frontiers in Computer Science

ORIGINAL RESEARCH
published: 01 October 2021
doi: 10.3389/fcomp.2021.710384

XFinder: Detecting Unknown Anomalies in Distributed Machine Learning Scenario

Haizhou Du[1]*, *Shiwei Wang*[2] *and Huan Huo*[3]

[1]School of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China, [2]Jiangsu Huaneng Smart Energy Supply Chain Technology Co., Ltd., Nanjing, China, [3]School of Computer Science, University of Technology Sydney, Sydney, NSW, Australia

In recent years, the emergence of distributed machine learning has enabled deep learning models to ensure data security and privacy while training efficiently. Anomaly detection for network traffic in distributed machine learning scenarios is of great significance for network security. Although deep neural networks have made remarkable achievements in anomaly detection for network traffic, they mainly focus on closed sets, that is, assuming that all anomalies are known. However, in a real network environment, unknown abnormalities are fatal risks faced by the system because they have no labels and occur before the known anomalies. In this study, we design and implement XFinder, a dynamic unknown traffic anomaly detection framework in distributed machine learning. XFinder adopts an online mode to detect unknown anomalies in real-time. XFinder detects unknown anomalies by the unknowns detector, transfers the unknown anomalies to the prior knowledge base by the network updater, and adopts the online mode to report new anomalies in real-time. The experimental results show that the average accuracy of the unknown anomaly detection of our model is increased by 27% and the average F1-Score is improved by 20%. Compared with the offline mode, XFinder's detection time is reduced by an average of approximately 33% on three datasets, and can better meet the network requirement.

Keywords: unknown anomaly detection, distributed machine learning, prior knowledge, incremental learning, buffer

OPEN ACCESS

Edited by:
Guangdong Bai,
The University of Queensland, Australia

Reviewed by:
Jianxin Li,
Deakin University, Australia
José Antonio Álvarez-Bermejo,
University of Almeria, Spain

***Correspondence:**
Haizhou Du
duhaizhou@shiep.edu.cn

Specialty section:
This article was submitted to Computer Security, a section of the journal Frontiers in Computer Science

Received: *16 May 2021*
Accepted: *20 August 2021*
Published: *01 October 2021*

Citation:
Du H, Wang S and Huo H (2021) XFinder: Detecting Unknown Anomalies in Distributed Machine Learning Scenario. Front. Comput. Sci. 3:710384. doi: 10.3389/fcomp.2021.710384

1 INTRODUCTION

With the rapid development of big data, data privacy and security have attracted more and more public attention. With the increase of data size and model complexity, it becomes more and more difficult for a single server to accomplish a machine learning task. To address the problem, distributed machine learning is developed. Distributed machine learning (Galakatos et al., 2018) uses multi-node machine learning algorithms and systems, which are designed to improve performance, accuracy, and scale to larger input data sizes. **Figure 1** is a typical distributed machine learning architecture. In this scheme, the whole dataset is divided into several subsets and stored distributedly on nodes. Each node also keeps a copy of the model and trains it based on the locally available part of the data. The central server aggregates the parameters from each node to go to the next iteration and eventually converge to a solution. Network traffic is the carrier of information transmission and interaction in cyberspace, anomaly detection for

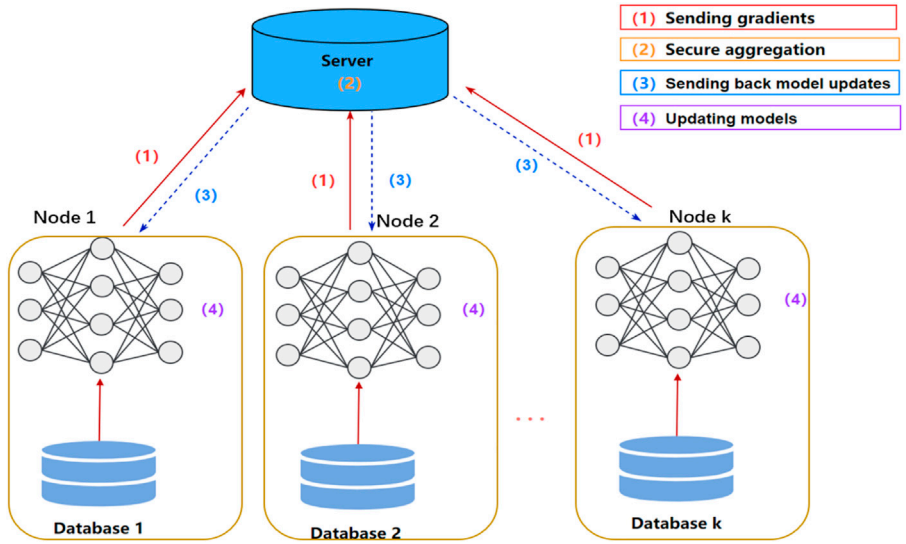

FIGURE 1 | Distributed machine learning architecture. In this scheme, the whole dataset is divided into subsets and stored distributedly on nodes. Each node has the complete model but only works with part of training data. The central server aggregates the parameter from each node's information of each local model and transfers the global parameters to the local model to guide the model training. Then, if the performance of the new global model is still not satisfactory enough, a new round of training can be started.

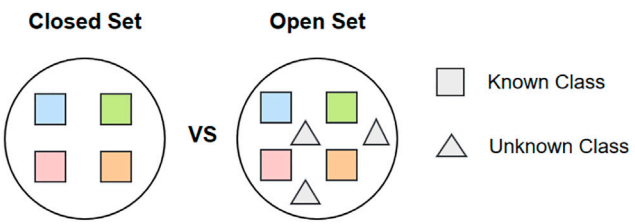

FIGURE 2 | Closed space vs. Open space. In closed space detection, all categories are known (squares in the picture), while in an open space detection not only known categories but also unknown categories (triangles in the picture) are included.

network traffic in a distributed machine learning scenarios is of great significance for network security.

Many methods have been used to detect abnormal network traffic, such as the information entropy measurement method (Yang, 2019), deep learning methods (Chouhan and Khan, 2019; Vinayakumar et al., 2019), and transfer learning (Niu et al., 2019). These methods have achieved satisfactory results in recognition accuracy. However, almost all of them are proposed for closed set scenarios, where all categories are known in advance. The detection of unknown anomalies is an open-set scenario, as shown in **Figure 2**. Compared with known anomalies, unknown anomalies are the most threatening risks to the system because they have no labeled samples and occur before the known anomalies. Therefore, it is necessary to detect unknown network anomalies in an open space.

In recent years, the detection methods of unknown anomaly traffic mainly include methods based on open-set recognition, transfer learning, and incremental learning. Sabeel et al. (2019) examined the performance of two famous machine learning models, deep neural networks (DNNs) and long short-term memory (LSTM), for detecting unknown DDoS/DoS attacks. The open deep network (P-ODN) (Shu et al., 2020) introduced prototype learning (Yang et al., 2018) into open-set recognition to derive more discriminative features. It has been proven that more characteristic features that have a more significant margin among categories will further improve the performance of unknown detection. Zhao et al. (2017) presented a transfer learning-enabled network attack detection framework to enhance the detection of new network attacks in a target domain. Moreover, many incremental learning methods are used to handle new instances of known categories (Tveit and Hetland, 2003; Yeh and Darrell, 2008). However, model retraining requires considerable time, which cannot meet the needs of the network for detecting unknown anomalies in real time.

To meet the needs of a proactive strategy in the network environment, detecting unknown anomalies in the online mode is a good solution. However, unknown anomaly detection in distributed machine learning scenario may face the following challenges:

- Unknown anomalies have no labeled samples, which are different from known anomalies, so how to distinguish unknown and known anomalies is the first difficulty for detecting unknown anomalies in distributed machine learning scenario.
- Unknown anomalies may contain many different types, so how to differentiate the unknown anomaly types is the second challenge for detecting unknown anomalies in distributed machine learning scenario.

- Model training takes time and usually cannot meet the needs of the online mode, so how to adapt to online mode detection is the third challenge for detecting unknown anomalies in distributed machine learning scenario.

To solve the above challenges, this study proposes an unknown anomaly detection framework in distributed machine learning scenario, called XFinder. Our main contributions are as follows:

- For the first time, we design XFinder, which is implemented in distributed machine learning scenario. XFinder detects multi-class unknown anomalies through three components: the *unknowns detector*, the *buffer*, and the *network updater*.
- To the best of our knowledge, this is the first time that the buffer technique (Arge, 2003) has been used in incremental training to improve the detection performance of the model. The *buffer* is designed to store the unknown anomaly information detected to help the model adapt to the online mode and differentiate anomaly types.
- The *unknowns detector* leverages the limited labeled data as prior knowledge to help separate the known and unknown anomalies. The *network updater* helps distinguish the types of unknown anomalies by converting the unknown into the known.
- Experiments on three public datasets are conducted to evaluate the performance of the model. With the *buffer*, the average accuracy of the unknown anomaly detection of XFinder is increased by 27%, and the average F1-score is increased by 20%. Compared with the offline mode, XFinder's detection time is reduced by an average of approximately 33% on the three datasets.

The rest of the article is organized as follows. We introduce the design and analysis of the XFinder framework in **Section 2**. We present the evaluation results in **Section 3**, discuss related study in **Section 4**, and conclude our study in **Section 5**.

2 XFINDER DESIGN

In this section, we describe the XFinder framework for unknown network anomaly detection in distributed machine learning scenario. The framework is composed of three parts, which are elaborated on in the following subsections.

2.1 Problem Statement

Assuming that given a set of $N + K$ data $\mathbf{X} = \{X_1, \ldots, X_N, X_{N+1}, \ldots, X_{N+K}\}$ with $X_i \in \mathbb{R}^D$, in which $\mathbf{U} = \{X_1, X_2, \ldots, X_N\}$ is unlabeled data and $\mathbf{V} = \{X_{N+1}, X_{N+2}, \ldots, X_{N+K}\}$ with $K \ll N$ is a very small set of labeled anomalies that provide some prior knowledge of anomalies. Our goal is to update the initial model $\mathbf{I}$ and classify different unlabeled data $\mathbf{U}$ in distributed machine learning.

2.2 The XFinder Model in Distributed Machine Learning

We designed a novel unknown network anomaly detection model called XFinder, as shown in **Figure 3**. The XFinder is the local model, which sends the parameters of each detection result to the server, and the server integrates all the results, and then returns the global parameters for the model update.

For the XFinder, first, an initial classification model is obtained through the initial training phase. As shown in **Figure 4**, we take the CNN and LSTM to learn the known's input and apply a softmax classifier to train the initial classification model. The initial model is used later in the incremental training phase. In the incremental training phase, we take the incremental training set (containing both the known set and the unknown set) and the outputs of the initial training phase as an input and then detect the unknowns.

Our incremental training phase consists of three major components:

- **Unknowns Detector**: The unknowns detector determines whether the current input data sample is from a known class or an unknown class. The detected new class is automatically labeled and added to the network updater and buffer.
- **Buffer**: To avoid the situation in which the detected unknown anomalies are learned again in the next training due to the untimely update of the network, we set up a buffer structure to store the detected unknown features and labels. When the input sample is detected as an unknown class, it needs to be compared with the samples in the buffer again. In addition, the weight information of these new classes is stored in the buffer to help update the network.
- **Network Updater**: The network updater adds these new categories found in the unknowns detector to update the network. By updating the network, the unknown class is transformed into a known class.

2.3 Unknowns Detector Module

In the initial training phase, we apply the cross-entropy loss to train the classification ability of the neural networks, which we denote as $loss_C$:

$$loss_C = -\frac{1}{S}\sum_{i=1}^{S}[l_i * \log(softmax(f_i))], \tag{1}$$

where S is the batch size of neural networks, f_i is the feature of the ith sample in the batch, and l_i is the ground truth.

Since prototype learning is effective in increasing interclass variation (Yang et al., 2018), we introduce prototype learning into open set recognition tasks, and further use prototypes to detect unknowns. Specifically, an $N \times N$ prototype matrix is initialized with zeros, where N is the number of known classes. Each row of the prototype matrix, shown in different colors in **Figure 5**, represents the prototype (or center) of each known class.

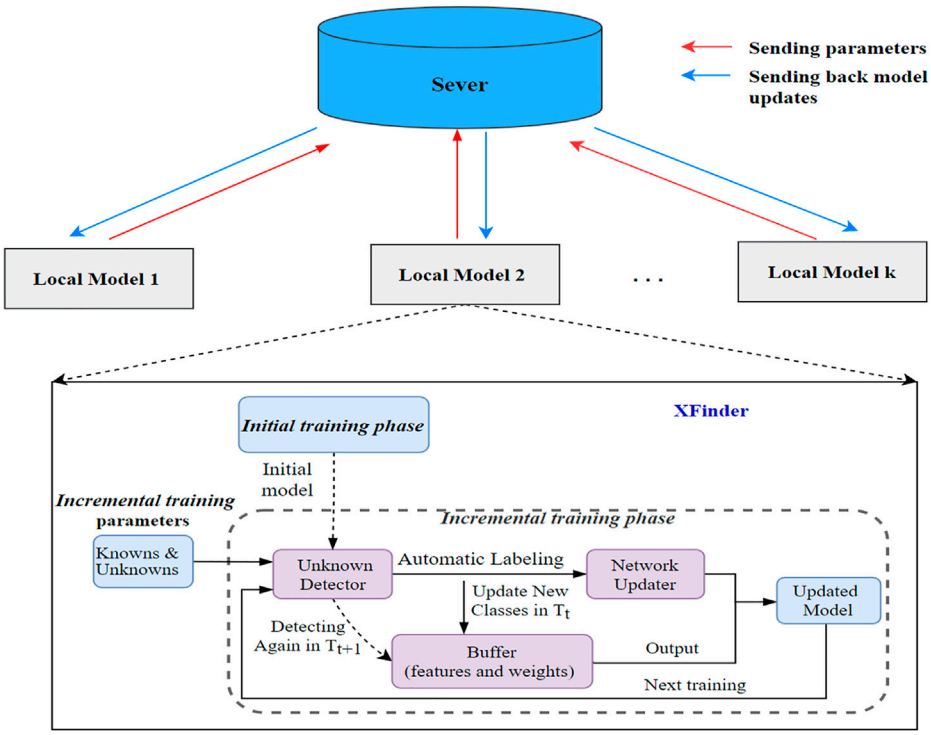

FIGURE 3 | The XFinder model in distributed machine learning. XFinder detecting unknown anomalies based on incremental learning. The unknown detector detects whether the input sample is known or unknown based on the initial training model, and the detected unknown sample is automatically labeled and added to the buffer to help update the network. The updated model is used for the next incremental training.

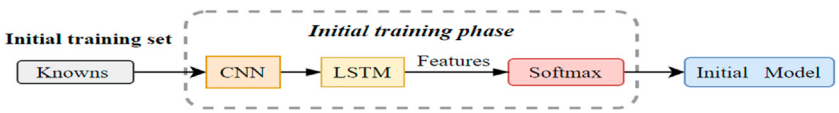

FIGURE 4 | The initial training phase.

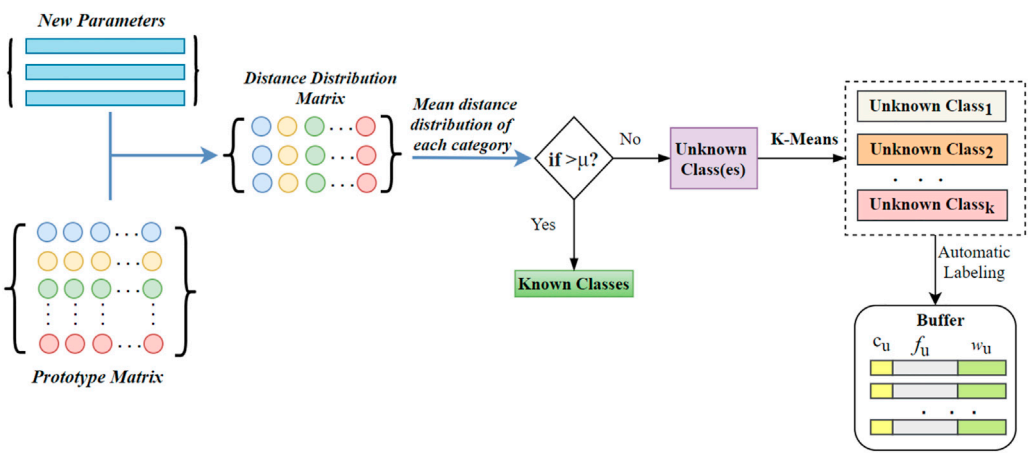

FIGURE 5 | Unknown's detector module.

Therefore, we modify the prototype loss first proposed by Yang et al. (2018) and apply it to our framework to train known prototypes. The prototype loss consists of a $L2$ loss ($loss_2$) and a distance-based classification loss ($loss_d$). Then the two losses are combined with a weighted argument ω:

$$loss_p = \omega * Loss_2 + Loss_d. \tag{2}$$

The S prototypes are selected based on the label of features, where S is the batch size of the initial network. The $L2$ loss ($loss_2$) is applied to the chosen prototypes and the features, guiding the prototypes to learn the characters of the features:

$$loss_2 = -\frac{1}{2S}\sum_{i=1}^{S}(f_i - p_i)^2, \tag{3}$$

where f_i is the feature of the ith data sample in the batch, and p_i is the corresponding prototype.

Since the prototypes and features are trained jointly, it is unstable to simply use $L2$ loss to make prototypes similar to the features. The prototypes would be easily misled by some outliers of the training data samples. Therefore, we refer to Shu et al. (2020) to add the distance-based classification loss to improve the classification capacity of the prototypes and increase the penalty of misclassified samples.

By calculating the Euclidean distance between each feature and each class prototype, a distance distribution matrix D is obtained as:

$$D_{ij} = \frac{1}{\|f_i - p_j\|_2^2 + \varepsilon}, \tag{4}$$

where $i = 1, \ldots, S$ and $j = 1, \ldots, N$. We take the reciprocal of distances between features and prototypes so that the features close to the prototype can obtain a larger probability value. We apply the $\varepsilon = 0.001$ to avoid dividing by zero. Therefore, classification can be implemented by assigning labels according to the largest value in each row of D. Then, the cross-entropy loss is applied on D:

$$loss_d = -\frac{1}{S}\sum_{i=1}^{S}[l_i * \log(softmax(D[i,:]))], \tag{5}$$

where the S is the batch size (the same as the row number of D), l_i is the ground truth, and $D[i,:]$ denotes the ith row of D.

As shown in **Figure 5**, when detecting unknown, the distance distribution matrix between features and prototypes is calculated first. Then, thresholds are calculated based on the mean distance distribution and then applied to the test samples to detect unknowns.

$$\eta_i = \frac{1}{C_i}\sum_{j=1}^{C_i} M_{ij}, \tag{6}$$

$$\mu_i = \rho * \eta_i, \tag{7}$$

where M_{ij} is the maximal confidence value of the ith correctly classified sample of class j. C_i is the number of correctly classified sample sets $\mathbf{C}$ of class i. ρ is the empirical parameter. When all the confidence values of the sample are less than μ_i, we consider the sample as unknown.

Different from previous studies (Bendale and Boult, 2016; Shu et al., 2018; Shu et al., 2020), which regard all unknown classes as one class, we further seek potentially different categories by clustering unknown samples to help detect the unknown in the next incremental training. Assuming that the unknown set in the Tth training is U_T, we use k-Means to partition the U_T into k clusters $\{C_1, C_2, \ldots, C_k\}$. The k-Means algorithm has the advantages of simple principle, easy implementation, fast convergence, and strong interpretability. For each cluster, samples are automatically labeled and then added to the buffer to help the next unknowns detection.

2.4 Buffer Module

To avoid the detected unknown information not being effectively utilized, which is caused by the untimely update of the network, we set buffer B to store the unknown information that has been detected at the current time.

Assuming that m new categories are found on the Tth detection and q new categories are found on the $T + 1$th detection, the update of buffer B is as follows:

$$B_T = \left[\left(\sum_{u=1}^{m}(f_u, l_u)\right), W_T\right], \tag{8}$$

$$B_{T+1} = B_T \cup \left[\left(\sum_{u=1}^{q}(f_u, l_u)\right), W_T + 1\right], \tag{9}$$

where f_u is the feature of uth classes in unknown samples, l_u is the label of uth classes, and W_T and $W_T + 1$ represent weight vectors at Tth and $T + 1$th times, respectively.

When the input sample is detected as unknown, it needs to be compared with the sample in the buffer again to determine whether the unknown has ever appeared. If the sample feature is found to be a known class in the buffer, the class to which the sample belongs is output. Otherwise, the sample is considered to be a new unknown.

2.5 Network Updater Module

After detecting the unknowns, the unknown sample is labeled. Then, these samples are used to update the network model. It is time-consuming and a waste of computing resources to retrain the entire system with the known data and new samples. It is also easy to overfit because the new classes are far short of training samples. In Shu et al. (2020), an update method for transferring knowledge from the training model was proposed, which helps to speed up the training phase and requires very few manual annotations.

In each iteration of the incremental training phase, the distance distribution between prototypes and the new sample features is calculated first. Then, the weight distribution is obtained by applying mean normalization. Different from Shu et al. (2020), in which all unknown classes are treated as one new class with every incremental training, our method can find different classes due to the buffer setting. Therefore, our weight update considers whether the detection sample is known or unknown.

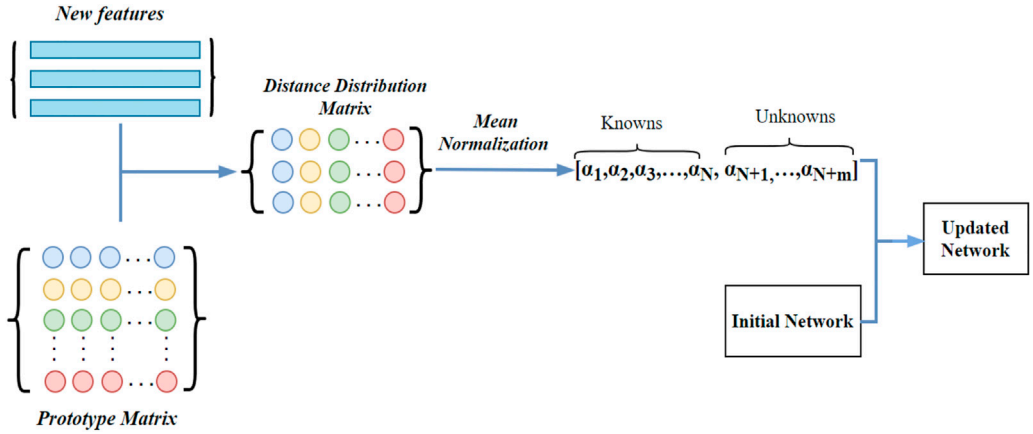

FIGURE 6 | Network updater module.

As shown in **Figure 6**, the weight distribution is $[\alpha_1, \alpha_2, \ldots, \alpha_N, \alpha_{N+1}, \ldots, \alpha_{N+m}]$, where $[\alpha_1, \alpha_2, \ldots, \alpha_N]$ are known's weight distribution, $[\alpha_{N+1}, \ldots, \alpha_{N+m}]$ are unknowns weight distribution, and $\sum_{n=1}^{N+m} \alpha_n = 1$. Assuming that A is the set of detected samples, the new weight ω_{n+1} is initialized as:

$$\omega_{n+1} = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} \alpha_n * \omega_n, & \text{if } A \text{ are all knowns} \\ \frac{1}{N}\sum_{n=1}^{N} \alpha_n * \omega_n + \frac{1}{M}\sum_{n=1}^{M} \alpha_m * \omega_m, & \text{else} \end{cases}, \quad (10)$$

where ω_n and ω_m are the weight columns of the nth class and mth class, respectively. N is the number of current classes, M is the number of unknown classes, and unknown weight information for M classes has been saved in the buffer.

Based on the above, the initial network is updated, and the updated network is used to detect unknowns in the next incremental training. With the update of the network, unknown anomalies are transformed into known anomalies, and the types of anomalies in the buffer increases constantly. Finally, the XFinder realizes the detection of multiple types of unknown anomalies.

TABLE 1 | The details of three datasets.

Datasets	Features	Class	Amount
KDD CUP 99	41	Normal	60,592
		DoS	2,29,853
		Probing	4,166
		R2L	16,189
		U2R	228
UNSW-NB 15	43	Normal	56,000
		Generic	40,000
		Exploits	33,393
		DoS	18,184
		Reconnaissance	12,264
		Analysis	10,491
		Backdoor	2000
		Shellcode	1746
		Worms	1,133
		Fuzzers	130
CICIDS 2017	78	Normal	4,40,031
		DoS Slowhttptest	5,499
		DoS Hulk	2,31,073
		DoS slowloris	5,796

3 EXPERIMENTS AND EVALUATION

In this section, we first introduce the experimental setup, including an experimental environment, datasets, parameters setting and metrics, and then carry out experimental evaluation from three aspects of initial training, single machine vs. distribution machine learning mechanism, unknown anomaly detection, and running time.

3.1 Experiments Setup

3.1.1 Experimental Environments

All the experiments were executed with the following specifications: GPU: Gfx 2.10 GHz, processor: AMD Ryzen 5 3550H, and RAM: 16 GB. Operating System: Windows 10 64bit. Model implementation: Dask, Tensorflow 2.0.0, Keras 2.3.1, and scikit-learn 0.21.3.

3.1.2 DataSets Introduction

To verify the effectiveness of XFinder, we conducted experiments on three public datasets, including KDD CUP 99, UNSW-NB 15, and CICIDS 2017. The details of three datasets as shown in **Table 1**. The data were preprocessed and numeralized before the experimental evaluation. In our experiments, we divide the datasets into known's and unknown's to simulate the open world scenarios.

- **KDD CUP 99 Dataset**[1]

The KDD CUP 99 dataset is a well-known benchmark in the research of network anomaly detection. The types of abnormal

[1]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

data in the KDD CUP 99 dataset are divided into four categories: DoS, R2L, U2R, and probing attack (Tavallaee et al., 2009). In this study, we use normal data and DoS anomaly data as known samples to train the initial model, and the remaining 3 anomalies as test samples to evaluate the unknown detection capability of the model.

- **UNSW-NB 15 Dataset**[2]

The UNSW-NB 15 dataset is a comprehensive dataset that do inclusively reflect network traffic and modern low footprint attacks (Moustafa and Slay, 2015). It has nine categories of attacks, namely, Generic, Fuzzers, Analysis, Backdoors, DoS, Exploits, Reconnaissance, Shellcode, and Worms. We use normal data and Generic anomaly data as known samples to train the initial model, and the remaining 8 anomalies as test samples to evaluate the unknown detection capability of the model.

- **CICIDS 2017 Dataset**[3]

The CICIDS 2017 dataset contains normal traffic and the latest common attacks, which similar to real-world data (Sharafaldin et al., 2018). In the experiment, we use normal data and one type of anomaly as known data for initial training, and randomly select three kinds of abnormal traffic from the dataset as unknown attacks, namely, Dos_Hulk, Slowhttptest, Slowloris.

3.1.3 Parameters Setting

In the initial training phase, known data samples are provided for the CNN-LSTM to train a classifier with the knowns. We use Adam (Kingma and Ba, 2015) as an optimizer to perform gradient descents and adopt early stopping to avoid overfitting. The sub-sampling size is set to 256 and epoch set to 30, which can get the convergence result in the experimental evaluation. Once the initial training phase is completed, the system will use the trained model to get activation values for detecting unknown category sample process. We test XFinder on a set of known and unknown samples which none of them appear in the training phase. To avoid overfitting, we set the experiment execution 10 times and take the average to get the result.

3.1.4 Metrics

We evaluate the performance of XFinder based on the following metrics:

- *Accuarcy*: defined as the percentage of correctly classified records over the total number of records.

$$Accuarcy = \frac{TP + TN}{TP + FP + TN + FN}, \tag{11}$$

where *TP* is the true positives, *FP* is the false positives, *TN* is the true negatives, and *FN* is the false negatives.

- *F1-Score*: defined as the harmonic mean of precision and recall and represents a balance between them.

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

where

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}, \tag{13}$$

where *Precision* is defined as the ratio of the number of true positives (*TP*) records divided by the sum of true positives (*TP*) and false positives (*FP*) classified records, and *Recall* is defined as the ratio of number of true positives records divided by the sum of true positives and false negatives (*FN*) classified records.

3.2 Initial Training Evaluation

In the initial training phase, the initial training set is all known samples. We use 10% of the initial training set as the validation set to test the performance of the initial model. **Figures 7–9** show the accuracy and loss performance of the initial model on KDD CUP 99, UNSW-NB 15, and CICIDS 2017 datasets during the initial training phase.

As can be seen from **Figures 7–9**, the verification set accuracy of the initial model on the three data sets is above 0.90 and can even reach 0.98 on the KDD CUP 99 dataset. After training 20 epoch, the accuracy of the verification set on the CICIDS 2017 dataset exceeds the accuracy of the training set, which shows that our model has good learning ability. It also lays a good foundation for the detection of unknown anomalies in subsequent incremental training.

3.3 Scalability in Distribution Machine Learning Scenario

To verify the feasibility of anomaly detection for network traffic under distribution machine learning mechanism, we evaluate the performance of k worker models in federated learning to detect unknown anomalies on the UNSW-NB 15 dataset, where $k = 1$ represents model detection performance in the single machine.

Figure 10 shows the detection performance with different worker models. We can see that the detection results of using multiple worker models are better than the detection performance in a single mechanism ($k = 1$). When $k = 3$, the detection results showed the best performance, with accuracy of 0.9163, f1-score of 0.8355, and average running time of multi-class anomaly detection of 7.4386 s. Therefore, the network anomaly detection under federated learning mechanism is effective, and we use $k = 3$ for subsequent experiments.

3.4 Unknowns Anomaly Detection Evaluation

We evaluate the effectiveness of the XFinder model in terms of the *Online Mode vs. Offline Mode*, *Multiple Types of Unknowns Detection*, and *Buffer Effectiveness*.

[2]https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

[3]https://www.unb.ca/cic/datasets/ids-2017.html

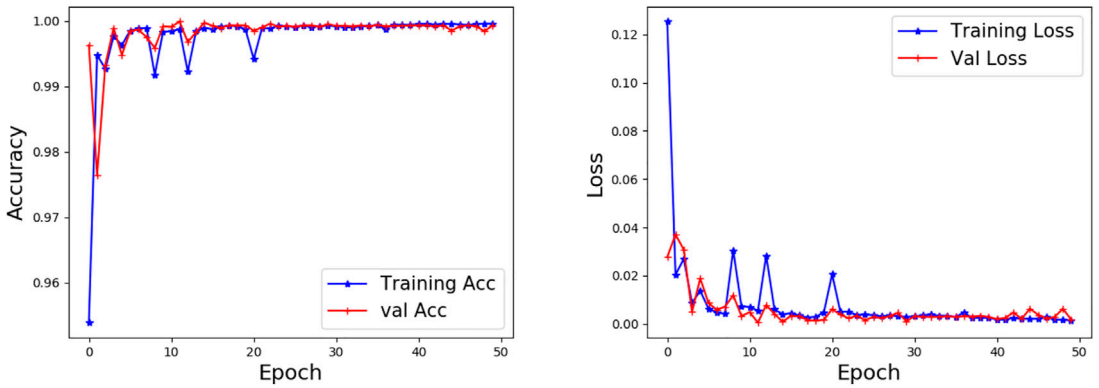

FIGURE 7 | Loss and accuracy performance of the initial model on KDD CUP 99 datasets.

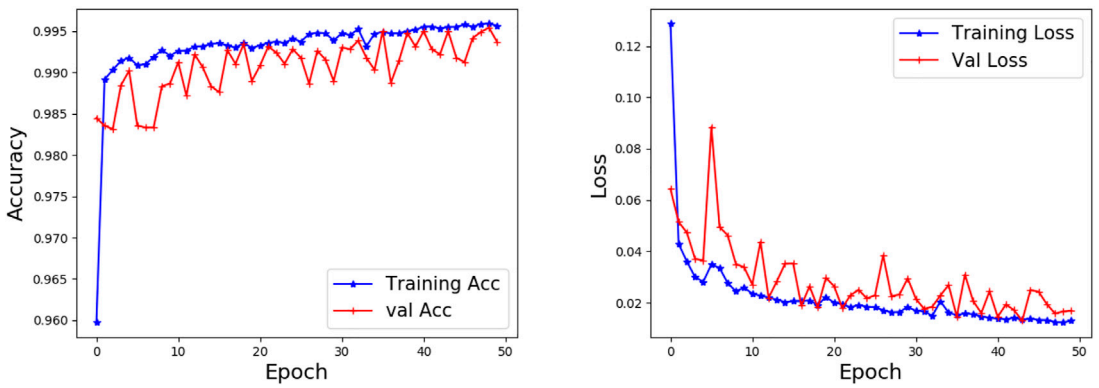

FIGURE 8 | Loss and accuracy performance of the initial model on UNSW-NB 15 datasets.

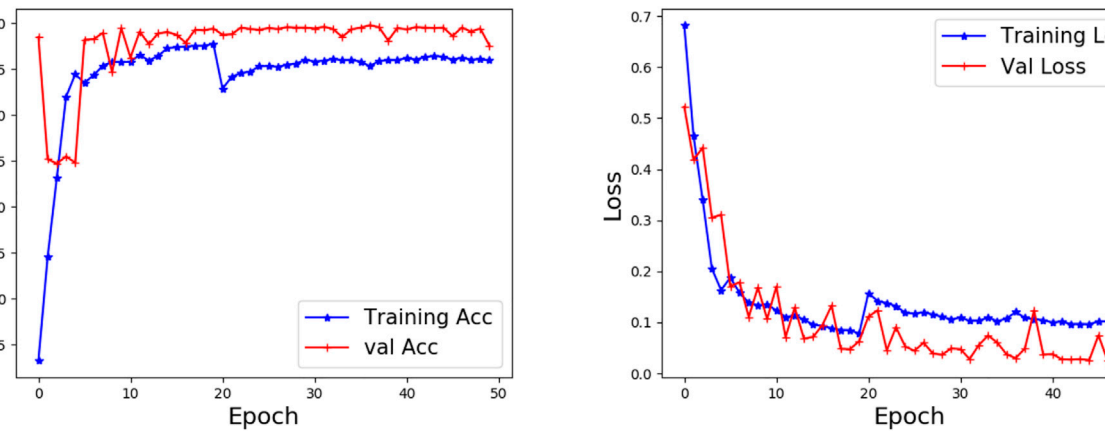

FIGURE 9 | Loss and accuracy performance of the initial model on CICIDS 2017 datasets.

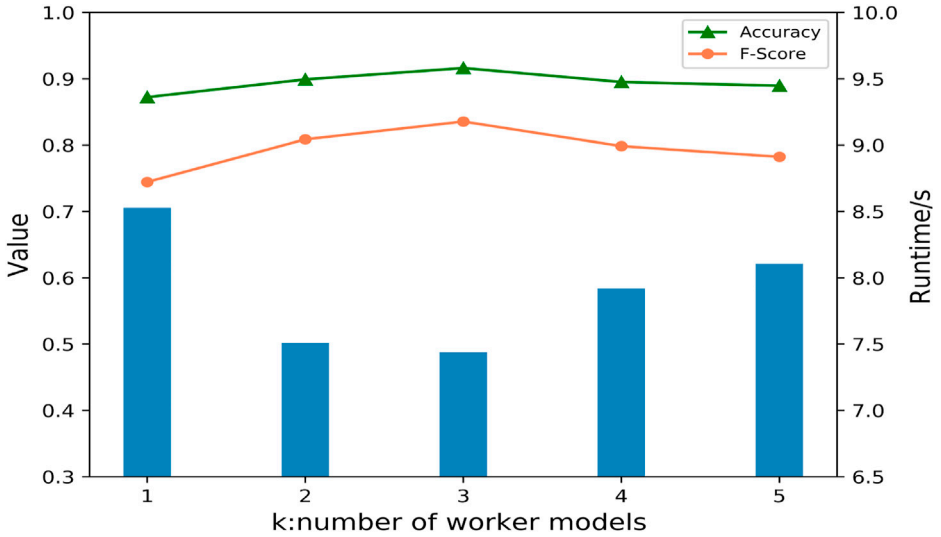

FIGURE 10 | Scalability of XFinder in distributed machine learning.

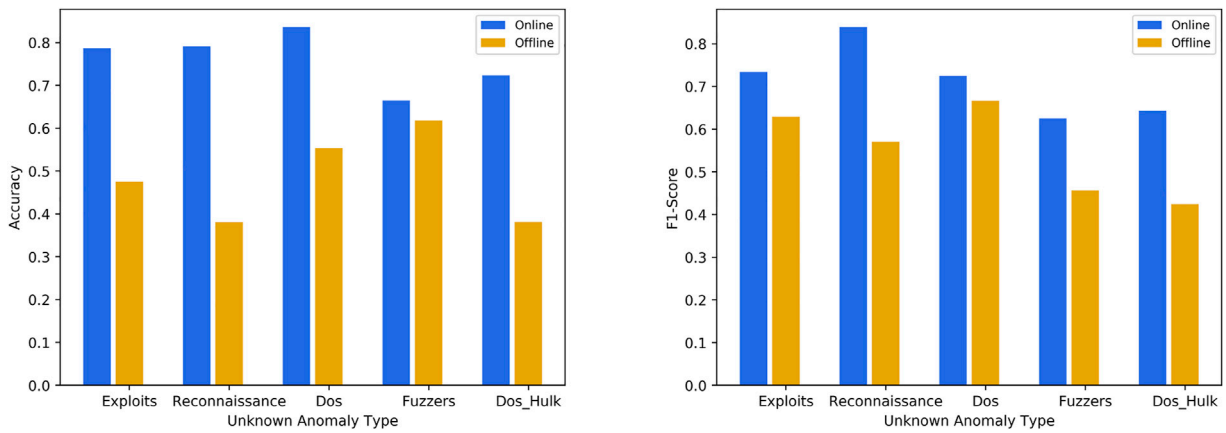

FIGURE 11 | The *Accuracy* and *F1—Score* performance of unknowns detection in the online mode and the offline mode.

3.4.1 Online Mode vs. Offline Mode

We compare the performance of unknown anomaly detection between the online mode and the offline mode. The online mode is implemented by our proposed approach, XFinder. The offline mode divides the data into an 80% training set and a 20% test set, the training set is used to train the model, and the test set is used to evaluate the performance of the model.

Figure 11 shows the accuracy and the f1-score of unknown anomaly detection in the online mode and the offline mode. We can see that both accuracy and f1-score performance are better in the online mode than in the offline mode, the average accuracy of the unknown anomaly detection of the model is increased by 27%, and the average F1-score is increased by 20%. This is our model XFinder that leverages the prior knowledge to separate the known and unknown; the online mode realizes self-learning, which can transfer the unknown anomaly to the knowledge base.

3.4.2 Multiple Types of Unknowns Detection

To test XFinder's ability to distinguish different unknown types of anomalies, we evaluate the model's ability to identify anomalies at different increments based on five types of unknown anomalies, including Exploits, Reconnaissance, Dos, Fuzzers, and Dos_Hulk.

Figure 12 shows the average detection accuracy and the f1-score of different unknown anomaly classes with different increments, respectively. The detection accuracy and the f1-score increase with the increase in the number of increments in general. This is because, with sufficient incremental training samples, the model can effectively learn the features of unknown anomalies for the next

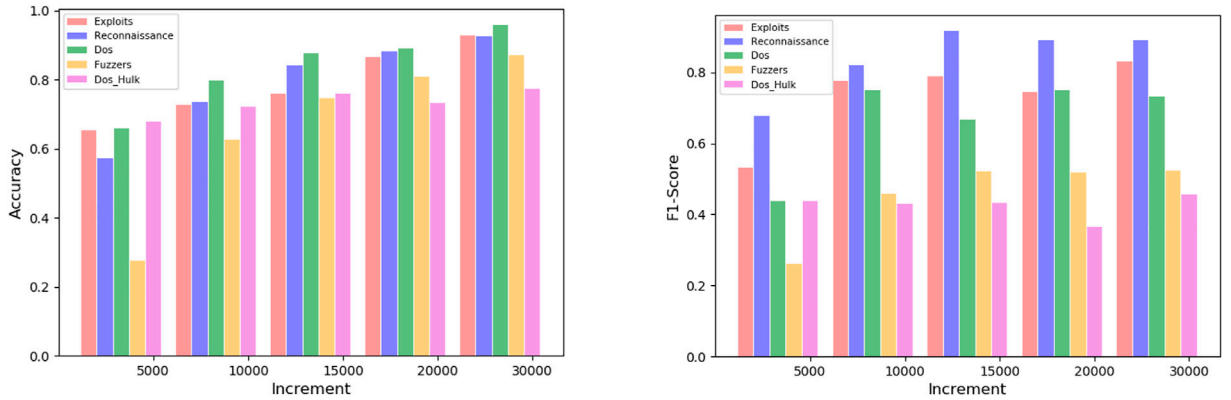

FIGURE 12 | The *Accuracy* and *F1—Score* performance of multi-class unknown anomalies in different increments.

TABLE 2 | Buffer effectiveness.

Unknown Anomaly	Accuracy		F1-score	
	No Buffer	Buffer	No Buffer	Buffer
Dos_Hulk	0.3817	0.7267	0.5849	0.6434
Exploits	0.4755	0.7865	0.3222	0.6290
Reconnaissance	0.3811	0.7914	0.2759	0.5706
Dos	0.5536	0.8361	0.5428	0.7251
Fuzzers	0.6183	0.6650	0.4277	0.6253
Average	0.4820	0.7612	0.4307	0.6387

unknown anomaly detection. XFinder merges the unknown anomaly to the knowledge base, which helps distinguish different unknown anomaly types, and it also proves the feasibility of using incremental learning to detect unknown anomalies.

3.4.3 Buffer Effectiveness

Since the training model requires time and is usually not suited for the online mode, we design a buffer to store the information of unknown anomalies detected in the current state for adapting to the online mode and distinguishing anomaly types.

Table 2 shows the comparison of detection performance for five unknown anomalies with or without buffer. **Figure 13** shows the comparative performance of Exploits unknown anomaly with or without buffer. We can see that the use of buffer can prevent the model from overfitting in the later stages of the iteration. The average accuracy of the unknown anomaly detection of the model is increased by 27%, and the average F1-score is increased by 20%. This is because the buffer avoids the situation that unknown anomalies cannot be effectively detected due to the failure of the model to be updated in time.

Therefore, the design of buffer realizes the rapid and accurate detection of unknown anomalies based on the online mode.

3.5 Runtime Evaluation

Table 3 show the time required for the XFinder model to detect various types of unknown anomalies in the KDD CUP 99, UNSW-NB 15, and CICIDS 2017 datasets in online and offline modes, respectively. We can see that the time in the online mode is significantly less than offline mode, and the

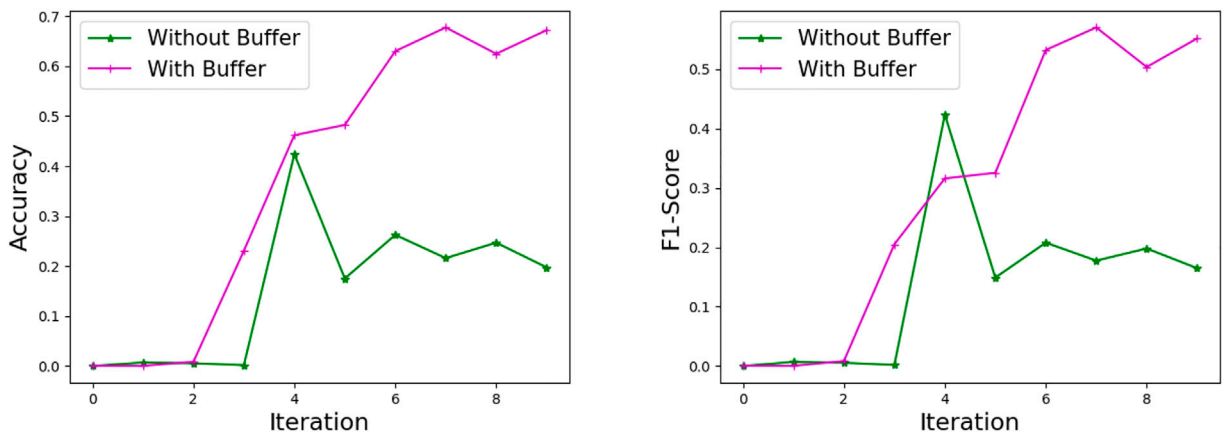

FIGURE 13 | The *Accuracy* and *F1—Score* performance of multi-class unknown anomalies in different increments.

TABLE 3 | Runtime evaluation.

Datasets	Unknown anomaly	Runtime (s)	Total time (s)	Offline time (s)
UNSW-NB 15	Exploits	10.5096	68.2657	114.7896
	Reconnaissance	6.9923		
	Dos	6.7311		
	Analysis	10.5662		
	Backdoor	9.8389		
	Shellcode	10.2905		
	Worms	5.324		
	Fuzzers	8.0131		
KDD CUP 99	Probing	6.6398	23.0672	45.8108
	R2L	7.9814		
	U2R	8.446		
CICIDS 2017	DoS_Slowhttptest	92.3941	319.8999	357.6348
	Dos_Hulk	139.6058		
	Dos_Slowloris	87.9		

online mode nearly halved the time compared to the offline mode.

This is because the offline mode needs to train the model first and then use the trained model to detect, while the online mode is training and testing simultaneously. Thus, XFinder realizes the near real-time update during the detection and meets the needs of the network.

4 RELATED WORK

In this study, distributed machine learning, the open-world approaches, and transfer learning for unknown anomaly detection in federated learning are discussed.

4.1 Distributed Machine Learning

Distributed machine learning (Galakatos et al., 2018) uses multi-node machine learning algorithms and systems, which are designed to improve performance, accuracy, and scale to larger input data sizes. With the development of big data, a single computer can no longer store and process machine learning tasks with large-scale data. In order to solve the above problems, distributed machine learning with data parallelism (Peteiro-Barral and Guijarro-Berdiñas, 2013) came into being. Nguyen et al. (2019) proposed an autonomous self-learning distributed system for detecting damaged Internet of Things devices. The system combines anomaly detection with federated learning and applies it to the anomaly detection field of the Internet of Things, which proved the excellent effect of distributed machine learning in anomaly detection. Sparks et al. (2013) designed an application programming interface to address the challenges of building machine learning algorithms in a distributed setting based on data-centric computing. Lin et al. (2020) proposed an ensemble distillation method for model fusion, which trained the central classifier through the unlabeled data on the outputs of the models from the clients. The use of blockchain technology (Gamage et al., 2020) has been proved to be effective to ensure the integrity of data transactions among entities. Haro-Olmo et al. (2021) seized the advantage of this circumstance to design a robust mechanism based on smart contracts and blockchain technology that allow the reliable processing of data. However, these anomaly detection methods lack the capability to detect, differentiate, and integrate unknown anomalies.

4.2 Open-World Approaches

Open-world learning aims to recognize the classes the learner has seen/learned previously and detect new classes that the learner has never seen before. The approaches for open-world classification can be roughly divided into two branches: traditional methods [e.g., SVM (Scheirer et al., 2014; Jain et al., 2014; Fei et al., 2016), sparse representation (Zhang and Patel, 2016), and nearest neighbor (Júnior et al., 2017)] and deep learning-based methods (Bendale and Boult, 2016; Ge et al., 2017; Shu et al., 2017).

In traditional methods, Scheirer et al. (2014) and Jain et al. (2014) both proposed leveraging extreme value models on the SVM decision scores to extend the SVM-based classification in an open-set setting. Fei et al. (2016) proposed a center-based similarity (CBS) learning strategy and build one-vs-rest CBS classifiers using SVM. Fei et al. (2016) then extended their study by adding the ability of incrementally or cumulatively learning new classes. Júnior et al. (2017) proposed a nearest neighbor-based classification approach based on the similarity scores, which is calculated by using the ratio of distances between the nearest neighbors. This approach identified any test sample with low similarity as unknown.

Considering that deep learning has achieved state-of-the-art performance in a wide range of visual understanding tasks, many deep learning-based open-world approaches have been studied. Zhang et al. (2020) investigated how to apply the extreme value theory (EVT) to unknown network anomaly detection systems and proposed a network intrusion detection method based on open set recognition. Bendale and Boult (2016) proposed the OpenMax function to replace the softmax function in CNNs. In this approach, the softmax probability distribution was redistributed to obtain the class probability of unknown

samples. Ge et al. (2017) introduced the G-OpenMax algorithm, which combined OpenMax with data augmentation using GANs. Instead of using softmax as the final output layer, Shu et al. (2017) proposed the deep open classifier (DOC) model to build a multi-class classifier with a one-vs-rest final layer that contained a sigmoid function for each seen class to reduce the open space risk. Recently (Shu et al., 2020), proposed a prototype-based open deep network (P-ODN) for open-set recognition, which proved the importance of more discriminable centers (or prototypes) for open-set recognition tasks.

However, these open-world approaches are not online modes and cannot meet the requirements of the network for the fast detection of anomalies.

4.3 Transfer Learning for Unknown Anomaly Detection

The purpose of transfer learning (Pan and Yang, 2009) is to use the knowledge with sufficient labeled data in the source domain to help build more accurate models in a related but different domain, which has only a few or no labeled data. Transfer learning approaches can be mainly categorized into three classes: instance-based (Gou et al., 2009), model-based (Gao et al., 2008), and feature-based (Zhao et al., 2019).

The study by Gou et al. (2009) applied an instance-based transfer learning approach to network intrusion detection. However, they required a large amount of labeled data from the target domain. Gao et al. (2008) proposed a model-based transfer learning approach and applied it to the KDD CUP 99 network dataset. Both of these instances and model-based transfer learning approaches rely on the assumption of homogeneous features to a great extent. Zhao et al. (2019) first applied a feature-based transfer learning method to improve the robustness of network anomaly detection. Wu et al. (2019) proposed a ConvNet model for network intrusion detection using transfer learning. Zhao et al. (2017) proposed a transfer learning-enabled network attack detection framework, which can enhance the detection of new network attacks in the target domain.

Although the transfer learning approach has achieved good performance in network anomaly detection, it cannot differentiate different anomaly types.

5 CONCLUSION

It is a significant task to detect unknown network anomalies in distributed machine learning scenario. In response to the challenges of unknown anomalies that are difficult to identify and contain multi-class anomalies in distributed machine learning, this study proposes XFinder, a dynamic network anomaly detection framework based on incremental learning. XFinder adopts an online mode to detect unknown anomalies and has three components: an unknown's detector, a buffer, and a network updater. The unknowns detector leverages the limited labeled data as prior knowledge to separate the known and unknown anomalies. The network updater merges the unknown anomalies to the knowledge base to help differentiate the unknown anomaly types, and the buffer is designed to adapt to the online mode and help to differentiate anomaly types.

We evaluated the performance of XFinder on three public datasets, the average accuracy of the unknown anomaly detection of XFinder increased by 27%, and the average F1-score increased by 20%. Compared with the offline mode, XFinder's detection time decreased by an average of approximately 33% on the three datasets, which can better meet the network requirements. Moreover, we verify the feasibility of unknown traffic anomaly detection in distributed machine learning through experiments.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material; further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

HD: conceptualization, methodology, software, and writing. SW: data curation, writing—original draft preparation, software, and validation. HH: writing—reviewing and editing.

REFERENCES

Bendale, A., and Boult, T. E. (2016). "Towards Open Set Deep Networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, June 27–30 2016, 1563–1572. doi:10.1109/cvpr.2016.173

Chouhan, N., Khan, A., and Khan, H.-u. -R. (2019). Network Anomaly Detection Using Channel Boosted and Residual Learning Based Deep Convolutional Neural Network. *Appl. Soft Comput.* 83, 105612. doi:10.1016/j.asoc.2019.105612

Fei, G., Wang, S., and Liu, B. (2016). "Learning Cumulatively to Become More Knowledgeable," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, August 13–17, 2016, 1565–1574. doi:10.1145/2939672.2939835

Galakatos, A., Crotty, A., and Kraska, T. (2018). *Distributed Machine Learning*. New York, NY: Springer New York, 1196–1201. doi:10.1007/978-1-4614-8265-9_80647

Gamage, H., Weerasinghe, H., and Dias, N. (2020). A Survey on Blockchain Technology Concepts, Applications, and Issues. *SN Comp. Sci.* 1, 1–15. doi:10.1007/s42979-020-00123-0

Gao, J., Fan, W., Jiang, J., and Han, J. (2008). "Knowledge Transfer via Multiple Model Local Structure Mapping," in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, Las Vegas, NV, August 24–27, 283–291. doi:10.1145/1401890.1401928

Ge, Z., Demyanov, S., Chen, Z., and Garnavi, R. (2017). *Generative Openmax for Multi-Class Open Set Classification*. arXiv [Preprint]. Available at: https://arxiv.org/abs/1707.07418

Gou, S., Wang, Y., Jiao, L., Feng, J., and Yao, Y. (2009). "Distributed Transfer Network Learning Based Intrusion Detection," in 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE), Rome, Italy, May 23–29, 2009, 511–515. doi:10.1109/ispa.2009.92

Haro-Olmo, F. J, Alvarez-Bermejo, J. A., Varela-Vaca, A. J., and López-Ramos, J. A. (2021). Blockchain-Based Federation of Wireless Sensor Nodes. *J. Supercomput.* 77 (7), 7879–7891.

Jain, L. P., Scheirer, W. J., and Boult, T. E. (2014). "Multi-class Open Set Recognition Using Probability of Inclusion," in *European Conference on Computer Vision* (Zurich, Switzerland: Springer), 393–409. doi:10.1007/978-3-319-10578-9_26

Kingma, D. P., and Ba, J. L. (2015). "Adam: A Method for Stochastic Gradient Descent," in ICLR: International Conference on Learning Representations, San Diego, CA, May 7–9, 2015.

Lars Arge, L. (2003). The Buffer Tree: A Technique for Designing Batched External Data Structures. *Algorithmica* 37, 1–24. doi:10.1007/s00453-003-1021-x

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. (2020). "Ensemble Distillation for Robust Model Fusion in Federated Learning," in *Advances in Neural Information Processing Systems*. Editors H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Vancouver, Canada: Curran Associates, Inc.), 2351–2363.

Mendes Júnior, P. R., De Souza, R. M., Werneck, R. d. O., Stein, B. V., Pazinato, D. V., de Almeida, W. R., et al. (2017). Nearest Neighbors Distance Ratio Open-Set Classifier. *Mach Learn.* 106, 359–386. doi:10.1007/s10994-016-5610-8

Moustafa, N., and Slay, J. (2015). "Unsw-nb15: a Comprehensive Data Set for Network Intrusion Detection Systems (Unsw-nb15 Network Data Set)," in 2015 military communications and information systems conference (MilCIS), Canberra, Australia, November 10–12, 2015 (IEEE), 1–6. doi:10.1109/milcis.2015.7348942

Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., and Sadeghi, A.-R. (2019). "DÏoT: A Federated Self-Learning Anomaly Detection System for IoT," in 2019 IEEE 39th International Conference on Distributed Computing Systems, Richardson, TX, July 7–9, 2019 (ICDCS), 756–767. doi:10.1109/ICDCS.2019.00080

Niu, J., Zhang, Y., Liu, D., Guo, D., and Teng, Y. (2019). "Abnormal Network Traffic Detection Based on Transfer Component Analysis," in 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, May 20–24, 2019 (IEEE), 1–6. doi:10.1109/iccw.2019.8756996

Pan, S. J., and Yang, Q. (2009). A Survey on Transfer Learning. *IEEE Trans. knowledge Data Eng.* 22, 1345–1359. doi:10.1109/TKDE.2009.191

Peteiro-Barral, D., and Guijarro-Berdiñas, B. (2013). A Survey of Methods for Distributed Machine Learning. *Prog. Artif. Intell.* 2, 1–11. doi:10.1007/s13748-012-0035-5

Sabeel, U., Heydari, S. S., Mohanka, H., Bendhaou, Y., Elgazzar, K., and El-Khatib, K. (2019). "Evaluation of Deep Learning in Detecting Unknown Network Attacks," in 2019 International Conference on Smart Applications, Communications and Networking (SmartNets), Sharm El Sheik, Egypt, December 17–19, 2019 (IEEE), 1–6. doi:10.1109/smartnets48225.2019.9069788

Scheirer, W. J., Jain, L. P., and Boult, T. E. (2014). Probability Models for Open Set Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 2317–2324. doi:10.1109/tpami.2014.2321392

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in Proceedings of the 4th International Conference on Information Systems Security and Privacy(ICISSP), Funchal - Madeira, Portugal, January 22–24, 2018, 108–116. doi:10.5220/0006639801080116

Shu, L., Xu, H., and Liu, B. (2017). *Doc: Deep Open Classification of Text Documents*. arXiv [Preprint]. Available at: https://arxiv.org/abs/1709.08716

Shu, Y., Shi, Y., Wang, Y., Huang, T., and Tian, Y. (2020). p-Odn: Prototype-Based Open Deep Network for Open Set Recognition. *Scientific Rep.* 10, 1–13. doi:10.1038/s41598-020-63649-6

Shu, Y., Shi, Y., Wang, Y., Zou, Y., Yuan, Q., and Tian, Y. (2018). "Odn: Opening the Deep Network for Open-Set Action Recognition," in 2018 IEEE International Conference on Multimedia and Expo (ICME), San Diego, CA, July 23–27, 2018 (IEEE), 1–6. doi:10.1109/icme.2018.8486601

Sparks, E. R., Talwalkar, A., Smith, V., Kottalam, J., Pan, X., Gonzalez, J., et al. (2013). "MLI: An API for Distributed Machine Learning," in 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, December 7–10 2013. doi:10.1109/ICDM.2013.158

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). "A Detailed Analysis of the Kdd Cup 99 Data Set," in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, Canada, July 8–10, 2009 (IEEE), 1–6. doi:10.1109/cisda.2009.5356528

Tveit, A., and Hetland, M. L. (2003). "Multicategory Incremental Proximal Support Vector Classifiers," in International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, Oxford, UK, September 3–5, 2003 (Springer), 386–392. doi:10.1007/978-3-540-45224-9_54

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., and Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access* 7, 41525–41550. doi:10.1109/access.2019.2895334

Wu, P., Guo, H., and Buckland, R. (2019). "A Transfer Learning Approach for Network Intrusion Detection," in 2019 IEEE 4th International Conference on Big Data Analytics (ICBDA), Pulau Pinang, Malaysia, November 19–21, 2019. (IEEE), 281–285. doi:10.1109/icbda.2019.8713213

Yang, C. (2019). Anomaly Network Traffic Detection Algorithm Based on Information Entropy Measurement under the Cloud Computing Environment. *Cluster Comput.* 22, 8309–8317. doi:10.1007/s10586-018-1755-5

Yang, H.-M., Zhang, X.-Y., Yin, F., and Liu, C.-L. (2018). "Robust Classification with Convolutional Prototype Learning," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, June 18–23 2018, 3474–3482. doi:10.1109/cvpr.2018.00366

Yeh, T., and Darrell, T. (2008). "Dynamic Visual Category Learning," in 2008 IEEE Conference on Computer Vision and Pattern Recognition, Alaska, June 23–28, 2008 (IEEE), 1–8. doi:10.1109/cvpr.2008.4587616

Zhang, H., and Patel, V. M. (2016). Sparse Representation-Based Open Set Recognition. *IEEE Trans. Pattern Anal. Mach Intell.* 39, 1690–1696. doi:10.1109/TPAMI.2016.2613924

Zhang, Y., Niu, J., Guo, D., Teng, Y., and Bao, X. (2020). Unknown Network Attack Detection Based on Open Set Recognition. *Proced. Comp. Sci.* 174, 387–392. doi:10.1016/j.procs.2020.06.104

Zhao, J., Shetty, S., and Pan, J. W. (2017). "Feature-based Transfer Learning for Network Security," in MILCOM-2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, October 23–25, 2017 (IEEE), 17–22. doi:10.1109/milcom.2017.8170749

Zhao, J., Shetty, S., Pan, J. W., Kamhoua, C., and Kwiat, K. (2019). Transfer Learning for Detecting Unknown Network Attacks. *EURASIP J. Info. Security.* 2019, 1–13. doi:10.1186/s13635-019-0084-4

Conflict of Interest: SW was employed by the company Jiangsu Huaneng Smart Energy Supply Chain Technology Co., Ltd., China.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors, and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.