Collaborative Anomaly Detection for Internet of Things based on Federated Learning

Seongwoo Kim*, He Cai*, Cunqing Hua*, Pengwenlong Gu*,Wenchao Xu†, Jeonghyeok Park‡
*School of Cyber Science and Engineering, Shanghai Jiao Tong University, China
†School of Computing and Information Sciences, Caritas Institute of Higher Education, Hong Kong
‡School of Computer Science and Engineering, Shanghai Jiao Tong University, China

***Abstract*—In this paper, we propose a federated learning(FL)-based collaborative anomaly detection system. This system consists of multiple edge nodes and a server node. The edge nodes are in charge of not only monitoring and collecting data, but also to train an anomaly detection neural network classification model based on the local data. On the other hand, the server aggregates the parameters from the edges and generates a new model for the next round. This system structure achieves light weight transmission between the server and the edge nodes, and user privacy can be well protected since raw data are not communicated directly. We implement the proposed scheme in the practical system and present experimental results that demonstrate results competitive with those of state-of-the-art models.**

***Index Terms*—Internet of Things, Anomaly detection, Federated learning, Machine learning**

I. Introduction

Due to the rapid development of communication technologies, smart technology devices have become increasingly prevalent. Currently, Internet of Things (IoT) is recognized as the one of the most prominent topics. The IoT environment consists of interconnections between heterogeneous network devices. According to results in several surveys, there were 8.4 billion IoT devices in the world in the year 2017 [1]. Moreover, it is estimated that this will soar to 30 billion devices by 2020 [2]. Furthermore, the number of IoT devices by 2025 is expected to be more than 75 billion [3]. Therefore, we can clearly recognize that the IoT market is rapidly expanding.

While these new technologies are enormously convenient, they give rise to inescapable risks and challenges. Particularly, promptly detecting cyber attacks is becoming one of the most important issues for IoT. In practice, cyber attacks, such as malware, botnet [4], and insider attack [5], among others, occur rarely, and typically these anomalies consume very few system resources, so it is very difficult to detect and trace them. Nevertheless, these cyber attacks generally occur in the data and features of network transmission, and are potentially recognizable as data anomalies, which is the basic premise for detecting anomalies.

Anomaly detection system in the IoT networks identifies misbehavior, such as faults and frauds from the massive amount of interactions between the devices. Therefore, the challenges include the enormous complexity of the interconnected devices. Conventional anomaly detection systems typically depend on data mining [6], statistics [7], machine learning, or information theory. Recently, the machine learning-based technique have gained much attention and popularity because it can learn the features of data sets autonomously and make relatively accurate classifications and predictions using different models. These methods in general fall into three categories: supervised (e.g., KNN [8], decision trees [9]), unsupervised (e.g. clustering [10]), and semi-supervised learning (e.g., semi-supervised SVM [11], semi-supervised Deep Learning [12]).

Most anomaly detection systems rely on a centralized management method to collect and process data generated from IoT devices or end users. With the explosion of big data generated by the increasingly prevalent smart devices, these conventional schemes consume high network bandwidth and incur long transmission latency, and they also face an issue with privacy since raw data has to be communicated between the edge devices and the server. Some efforts have been made to address these issues. For example, state-of-art deep learning based anomaly detection systems in IoT are presented by specific techniques; [13] adopt autoencoders (AE) to present anomaly detection scheme in wireless sensor networks (WSNs) under IoT environment, and [14] propose an unsupervised anomaly detection algorithm utilizing a CNN backbone and a restricted boltzmann machine (RBM)-based system. In [15], a collaborative anomaly detection system is proposed based on the active learning framework, whereby the edge node only selects the most valuable data to transmit to the server using uncertainty sampling strategies, which can reduce the bandwidth demand and transmission latency. In [16], the cyber attack detection model for an edge network empowered by FL is proposed. Each participant sends its trained model to the server after local training, eventually achieving enhanced data privacy and reducing traffic load for the whole network.

In this paper, we propose a FL-based anomaly detection system. In this system, each edge node trains an anomaly detection neural network classification model based on local data, and then the model parameters are reported to the central server. By aggregating the parameters from multiple edges, the server generates a new model and distributes this to the edge nodes for training the local model in the next round. In this way, the data transmission between the server and the edge nodes can be significantly reduced, and user privacy can be well-preserved since raw data communication between the edge nodes and the server is not necessary. We implement the proposed scheme in a practical system, and experimental results are provided to demonstrate the performance of the proposed scheme which performs better than non-FL's.

The rest of this paper is organized as follows. In Section II, we introduce the system model considered in this paper, and discuss the details of the training of classification model at the edge nodes, as well as the global model aggregation and update at the server. We present experimental results in Section III to evaluate the performance of the proposed scheme, and we draw conclusions in Section IV.

II. Design of the Anomaly Detection System

The anomaly detection systems we propose are based on the federated learning framework. As shown in Fig. 1, in this system, each edge node is in charge of collecting and pre-processing the local data for training the anomaly classification model. After conducting training of the model based on the local data, the parameters of the model are reported to the cloud server, whereby a new model can be generated based on the aggregated parameters, which are then fed back to the edge nodes to continue the training for the next round. This process is repeated until the desired classification accuracy is achieved or the result converges.

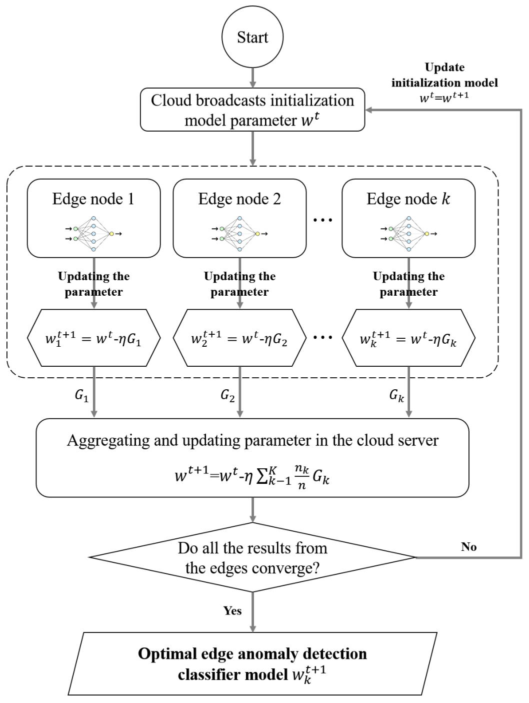
Using this FL framework, the edge nodes and the cloud server interact with each other with the parameter instead of the raw data. The virtues of the system can be summarized as follows: [17]

- User privacy: it is more advantageous for evasion of privacy security leakage based on the fact that the raw data of users need not be sent to the cloud; the parameters of the model are transmitted instead.
- Network bandwidth efficiency: participating devices only send the updated model parameters instead of the raw data, the cost of data communication can be significantly reduced, and this system can mitigate the burden on the networks.
- Low latency: the communication latency is much lower than that of systems where decisions are made in the cloud, which is desirable for certain time-critical applications [18].

In the following, we discuss the details for the pre-processing and training of the local data at the edge nodes, as well as the aggregation of parameters and update of global model at the cloud server.

A. Data pre-processing

For vectorizing the raw data for training, pre-processing is a compulsory stage. As shown in Fig.2, it is divided into three steps; namely, feature selection, feature extraction, and dimensional reduction. For the feature selection and



Start
Update initialization model w^t=w^{t+1}
Cloud broadcasts initialization model parameter w^t
Edge node 1
Edge node 2
Edge node k
Updating the parameter
Updating the parameter
Updating the parameter
$w_1^{t+1} = w^t$-ηG_1
$w_2^{t+1} = w^t$-ηG_2
$w_k^{t+1} = w^t$-ηG_k
G_1
G_2
G_k
Aggregating and updating parameter in the cloud server
w^{t+1}=w^t-$\eta \sum_{k=1}^{K} \frac{n_k}{n} G_k$
Do all the results from the edges converge?
No
Yes
Optimal edge anomaly detection classifier model w_k^{t+1}

Fig. 1. Architecture of the collaborative anomaly detection system based on the FL framework.

extraction, they have a crucial impact on the detection result, processing the redundancy elimination of the original data and the text normalization is organized to obtain the formatted data set. For the sensitive data within the threshold time limit, the data is statistically modeled to obtain its operating characteristics and forms, and to classify and locate abnormal information. The one-hot sequence conversion is used to convert text word data to binary representations: valid indices are set to 1, otherwise 0. Then, the local data optimization is conducted through data normalization using the Z-score method as follows:

$$Z_x = \frac{X_i - \bar{X}}{\bar{S_x}}, \tag{1}$$

where $\bar{X}$ and $\bar{S_x}$ are the mean value and standard deviation of the original data set.

Finally, the principal component analysis (PCA) method [19] is adopted to reduce the dimensionality of the data. Using the PCA algorithm, the redundant information can be removed to reduce the weight of data while retaining important information in the data set, thereby significantly reducing the computational cost in the following training procedure.

B. Local model training at the edge node

Initially, the cloud server decides the basic parameter settings such as learning rate, hyper-parameter of the global model, etc. Then the server transmits the initial global model parameters to every edge node. Afterward, each edge node organizes pre-processing as the preliminary

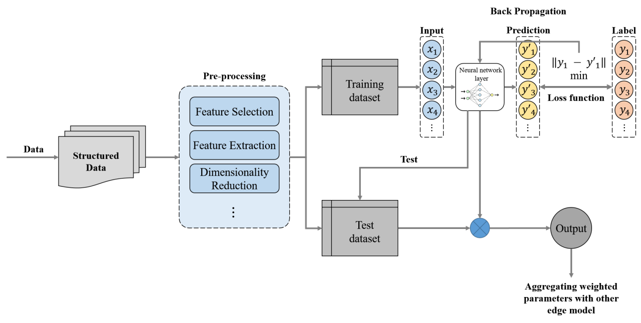


Fig. 2. Workflow at the edge node.

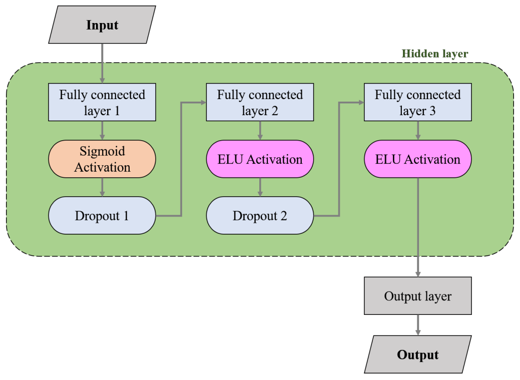


Fig. 3. Anomaly detection classifier based on multi-layer perceptron on edges

work for local training using the neural network, which has excellent feature extraction and classification performance. Furthermore, the weights and bias parameters in the neural network can directly characterize the network structure and model values. In specific terms for the neural network, a multi-layer perceptron is chosen as the anomaly detection classifier model at the edge node. As shown in Fig. 3, our multi-layer perceptron structure consists of an input layer, three hidden layers, and one output layer. The input layer inserts corresponding samples ($x_1, x_2, .., x_n$ in X). The hidden layers are connected with each *fully connected layer* to ensure effective feature extraction and classification. The hidden layers output $f(WX+b)$, where W is the corresponding weight, b is the offset, and f is activation function. Moreover, the first two hidden layers are followed by two dropout layers to cut off some neurons, which can address the overfitting problem.

The *sigmoid* [20] function is used in the first hidden layer to avoid the gradient explosion problem, and the *ELU* [21] function is used in the second and last layer, which is beneficial for retaining the data features more completely. The result of each activation function is passed to the next layer. In the output layer, the prediction of normal/abnormal labels is conducted based on the results of the hidden layers, whereby the mapping function of the output layer maps the result to the binary classification single output result space. The *RMSProp* [22] is used as an optimization method, and *Binary cross entropy* is used as a loss function in the output layer. After the training of the local model, the edge node sends the parameters to the server.

C. Global model aggregation and update

Assuming there are n total samples, (x_i, y_i) is the ith sample, and w is the model parameter. Then the loss function for the sample is $f_i(w) = l(x_i; y_i; w)$, and the overall loss function for all samples is given by:

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w). \tag{2}$$

Suppose there are K edges in the system, the sample set of the kth edge is L_k, and n_k denotes the number of samples in the kth edge. Thus the average loss function for the kth edge is given by:

$$h_k(w) = \frac{1}{n_k}\sum_{i\in L_k} f_i(w) \tag{3}$$

Then the overall loss function for all edge nodes is as follows:

$$f(w) = \sum_{i=1}^{K}\frac{n_k}{n}h_k(w) = \frac{1}{n}\sum_{i=1}^{K}\sum_{i\in L_k} f_i(w) \tag{4}$$

In each round t, each edge initially updates its local model and compute the average gradient at the edge as follows:

$$G_k^t = \bigtriangledown h_k(w_t). \tag{5}$$

Then the model parameter of the kth edge is updated using the gradient algorithm as follows:

$$w_k^{t+1} = w_k^t - \eta G_k^t, \tag{6}$$

where η is the learning rate. This updated parameter is reported to the server.

After receiving the parameters from edge nodes, the cloud server aggregates the parameters using the weighted average aggregation (FedAvg) [23] as follows:

$$W_{t+1} = \sum_{k=1}^{K}\frac{n_k}{n}w_k^{t+1} \tag{7}$$

In this way, the cloud server builds a new global model for the next round based on the aggregated parameters, which is fed back to all edge nodes.

This process is repeated a certain model convergence criteria is satisfied, which is given by:

$$\frac{train\_loss_{t+1} - train\_loss_t}{train\_loss_t} < \delta, \tag{8}$$

where δ is a prescribed threshold, and $train\_loss_t$ is given by:

$$train\_loss_t = \sum_{k=1}^{K}\frac{n_k}{n}G_k(t) \tag{9}$$

To prevent the case that the model is not converged, the number of limit round should be opted. The opted number of the maximum rounds will be mentioned particularly in Table I

In cloud-edge interactions, the convergence of the cloud model is used as the standard to define the system model convergence. For a given edge, it obtains the initial updated model from the cloud at the start of each round of training. It then updates the model and transmits these updates to the cloud. If the cloud model converges, and the cloud anomaly detection classification will be redistributed to each edge as the final model.

III. Experimental Results

A. Experiment Setup

Our experiment is based on the **NSL-KDD data set**, which is used to train and test the models. It contains a set of internet records with 41 features and 22 types of attack labels, which consist of 67,343 normal data and 58,630 abnormal data, which contains 45,927 pieces of DDOS attack data, amounting to 125,973 total samples.

The parameters used in the experiment are summarized in Table I.

TABLE I
Experimental Setup Parameters

Parameter	**Value**
Number of Local Epochs	20-40
Batch Size	90-120
Number of Clients	4
Max Data set Size	10,000
Convergence Threshold (δ)	0.01
Max Round Limit	50
Data set Setting	6 : 1 : 3 (training : validation : test)

The *keras* library is used to implement the neural network based anomaly detection classifier in our work. The *socket* library is adopted as the software environment for achieving communication between the server and edge nodes.

1) Evaluation method: The evaluation method is based on 6 metrics; training accuracy, validation accuracy, test accuracy, training loss, validation loss, and test validation. The accuracy evaluation function is defined as follows:

$$accuracy = \frac{n_{correct}}{n}, \tag{10}$$

where the $n_{correct}$ is the number of correct samples in all samples n. The accuracy values can directly reflect the model's performance, and the loss values represent the effects of model training and testing.

2) Data Pre-processing: Considering the application in a realistic scenario, the data distribution on different edge nodes is supposed to be as diverse as possible. Since the probability of each sampling is random, the new data sets obtained after sampling have different distributions.

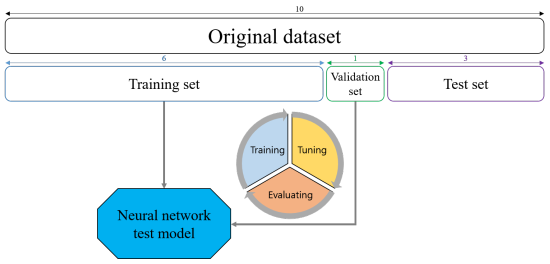


Fig. 4. Data set division in the proposed system.

Through the random sampling, non-independent and identically distributed (non-IID) data are obtained, which is closer to the actual system data state.

3) Data set division: To avoid underfitting and overfitting problems, the original data set is divided into three subsets; training set, validation set, and test set as the ratio of 6: 3: 1. This way can estimate the generalization error effectively. The performance of the model can be evaluated, as it can guarantee that the model performance satisfies the requirements. We then test the generalization performance and accuracy of the model on the test set (see Fig. 4).

In the training process, the training accuracy and loss on the verification set are used as references to adjust the hyper-parameters of the model to make the system model reach the optimal state.

After the model has converged, the module test is organized on each edge test data set and evaluates the effectiveness of the model by the aforementioned 6 factors. Afterward, the system uses different edge test sets to test each edge's model and evaluates the generalization ability of each local anomaly detection classification model after optimizing the model parameters.

B. Experiment results

In this section, we evaluate the effectiveness of the collaborative anomaly detection system based on the federated learning framework. As a comparison, we also consider the non-FL scheme whereby each edge only trains its classification model based on its local data. Thus, the interaction between cloud server and participant edges is eliminated from the proposed scheme, but each edge's local training model is totally same as proposed FL-based anomaly detection system. It will not aggregate the edge parameters in every round, the four edges works respectively themselves. Therefore, due to the comparison between FL and non-FL performance, the federated learning's effectiveness can be identified.

Before the comparison, we adopt several parameters to optimize the system settings. To find the best set of parameters, a numbers of epoch and batch sizes are tried to generate higher test accuracy and lower test loss. The experiments are organized based on 4 edge nodes and summarized in Table II. They start from small numbers

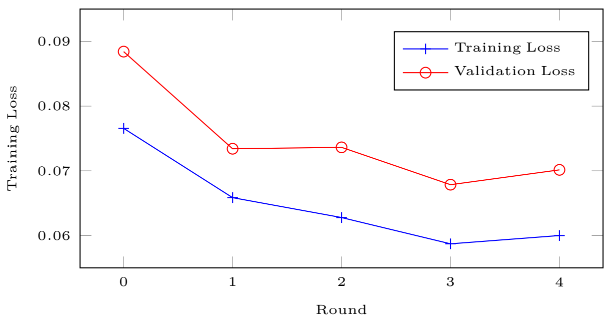


Fig. 5. The training loss of the cloud model.

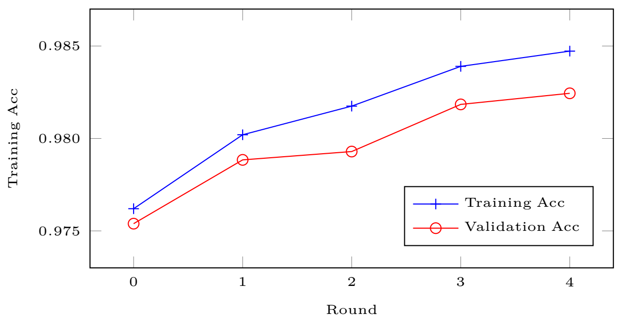


Fig. 6. The training accuracy of the cloud model.

of epochs and batches before finally achieving the highest test accuracy value (97.92%) and the relatively lowest test loss (8.43%) on 30 epoch size and 120 batch size.

TABLE II
PARAMETER ADOPTION EXPERIMENTAL RESULTS

# of data set	# of epochs	# of batches	% of test accuracy	% of test loss
10,000	20	90	97.28	8.76
10,000	30	90	97.14	9.04
10,000	20	100	97.02	9.78
10,000	30	100	97.02	9.42
10,000	20	120	96.68	10.15
10,000	**30**	**120**	**97.92**	**8.43**
10,000	40	120	97.11	9.55

Based on this parameter settings in Table II, the comparison experiment between the FL-based system and the non-FL system is conducted in the following section.

1) System model assessment: Fig 5 shows the training loss and validation loss based on an epoch size of 30 and a batch size of 120. The training loss and validation loss are calculated and recorded in each round. At each round, the loss of system training and validation decrease together and eventually converge on the fourth round. This trend demonstrates that the overall performance of the system is improving round by round. Consequently, the training and validation losses converge to 6.00% and 7.01%, respectively.

On the other hand, the training and the validation accuracy are shown in Fig 6. It can be seen that the training and validation accuracy keeps increasing, which suggests that the classification model is becoming enhanced. Consequently, the final training and validation accuracy reached respectively 98.47% and 98.24%. As summarized in Table II, a test accuracy of 97.92% and a test loss of 8.43% can be achieved.

C. Model comparison

We also compare the local model performance without interaction with the cloud. The final evaluation metrics include edge training accuracy and loss, edge validation accuracy and loss, and edge test accuracy and loss.

Based on the above-mentioned metrics, the performance comparison experiment between FL-based models and non-FL models is shown in Table III.

TABLE III
THE PERFORMANCE COMPARISON BETWEEN FL AND NON-FL MODELS

	Training acc Training loss	Valid acc Valid loss	Test acc Test loss
Edge1 (FL)	99.17 3.73	98.14 9.41	98.80 6.01
Edge2 (FL)	98.24 7.25	98.08 6.83	97.34 9.78
Edge3 (FL)	98.72 4.86	98.08 5.66	98.32 7.04
Edge4 (FL)	97.76 8.16	98.68 6.17	97.24 10.89
FL (aggregated)	**98.47 5.60**	**98.24 7.01**	**97.92 8.43**
Edge1 (non-FL)	97.13 7.35	96.76 8.14	96.74 8.53
Edge2 (non-FL)	97.64 6.32	97.42 8.28	96.92 7.93
Edge3 (non-FL)	98.26 5.54	98.50 5.40	97.70 6.93
Edge4 (non-FL)	98.59 4.85	97.66 8.33	97.72 7.27
non-FL (aggregated)	**97.91 6.02**	**97.59 7.54**	**97.27 7.67**

From the macroscopic point of view, the FL model achieves 98.47% training accuracy (acc), 98.24% validation acc, and 97.92% test acc. On the other hand, the non-FL model achieves 97.91% training acc, 97.59% validation acc, and 97.27% test acc. Moreover, FL's loss values are 5.60%, 7.01%, and 8.43%, and non-FL's are 6.02%, 7.54%, and 7.67%, respectively. Both FL and non-FL based schemes have similar trends in results, but the FL-based scheme performs better in accuracy, though its loss is slightly worse. In the non-FL, each edge builds its own model with only its own local data, Thus, the models should be more localized than the FL-based scheme. Therefore we assume that non-FL scheme performed better in loss data. On the contrary, in the FL-based scheme, each local model gets aggregated as a global model, which tends to better generalize compared to non-FL models, as it adopts the learnings of other edges, providing for achieve better accuracy than the non-FL scheme.

Consequently, as dynamical IoT circumstances possess many random instantiations of any of a number of parameters (users, edges, devices) that connect and disconnect at any time. Due to the unique characteristics of the IoT circumstance, even though FL-based model does not perform significantly better than non-FL model, it is still a viable solution as it offers a very generalized model to all edges.

IV. Conclusion

This paper proposes a federated learning-based anomaly detection system in IoT environment. We adopt a neural network to act as the anomaly classification model at the edges and the federated learning framework to communicate and aggregate model parameters at the server in order to build a generalized model within every edge area. This system allows us to reduce the bandwidth requirement and to improve transmission latency and user privacy protection. We conduct extensive experiments to evaluate the FL-based anomaly detection system and the non-FL anomaly detection system. Experimental results show that even though FL shows slightly worse loss performance due to less edge-localization, it achieves better accuracy by building applicable models across a wide variety of cases.

Acknowledgment

This work was supported by National Key Research and Development Program of China (2018YFC0309604, 2018YFB1700203) , Natural Science Foundation of Shanghai (NO. 19ZR1426400), and Natural Science Foundation of China (No. 61831007).

References

[1] R. Köhn, “Konzerne verbünden sich gegen hacker,” Feb 2018. [Online]. Available: https://www.faz.net/aktuell/wirtschaft/digitec/grosse-internationale-allianz-gegen-cyber-attacken-15451953.html

[2] A. Nordrum, “Popular internet of things forecast of 50 billion devices by 2020 is outdated (2016),” *Dosegljivo: https://spectrum. ieee. org/tech-talk/telecom/internet/popular-internet-ofthings-forecast-of-50-billion-devices-by-2020-is-outdated.[Dostopano: 11. 8. 2017]*, 2017.

[3] D. Lynkova, “Iot statistics and trends to know in 2020,” Oct 2019. [Online]. Available: https://leftronic.com/internet-of-things-statistics/

[4] E. Bertino and N. Islam, “Botnets and internet of things security,” *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[5] A. Y. Khan, R. Latif, S. Latif, S. Tahir, G. Batool, and T. Saba, “Malicious insider attack detection in iots using data analytics,” *IEEE Access*, vol. 8, pp. 11 743–11 753, 2019.

[6] C. Phua, V. Lee, K. Smith, and R. Gayler, “A comprehensive survey of data mining-based fraud detection research,” *arXiv preprint arXiv:1009.6119*, 2010.

[7] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Network anomaly detection: methods, systems and tools,” *Ieee communications surveys & tutorials*, vol. 16, no. 1, pp. 303–336, 2013.

[8] Z. H. Zhou, “Ml-knn: A lazy learning approach to multi-label learning,” *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[9] L. Rokach and O. Maimon, “Top-down induction of decision trees classifiers - a survey,” *IEEE Transactions on Systems Man & Cybernetics Part C*, vol. 35, no. 4, pp. 476–487, 2005.

[10] I. Syarif, A. Prugel-Bennett, and G. Wills, “Unsupervised clustering approach for network anomaly detection,” in *International conference on networked digital technologies*. Springer, 2012, pp. 135–145.

[11] A. Astorino, E. Gorgone, M. Gaudioso, and D. Pallaschke, “Data preprocessing in semi-supervised svm classification,” *Optimization*, vol. 60, no. 1-2, pp. 143–151, 2011.

[12] C. Baur, S. Albarqouni, and N. Navab, “Semi-supervised deep learning for fully convolutional networks,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2017, pp. 311–319.

[13] T. Luo and S. G. Nagarajan, “Distributed anomaly detection using autoencoder neural networks in wsn for iot,” in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[14] B. A. Mudassar, J. H. Ko, and S. Mukhopadhyay, “An unsupervised anomalous event detection framework with class aware source separation,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2671–2675.

[15] H. Cai, C. Hua, and W. Xu, “Design of active learning framework for collaborative anomaly detection,” in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2019, pp. 1–7.

[16] A. Abeshu and N. Chilamkurti, “Deep learning: The frontier for distributed attack detection in fog-to-things computing,” *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169–175, 2018.

[17] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, “Federated learning in mobile edge networks: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, 2020.

[18] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, “Real-time video analytics: The killer app for edge computing,” *computer*, vol. 50, no. 10, pp. 58–67, 2017.

[19] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, “A novel anomaly detection scheme based on principal component classifier,” MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING, Tech. Rep., 2003.

[20] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[21] Y. Li, C. Fan, Y. Li, Q. Wu, and Y. Ming, “Improving deep neural network with multiple parametric exponential linear units,” *Neurocomputing*, vol. 301, pp. 11–24, 2018.

[22] T. Kurbiel and S. Khaleghian, “Training of deep neural networks based on distance measures using rmsprop,” *arXiv preprint arXiv:1708.01911*, 2017.

[23] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, “Federated learning of deep networks using model averaging. corr abs/1602.05629 (2016),” *arXiv preprint arXiv:1602.05629*, 2016.