment, and the prediction effect for other failure types needs to be further verified. In addition, to improve the effectiveness and efficiency of the model for industrial areas, more improved strategies should be adopted in FP-STE to speed up the training of the model and further improve the accuracy of failure prediction under the extreme imbalance of samples. We hope our work can provide some references for the failure prediction research of large-scale data centers in the future.

Funding Statement: This work was supported in part by National Key Research and Development Program of China (2019YFB2103200), NSFC (61672108), Open Subject Funds of Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory (SKX182010049), the Fundamental Research Funds for the Central Universities (500419319 2019PTB-019), the Industrial Internet Innovation and Development Project 2018 of China.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Chalermarrewong, T., Achalakul, T., See, S. C. W. (2012). Failure prediction of data centers using time series and fault tree analysis. *IEEE 18th International Conference on Parallel and Distributed Systems, Singapore,* 794–799.
2. Lin, Q., Hsieh, K., Dang, Y., Zhang, H., Sui, K. et al. (2018). Predicting node failure in cloud service systems. *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Lake Buena Vista, Florida,* 480–490.
3. Mariani, L., Pezzè, M., Riganelli, O., Xin, R. (2020). Predicting failures in multi-tier distributed systems. *Journal of Systems and Software, 161,* 110464. DOI 10.1016/j.jss.2019.110464.
4. Salfner, F., Lenk, M., Malek, M. (2010). A survey of online failure prediction methods. *ACM Computing Surveys, 42(3),* 1–42. DOI 10.1145/1670679.1670680.
5. Li, L., Vaidyanathan, K., Trivedi, K. S. (2002). An approach for estimation of software aging in a web server. *Proceedings International Symposium on Empirical Software Engineering, Nara, Japan,* IEEE, 91–100.
6. Hoffman, G., Malek, M. (2006). Call availability prediction in a telecommunication system: a data driven empirical approach. *25th IEEE Symposium on Reliable Distributed Systems (SRDS'06), Leeds, UK,* 83–95.
7. Miao, H., Li, B., Sun, C., Liu, J. (2019). Joint learning of degradation assessment and RUL prediction for aeroengines via dual-task deep LSTM networks. *IEEE Transactions on Industrial Informatics, 15(9),* 5023–5032. DOI 10.1109/TII.2019.2900295.
8. Jin, S., Zhang, Z., Chakrabarty, K., Gu, X. (2018). Failure prediction based on anomaly detection for complex core routers. *Proceedings of the International Conference on Computer-Aided Design, San Diego, CA, USA,* 1–6.
9. Zheng, W., Wang, Z., Huang, H., Meng, L., Qiu, X. (2016). SPSRG: a prediction approach for correlated failures in distributed computing systems. *Cluster Computing, 19(4),* 1703–1721. DOI 10.1007/s10586-016-0633-2.

10. Sun, M., Qian, H., Zhu, K., Guan, D., Wang, R. (2017). Ensemble learning and SMOTE based fault diagnosis system in self-organizing cellular networks. *GLOBECOM 2017–2017 IEEE Global Communications Conference, Singapore,* 1–6.
11. Chigurupati, A., Thibaux, R., Lassar, N. (2016). Predicting hardware failure using machine learning. *Annual Reliability and Maintainability Symposium (RAMS), Tucson, AZ, USA,* 1–6.
12. Mohammed, B., Awan, I., Ugail, H., Younas, M. (2019). Failure prediction using machine learning in a virtualised HPC system and application. *Cluster Computing, 22(2),* 471–485. DOI 10.1007/s10586-019-02917-1.
13. Xu, C., Wang, G., Liu, X., Guo, D., Liu, T. Y. (2016). Health status assessment and failure prediction for hard drives with recurrent neural networks. *IEEE Transactions on Computers, 65(11),* 3502–3508. DOI 10.1109/TC.2016.2538237.
14. Zhang, L., Zhu, X., Zhao, S., Xu, D. (2017). A novel virtual network fault diagnosis method based on long short-term memory neural networks. *IEEE 86th Vehicular Technology Conference (VTC-Fall), Toronto, Canada,* 1–5.
15. Fu, X., Ren, R., McKee, S. A., Zhan, J., Sun, N. (2014). Digging deeper into cluster system logs for failure prediction and root cause diagnosis. *IEEE International Conference on Cluster Computing (CLUSTER), Madrid, Spain,* 103–112.
16. Yu, Y., Chen, H. (2019). An approach to failure prediction in cluster by self-updating cause-and-effect graph. *International Conference on Cloud Computing, Cham, Springer, 114–129.*
17. Liang, Y., Zhang, Y., Xiong, H., Sahoo, R. (2007). Failure prediction in IBM BlueGene/L event logs. *Seventh IEEE International Conference on Data Mining, Omaha, Nebraska, USA,* 583–588.
18. Botezatu, M. M., Giurgiu, I., Bogojeska, J., Wiesmann, D. (2016). Predicting disk replacement towards reliable data centers. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA,* 39–48.
19. Ganguly, S., Consul, A., Khan, A., Bussone, B., Richards, J. et al. (2016). A practical approach to hard disk failure prediction in cloud platforms: big data model for failure management in datacenters. *IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService), Oxford, United Kingdom,* 105–116.
20. Zhang, S., Liu, Y., Meng, W., Luo, Z., Bu, J. et al. (2018). Prefix: switch failure prediction in datacenter networks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2(1),* 1–29. DOI 10.1145/3179405.
21. Sun, X., Chakrabarty, K., Huang, R., Chen, Y., Zhao, B. et al. (2019). System-level hardware failure prediction using deep learning. *56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA,* 1–6.
22. Zhao, R., Wang, D., Yan, R., Mao, K., Shen, F. et al. (2017). Machine health monitoring using local feature-based gated recurrent unit networks. *IEEE Transactions on Industrial Electronics, 65(2),* 1539–1548. DOI 10.1109/TIE.2017.2733438.
23. Zilly, J. G., Srivastava, R. K., Koutník, J., Schmidhuber, J. (2017). Recurrent highway networks. *Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia*.
24. Liu, C. L., Hsaio, W. H., Tu, Y. C. (2018). Time series classification with multivariate convolutional neural network. *IEEE Transactions on Industrial Electronics, 66(6),* 4788–4797. DOI 10.1109/TIE.2018.2864702.
25. Chen, T., Guestrin, C. (2016). Xgboost: a scalable tree boosting system. *Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA,* 785–794.
26. Khan, S. H., Hayat, M., Bennamoun, M., Sohel, F. A., Togneri, R. (2017). Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems, 29(8),* 3573–3587.
27. Wu, Z., Lin, W., Ji, Y. (2018). An integrated ensemble learning model for imbalanced fault diagnostics and prognostics. *IEEE Access, 6,* 8394–8402. DOI 10.1109/ACCESS.2018.2807121.
28. Tong, V., Tran, H. A., Souihi, S., Mellouk, A. (2018). A novel QUIC traffic classifier based on convolutional neural networks. *IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates,* 1–6.