An Efficient Federated Learning Framework for IoT Intrusion Detection

Yushen Chen[1], Fang Fang[1,2], Boyu Wang[1], and Lan Zhang[3]
[1]Department of Computer Science, Western University, London, Canada
[2]Department of Electrical and Computer Engineering, Western University, London, Canada
[3]Department of Electrical and Computer Engineering, Clemson University, Clemson, USA
Emails: {yche2692, fang.fang}@uwo.ca, bwang@csd.uwo.ca, and lan7@clemson.edu

2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall) | 979-8-3315-1778-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/VTC2024-Fall63153.2024.10757488

***Abstract*—The exponential growth of the Internet of Things (IoT) ecosystems has raised significant cybersecurity concerns. Deep learning (DL)-based methods have shown promising performance in detecting potential cyber threats in IoT networks. However, as these methods often involve data centralization, they can pose serious data privacy issues for IoT users and increase the communication burden of local networks. Federated learning (FL), as a distributed learning paradigm, enables privacy-preserving training of IoT intrusion detection models by requiring only model updates from IoT devices. However, the resource-constrained nature of IoT devices can significantly decrease FL training efficiencies, such as increased training latency and delayed convergence speed. Moreover, the data heterogeneous issues of IoT devices can also impact the accuracy and robustness of the trained model. To address these challenges, we propose an efficient FL framework, FedKD-Prox, based on federated proximal (FedProx) and knowledge distillation (KD). To improve the prediction accuracy within a limited time budget, the proposed framework aims to efficiently exploit the computation capability of the IoT trainers, reduce the communication overhead of FL, and alleviate the impact of heterogeneous data issues. The simulation results show that FedKD-Prox achieves higher accuracy and improves the robustness of the trained intrusion detection model.**

***Index Terms*—Intrusion detection, Internet of Things, federated learning, knowledge distillation, federated proximal.**

I. Introduction

The rapid proliferation of Internet of Things (IoT) devices and its expanding ecosystem creates a broader surface for potential cyber threats. Deep learning (DL) approaches have increasingly gained attention in recent years as they provide intelligence and insights from the gathered data of the source nodes. The performance has been proven by various literature such as [1]–[3]. However, these methods may not be suitable for IoT networks. Due to the sensitive nature of the private data stored in IoT devices, a centralized training process could compromise data security and result in serious privacy concerns. Moreover, training DL models, especially for intrusion detection, often require a large amount of training data. Uploading these data could burden the communication resources of IoT devices.

Federated learning (FL) has been introduced to address data privacy issues and reduce communication costs. It allows the training devices to keep their data private and communicate through model updates. Existing research has applied FL in developing intrusion detection systems to various network applications, such as slicing network [4], vehicle-to-vehicle network [5], unmanned aerial vehicles network [6], and energy harvesting [7]. Despite these promising solutions, the IoT devices still pose challenges to FL due to their resource-constrained and distributed nature, including increased training time and data heterogeneity. However, none of the above works thoroughly considered them.

Extensive research has been conducted to tackle the aforementioned challenges. Existing works addressing training latency mainly focus on methods based on client selection, such as staleness-based client prioritizing scheme [8] and availability-based client assessment [9]. However, these methods could pose biases to the trained model if the dropped devices have certain data characteristics. Other latency optimization works emphasize the FL communication overhead, such as model pruning [10] and updates significance ranking [11]. However, these methods might induce potential errors in the trained model and cause under-utilization of computation resources. To handle data heterogeneity, a variety of methods have been proposed since the emergence of FL, such as cluster-based [12], personalized-based [13], and data augmentation-based [14] methods. However, these methods generally require intensive client-side computation and communication, so they are unsuitable for resource-constrained IoT devices.

Motivated by the aforementioned challenges in this paper, we propose FedKD-Prox, an enhanced distributed learning framework for IoT intrusion detection models based on the integration of knowledge distillation (KD) and federated proximal (FedProx). KD [15] can improve the accuracy of a student/simple model by supplying the distilled knowledge from a teacher/complex model. We propose to utilize the knowledge-transferring process in KD to optimize the communication overhead between the server and IoT devices in FL, enabling efficient model updates without interrupting the primary tasks of the IoT devices. On the other hand, since IoT devices are normally designed to handle specific tasks, they have diverse network traffic data, which results in heterogeneous training data. To address this, we propose to use FedProx [16] to alleviate the impact of data heterogeneity by penalizing the deviation between the local model and the server global model. It also reduces the training latency by incorporating stragglers to perform partial training. As shown in the simulation results, incorporating FedProx significantly improves the robustness of

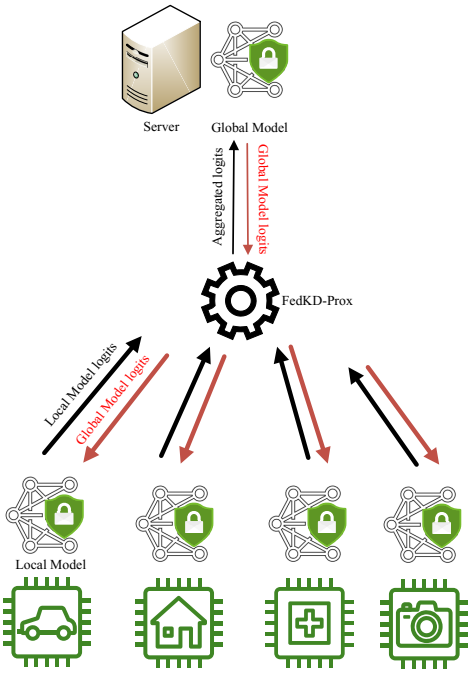

Fig. 1. FedKD-Prox over IoT network

the trained IoT intrusion detection models and minimize the intensity of local model drifting. Furthermore, we develop a knowledge-based FedProx to reduce the computation cost for model penalty calculation and enable an adaptive penalization tuning for each device. Our simulations demonstrated that FedKD-Prox outperforms other state-of-the-art FL algorithms in terms of intrusion detection accuracy and convergence speed within the time budget.

II. Proposed FedKD-Prox Framework

FedKD-Prox essentially consists of three components. Bi-directional knowledge distillation (BKD) allows the IoT devices and server to communicate with each other by sharing their model knowledge. Secondly, knowledge-based FedProx enables clients to compute model penalization via the distilled model knowledge instead of entire model parameters. Lastly, knowledge-based penalty tuning enables the server to adaptively adjust the strength of model penalization on each device's local model.

A. System model

Fig. 1 shows the architecture of FedKD-Prox in training IoT intrusion detection models. We describe the function of each layer as follows:

- **Intrusion detection layer** deploys the trained intrusion detection model on IoT devices and is responsible for buffering the traffic data as the model training data.
- **Model training layer** enables IoT devices to train their local models, extract local model knowledge, and send it as local updates to the server.
- **Communication layer** manages the traffic between IoT devices and the server during the training process. It detects potential malicious IoT devices and activities proposed in [17].
- **Aggregation/server layer** aggregates the model updates, extracts global knowledge to the clients, guides the training process of each IoT device, and improves the utilization of devices' computation resources.

B. BKD

As shown in fig.1, there are two communication steps involved in FL for training IoT intrusion detection models: devices transmit local updates to the server; the server distributes the aggregated model to the devices. We apply the standard KD in each communication step to enable each step to communicate via distilled model knowledge.

KD is essentially a model compression technique. It aims to train a student model with a simple structure to mimic the behaviours of a complex teacher model. To produce class probabilities for classification problems, a DL neural network generally uses its softmax/output layer to convert logits z_i into a soft targets q_m, which can be expressed as

$$q_i = \frac{\exp(z_i/C)}{\sum_{j=1}^{M}\exp(z_j/C)}, \tag{1}$$

where $m = 1, \cdots, M$ is the number of classifications, and the temperature hyperparameter C controls the smoothness of soft targets over the classes probability distribution.

In KD, by collecting the logits before the softmax layer, we define $Z = [z_1, \cdots, z_m]^T$ as the knowledge of a trained model. For each data batch, the process in which a student model uses a teacher's model knowledge Z_t to regularize its original loss function is called distillation. Specifically, the student model learns the teacher's knowledge by comparing its student knowledge Z_s with Z_t using the Kullback-Leibler divergence denoted by KL. We express this distillation process as

$$L_{KD} = KL(Z_s, Z_t) \times C^2, \tag{2}$$

where the authors in [15] suggested to scale the distillation loss L_{KD} by C^2. Finally, the student model uses a joint loss function $h_s(\cdot)$ weighted by α

$$\min_{w} h_s(w) = \alpha F_s(w) + (1-\alpha)L_{KD}, \tag{3}$$

where $F_s(\cdot)$ is the student model's original loss function that minimizes its model parameters w.

Applying KD in server-to-devices and devices-to-server communication steps in FL can be considered as a BKD paradigm. The server and devices no longer need to transmit the entire model to each other as required in the traditional FL framework. Instead, they transmit the model knowledge. To enable BKD, the server and devices additionally use a shared/proxy dataset to perform KD [18]. This shared dataset can be requested from devices or retrieved from a publicly available repository before training. In the following, we describe how BKD can be applied in FL for training IoT intrusion detection models:

- Devices-to-server: Assuming there are multiple devices as active trainers, each IoT device trains its student model using its local data and the model knowledge received from the server via (3). For local data, each local training batch in the device consists of a part of the device's private data and one batch of the shared dataset. Upon completing local training, each device acts as a teacher to extract the model logits, or batch knowledge, from each data batch of the shared dataset. These individual batch knowledge components are then aggregated to form the overall model knowledge, which is sent to the server as model updates.
- Server-to-devices: The server aggregates the received model updates by averaging the model knowledge from all devices. Then, the server computes its model logits for each batch of the shared dataset and uses the aggregated knowledge to calculate the distilled loss for each batch. Finally, the server trains its model using the distilled loss via (3) and acts as a teacher to extract the model knowledge. Lastly, the server broadcast the model knowledge to all devices.

These two KD-based communication steps repeat until the stop criteria are met.

C. Knowledge based FedProx

Besides the network traffic delay, devices with insufficient computing resources will cause training delay in FL. One solution is to selectively drop certain devices to improve the overall training speed, but it will reduce the prediction accuracy. This is because dropping the stragglers will also drop their stored data. As a result, the trained model will be biased towards the characteristics of the device selection policy [16]. To address this issue, FedProx can be used to incorporate the contributions of non-qualified/dropped devices by allowing them to perform partial training rather than completed training.

Assuming there are K devices, FedProx allows the device $k = 1, 2, \cdots, K$ to find inexact model parameters for each FL round. Specifically, FedProx compares the difference in the rate of change between the local objective function $h_k(w)$ and the local objective function $h_k(w^t)$. Here, w is the locally trained model parameters of the device k at the current local training epoch, whereas w^t is the global parameters initially received from the server. We consider the local parameters to be good enough if

$$\|\nabla h_k(w)\| \leq \sigma^k \|\nabla h_k(w^t)\|, \tag{4}$$

where ∇ denotes the gradient operator, and $\nabla h_k(w) = \nabla F_k(w)$. Here, $\sigma^k \in [0, 1]$ scales the inexactness of w and is associated with the client k's available computing resources

However, due to the distributive nature of IoT devices, their data, particularly traffic data, are often non-independent and identically distributed. Involving too many model updates trained by non-IID data will decrease the accuracy and robustness of the trained model. To address the issue, FedProx adds a proximal term that regularizes the impact of each local update on the global model, which can be expressed as

$$\min_w h_k(w, w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2, \tag{5}$$

where μ is a penalty constant that scales the proximal term $\frac{\mu}{2}\|w - w^t\|^2$.

To obtain this proximal term, the device must iterate and determine the parameter deviation of each network layer between the local and global models. This process will cause an additional computation burden to the device. To address this issue, we propose a knowledge-based FedProx that obtains the proximal term only through model knowledge. For each batch $b = 1, 2, \cdots, B$ in B training batches, we replace w and w^t in the proximal term with the device's batch knowledge $Z_{s,b}$ and server's batch knowledge $Z_{t,b}$ as

$$\min_w h_k(w) = F_k(w) + \frac{\mu}{2}\|Z_{s,b} - Z_{t,b}\|^2. \tag{6}$$

Using (8) can accelerate the computation of the proximal term since the size of model knowledge only depends on the number of logits in the last layer. Instead of iterating all the parameters of network layers of the local and global model in each data batch, only one layer is sufficient to obtain the proximal term.

D. Adaptive model penalty tuning

However, the performance of FedProx highly depends on the value of μ that applies to all the training devices. That is, a large μ may constrain the local model to be closely aligned with the initially received model, whereas a small μ would have negligible impact. A fixed μ might be unfair to the local model trained using balanced data. It receives the same penalization as those trained using unbalanced data. The original FedProx lacks a mechanism to adaptively determine the value of μ for each trainer.

To address the problem, we propose *Remark 1* to dynamically tune the model penalty based on model knowledge. It first identifies which device's local model requires additional or less penalization by comparing its model knowledge with that of the server model. Then, the server adjusts the penalty constant μ_k for each device k, either increasing or decreasing it based on the assessment.

Before aggregating local knowledge, the server utilizes *Remark 1* to assess the knowledge received from each device. This process identifies the local knowledge that significantly deviates from the server's model knowledge. Specifically, the server determines which clients necessitate penalty adjustments by comparing its previously extracted server knowledge with that received from each device.

Remark 1 (Outlier client identification): For each received device's model knowledge, the server calculates the Euclidean distance between the device's model knowledge and the server's model knowledge from the previous FL round. Then, we use the interquartile range (IQR) to determine a $lowerbound$ and $upperbound$ among these norms

Algorithm 1 : FedKD-Prox

Input: Initial μ; Shared dataset D; Warm-up rounds U; FL rounds T; Devices $k \in K$
for each warm-up round $u = 0, 1, \cdots, U$ **do**
    Devices download initial model parameters
    Devices train local models and send model updates
    Server evaluates the computing power of each device
    Server aggregates the local updates
end for
Server sets σ^k to each device k (using 4)
for each FL round $t = 0, 1, \cdots, T$ **do**
    Devices download server model knowledge
    Devices train local model (using 7)
    Devices extract the knowledge using D and send it to the server
    Server adjusts μ_k for each device k (Remark 1)
    Server aggregates local knowledge
    Server trains the global model using D and extracts the model knowledge
    Server sends the model knowledge to all devices
end for
Devices deploy the trained intrusion detection model

For each device below the $lowerbound$, the server decreases the value of μ for the device because the local model is over-penalized. It is too close to the initial model and might not be able to reflect sufficient local data variance. For each device above the $upperbound$, the server increases μ because the local model is under-penalized. It is too far from the initial model and could bias the trained FL model. We set the adjustment amount to be $\pm 50\%$ of the current value of μ. Authors in [19] also utilized IQR to determine outlier devices in FL. Their approach aims to quantify the deviation of training data among heterogeneous clients but can expose potential risks on data privacy. Finally, we define a joint local objective function as a summation of the original local loss, KD loss, and the adaptive proximal term, which can be expressed as

$$\min_{w} h_k(w) = \alpha(F_k(w) + \frac{\mu_k}{2}\|Z_{s,b} - Z_{t,b}\|^2) + (1-\alpha)L_{KD}. \quad (7)$$

E. FedKD-Prox algorithm

By combining the three aforementioned components, we summarize FedKD-Prox in Algorithm 1. Firstly, the server builds the shared dataset by requesting shareable data from each device or retrieving it from a public repository. Prior to executing BKD, the server and devices exchange regular model updates and perform standard FL. Each device performs a uniform amount of training, and the server assesses the computing capabilities of each device by measuring the response times.

Following this initialization phase, each device k receives a specific σ^k from the server that allows it to solve its local objective partially. After that, the server and devices engage in BKD and perform FL through knowledge-based FedProx. Finally, the trained intrusion detection model is deployed across the devices.

TABLE I
DATASETS SUMMARY

	Samples	Categories	Features	Devices
CICIoT	46,239,784	33	46	105
CICIoMT	7,160,831	19	49	40
CICIDS	2,937,876	16	45	12

TABLE II
TRAINING SETUP

Hyperparameters	**Value**
Shared dataset	1% of each device's dataset
Training data	90% of sampled dataset
Testing data	10% of sampled dataset
Number of trainers	50
Time budget (FL rounds)	30
Local iteration threshold	40
Initial penalty constant (μ)	0.01
Device model hidden layers	2
Nodes per hidden layers	500
Batch size	128
Learning Rate	0.01
Optimizer	Stochastic Gradient Descent
Criterion	Cross Entropy Loss

III. SIMULATION

We implement the proposed Algorithm 1 and compare it with FedAvg and vanilla FedProx.

A. Dataset

As shown in Table I, we evaluate our proposed framework based on real intrusion classification datasets conducted by the Canadian Institute for Cybersecurity, which are CICIoT2023 [1], CICIoMT2024 [20], and CICIDS2017 [21]. These datasets are well-developed based on executing a real topology of IoT devices in dedicated networks.

Due to the large size of the original datasets, we extract 1×10^6 samples from each dataset. In addition, the original datasets are imbalanced regarding the deficiency between the majority and minority classes. We apply the synthetic minority oversampling technique (SMOTE) to generate a considerable amount of synthetic samples to reduce the potential model bias in addition to the simulated data heterogeneity.

B. FL scenarios setup

First, we propose an ideal federated learning scenario where devices are equally powerful and each contains identically independent distributed (IID) data. Since the objective function of FedKD-Prox includes two regularization terms (distilled loss and proximal term), both model accuracy and convergence speed may be affected in this ideal FL scenario. This setup serves as an ablation study to verify the performance of FedKD-Prox. Conversely, we propose a practical FL scenario in which devices vary in power and each holds heterogeneous or non-IID data.

To simulate this ideal environment, each client performs the same training iterations and stores equally partitioned

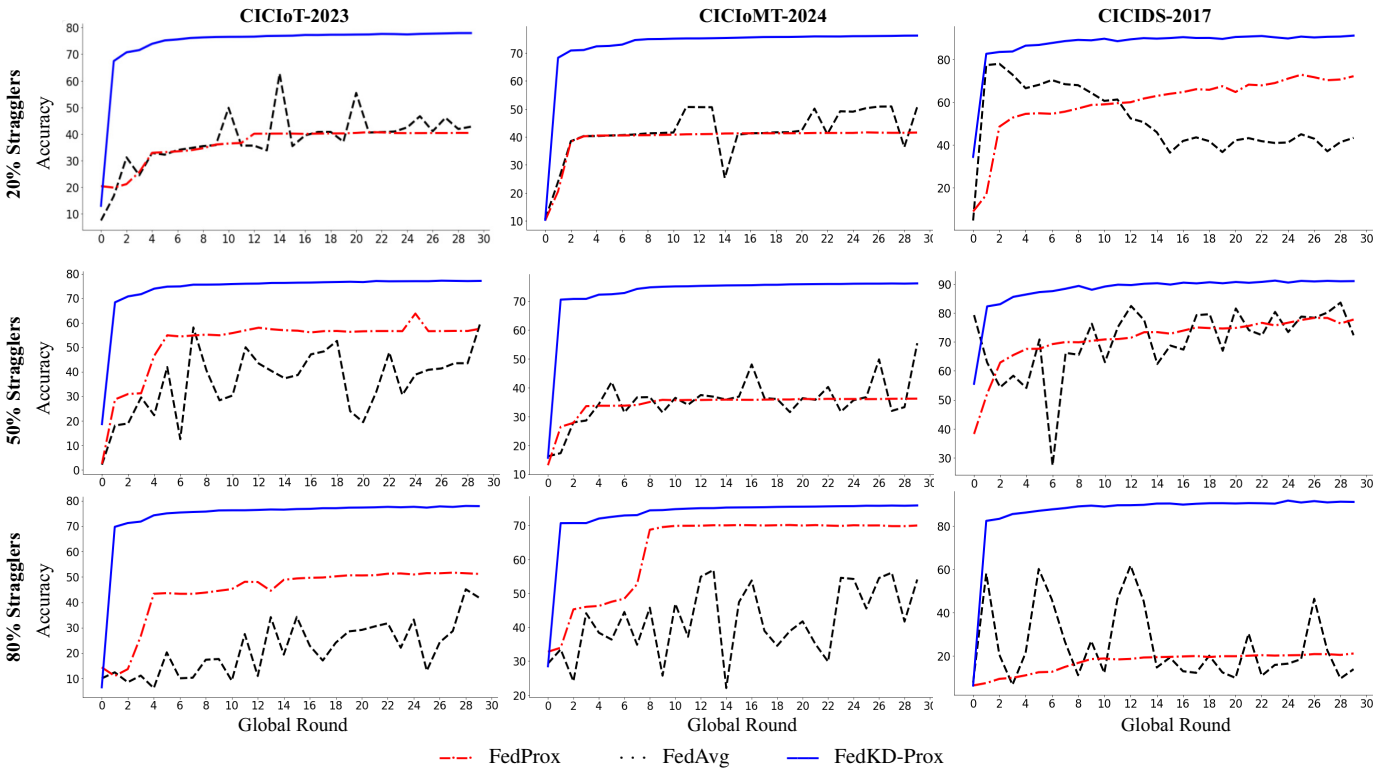

Fig. 2. Testing accuracy under practical FL scenario

data. If our baselines perform better than FedKD-Prox in the ideal scenario but less favorable in the practical scenario, this contrast will confirm the effectiveness of FedKD-Prox.

To simulate data heterogeneity in the practical FL scenario, the intention is that each device will have limited knowledge of the overall data distribution. Thus, the trained local model can exhibit biases toward their own local data. To achieve this, each device has the data associated with a random number of attack classes/labels.

To simulate imbalanced computing power among IoT devices in FL, we conduct simulations with different numbers of stragglers, ranging from 20%, 50%, and 80% of total number of devices. To reflect (6), we assign a random number of training iterations to each device to represent the maximum training work that the device can perform within the time budget. Then, we set a maximum iteration threshold for all devices, with stragglers performing training below the threshold while non-stragglers perform the maximum iteration. FedAvg employs client-selection methods to improve training speed. We assume the server drops a device if it does not perform sufficient local training iterations. We set this dropping baseline to be 50% of the iteration threshold. With these setups, the time budget can be reflected in the total number of FL training rounds. We set the time budget to be 30.

C. Model implementation

We implement the three algorithms using Pytorch [22] and adapt them to the same training setup shown in Table II. We use a simple artificial neural network (ANN) for each device with only two hidden layers to simulate the IoT computing constraints. The server has a more complex ANN model with five hidden layers. It performs training tasks only in FedKD-Prox.

IV. Evaluation of FedKD-Prox

We evaluate the performance of FedKD-Prox by comparing the prediction accuracy and convergence speed with baselines under the two FL scenarios.

A. Model performance evaluation

As shown in Fig. 2, when devices have limited resources and store heterogeneous data, FedKD-Prox significantly outperforms FedAvg and FedProx in testing accuracy and convergence speed. In this case, FedAvg can not be able to converge within the given time budget. Although FedProx overall has a higher prediction rate and shows some signs of converging by involving all the stragglers, its performance fluctuates when exposed to different numbers of stragglers. This is caused by a fixed penalty term μ held by all clients and used throughout the training process. That is, a fixed μ might not effectively regulate the local models, as it could lead to either excessive or inadequate penalization for some clients.

The dynamic μ adjustment in FedKD-Prox helps to alleviate this issue by assigning each device with a proper μ. This is one of the reasons that FedKD-Prox has stable prediction performance regardless of different numbers of stragglers. Another key factor is that the server incorporates a much more

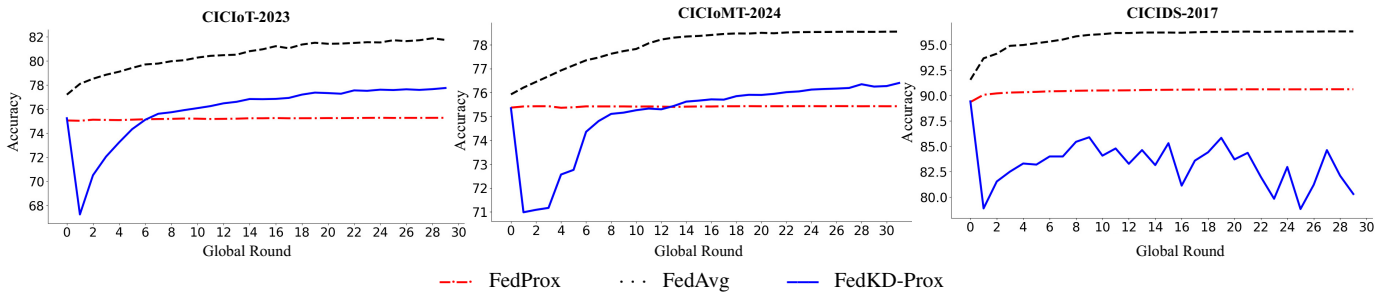

Fig. 3. Test accuracy under ideal FL scenario

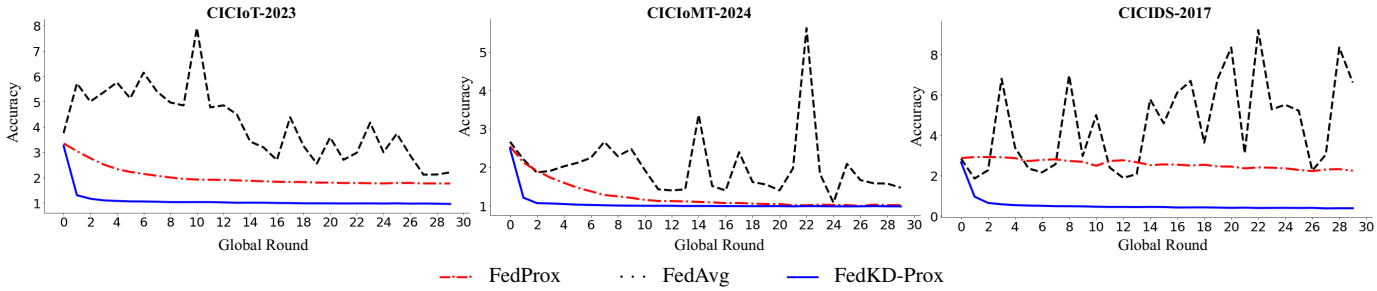

Fig. 4. Training loss under practical FL scenario with 80% Stragglers

TABLE III
NUMBER OF MODEL PARAMETERS TRANSMITTED

	CICIoT	**CICIoMT**	**CICIDS**
Local model	540,034	533,515	540,034
Compressed to	456,960	215,040	255,360
Server model	1,291,534	1,285,019	1,285,015
Compressed to	456,960	215,040	255,360

complex model structure in KD that guides the training process of all trainers.

Fig. 3 demonstrates the result of the ablation study. FedAvg has a higher prediction accuracy than FedProx and FedKD-Prox. This accuracy gap in the ideal FL scenario is associated with the model penalization mechanism in FedProx and FedKD-Prox. It forces the local model to be close to the starting point and makes the local model less impactful to the global model. We find that a higher μ can result in a larger model penalization, leading to a larger gap. Therefore, our ablation study confirms the effectiveness of FedKD-Prox.

B. Communication cost reduction

FedKD-Prox reduces the communication overhead between the server and devices. As shown in Table III, the algorithm transmits significantly fewer parameters in both client-to-server and server-to-client communication steps compared to transmitting the entire model in FedAvg and vanilla FedProx.

In FedKD-Prox, the size of model updates is mainly determined by the number of attack classes/labels to be classified and the size of the shared dataset. The reduction shown in Table III is achieved using a shared dataset comprising 1% of each device's local data, which is assumed to be shareable. This feature allows the server and devices to have a flexible number of hidden layers. These layers do not add to the communication burden as opposite to other FL algorithms. Thus, the server and devices can train a more complex model to produce a better result without exploiting the communication resource.

C. Convergence analysis

As shown in Fig. 4, FedKD-Prox converges to the minimal training loss rapidly with the presence of heterogeneous data and constrained computing resources. This performance gap can be attributed to two key factors. Firstly, FedKD-Prox effectively addresses data heterogeneity and leverages the contributions from stragglers. Secondly, the involvement of the server in model training also plays a crucial role. More importantly, by receiving a small fraction of local data from each device as part of the shared dataset, the server gains a broader view of the overall data distribution among devices. This helps the server produce valuable model knowledge that further reduces the impact of heterogeneous data in each device and accelerates the model convergence.

However, requesting the local data might not be realistic when data privacy is the top priority. A simple way is to select and use a publicly available dataset as the shared dataset. However, these datasets might not be well-generalized and adapted to specific training environments. Thus, potential future work lies in developing proper ways to obtain the shared dataset in resource-constrained settings.

V. CONCLUSION

In this work, we propose FedKD-Prox to enhance the FL-based intrusion detection model in IoT networks based on KD

and FedProx. The framework improves the detection rate of the model under the limited resources budget by involving partial trainers/stragglers. It also effectively mitigates the impact of data heterogeneity through a knowledge-based model penalization. Meanwhile, we propose a BKD paradigm to effectively reduce the communication overhead in FL. Simulation results have shown that FedKD-Prox can efficiently train intrusion detection models with guaranteed convergence and accuracy under resource-constrained and heterogeneous environments. The proposed framework could be applied to other model training settings that require privacy preservation and have limited edge resources.

VI. Acknowledgment

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Program under Grant RGPIN-2023-04082 and RGPIN-2020-06547. The work of L. Zhang is partially supported by the National Science Foundation under Grants CCF-2427316, CCF-2426318. and CNS-2418308.

References

[1] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, 2023.

[2] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[3] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138 509–138 542, 2021.

[4] S. Wijethilaka and M. Liyanage, "A federated learning approach for improving security in network slicing," in *IEEE GLOBECOM 2022*, 2022, pp. 915–920.

[5] T. H. Ahmed, J. J. Tiang, A. B. Mahmud, and G. C. Chung, "Detection and mitigation of sql and jamming attacks on switched beam antenna in v2v networks using federated learning," in *IEEE ISIEA 2023*, 2023, pp. 1–6.

[6] X. He, Q. Chen, L. Tang, W. Wang, T. Liu, L. Li, Q. Liu, and J. Luo, "Federated continuous learning based on stacked broad learning system assisted by digital twin networks: An incremental learning approach for intrusion detection in uav networks," *IEEE Internet of Things Journal*, vol. 10, no. 22, pp. 19 825–19 838, 2023.

[7] Q. Pan, J. Wu, A. K. Bashir, J. Li, W. Yang, and Y. D. Al-Otaibi, "Joint protection of energy security and information privacy for energy harvesting: An incentive federated learning approach," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3473–3483, 2022.

[8] B. Wu, F. Fang, and X. Wang, "Joint age-based client selection and resource allocation for communication-efficient federated learning over noma networks," *IEEE Transactions on Communications*, vol. 72, no. 1, pp. 179–192, 2024.

[9] P. Agbaje, A. Anjum, Z. Talukder, M. Islam, E. Nwafor, and H. Olufowobi, "Fedcime: An efficient federated learning approach for clients in mobile edge computing," in *IEEE EDGE 2023*, 2023, pp. 215–220.

[10] H. Liu, Y. Shi, Z. Su, K. Zhang, X. Wang, Z. Yan, and F. Kong, "Fedadp: Communication-efficient by model pruning for federated learning," in *IEEE GLOBECOM 2023*, 2023, pp. 3093–3098.

[11] Y. Li, J. Bai, D. Li, and W. Li, "Communication-efficient federated learning with an event-triggering strategy," in *IEEE DDCLS 2022*, 2022, pp. 347–352.

[12] Z. Li, Z. Guan, S. Yuan, N. An, and X. Liang, "Rocfl: A robust clustered federated learning framework towards heterogeneous data," in *IEEE ICN 2023*, 2023, pp. 259–264.

[13] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 9587–9603, 2023.

[14] H. Zhang, Q. Hou, T. Wu, S. Cheng, and J. Liu, "Data-augmentation-based federated learning," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 22 530–22 541, 2023.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems*, vol. 2, 2020, pp. 429–450.

[17] G. Qiang, F. Fang, and X. Wang, "Security and efficiency enhancement for split learning: A machine learning based malicious clients detection approach," in *IEEE PIMRC 2023*, 2023, pp. 1–6.

[18] P. Qi, X. Zhou, Y. Ding, Z. Zhang, S. Zheng, and Z. Li, "Fedbkd: Heterogenous federated learning via bidirectional knowledge distillation for modulation classification in iot-edge system," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 1, pp. 189–204, 2023.

[19] Y. Jeong and T. Kim, "A cluster-driven adaptive training approach for federated learning," *Sensors*, vol. 22, no. 18, 2022.

[20] S. Dadkhah, E. C. P. Neto, R. Ferreira, R. C. Molokwu, S. Sadeghi, and A. Ghorbani, "Ciciomt2024: Attack vectors in healthcare devices-a multi-protocol dataset for assessing iomt device security," *Preprints*, February 2024.

[21] S. Iman, L. Arash, Habibi, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.