2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit) | 979-8-3503-1102-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/EUCNC/6GSUMMIT58263.2023.10188312

Distributed Learning-Based Intrusion Detection in 5G and Beyond Networks

Cheolhee Park*, Kyungmin Park†, Jihyeon Song‡, and Jonghyun Kim§
Electronics and Telecommunications Research Institute
Daejeon, South Korea, 34129
Email: {*chpark0528, †kmpark, ‡dmon95, and §jhk}@etri.re.kr

***Abstract*—As mobile technology has evolved over generations, communication systems have advanced along with it. Moreover, the 6th generation (6G) of mobile networks is expected to evolve into a more decentralized and open environment. Meanwhile, with these advancements in network systems, the attack surface that can be exposed to adversaries has expanded, and potential threats have become more sophisticated. To secure network systems from these potential attacks, various studies have focused on intrusion detection systems. In particular, studies on artificial intelligence-based network intrusion detection systems have been actively conducted and have shown remarkable results. However, most of these studies concentrate on centralized environments and may not be suitable for deployment in distributed systems. In this paper, we propose a distributed learning-based intrusion detection system that can efficiently train predictive models in a decentralized environment and enable learning in systems with varying computing capabilities. We leveraged a state-of-the-art split learning approach, which allows for models to be trained in distributed systems with different computing resources. In our experiments, we evaluate the models using data collected in a 5G mobile network environment and demonstrate that the proposed system can be applied for network security in the next-generation mobile environment.**

***Keywords*—5G, 6G, network security, intrusion detection system, distributed learning, split learning**

I. Introduction

With the development of mobile communication technology, numerous data is communicated through the network. Moreover, 5G-advanced and 6th generation mobile networks that pursue openness and decentralization are expected to handle complex data flows across a range of domains such as computers, sensors, and the Internet of Things (IoT). However, as network systems have evolved, access points have become more diverse and the attack surface that can be exposed to adversaries has expanded. Additionally, cyberattacks have become more sophisticated and have rapidly increased in frequency. Accordingly, to address these potential threats, many studies have been conducted on network intrusion detection systems (NIDS) to detect cyberattacks in advance.

Recently, research on artificial intelligence (AI)-based NIDS has been conducted extensively and remarkable results have been reported (e.g., [1]-[10]). Initially, most research focused on applying basic machine learning models such as regression models and support vector machines, and it has been extended to applying deep learning models such as deep neural networks, long short-term memory, and generative models. While these approaches have shown remarkable results in detecting network intrusions, most of them were designed for centralized environments. In other words, applying these approaches to a distributed environment may be inefficient, resource-intensive, or even unfeasible.

Deploying AI models in a distributed environment can be straightforward in scenarios without constraints. However, building a distributed learning system that considers privacy and computing resources can be challenging. For example, when building an AI learning system in mobile communication environments, the central server have to analyze information without direct access to data generated from user equipment to preserve user privacy. Moreover, when each node has different computing resources, administrators have to consider the node with the worst computing resources as a criterion to build a distributed learning system, which can result in reduced performance of the predictive model. In this regard, federated learning (FL)-based approaches [11]-[21] have been proposed to train network intrusion detection models in distributed environments while preserving data privacy. FL trains the model through an iterative process of local model training and global model aggregation, where each participating node trains a model on its own data and the central node updates the global model with the locally trained parameters. This enables model training in a decentralized environment while preserving data privacy (without direct access to local data). However, FL-based approaches do not consider the computing power of each node, and thus can result in an uneven distribution of computational workload, leading to slow convergence and longer training times.

In this paper, we propose a distributed learning-based network intrusion detection system that is capable of training data in a decentralized environment while considering the computing power of each local node and preserving privacy. To address the aforementioned issues, we utilize a state-of-the-art learning system, Split Learning [22], which enables distributed learning that takes into account the different computing power of each node. In particular, we focus on the SplitNN model, which not only trains a global model in a data-distributed environment, but also splits the global model architecture while considering the computing power of each local node. To demonstrate the efficacy of the system, we conduct experiments and evaluate the system on 5G-NIDD [23], a real 5G dataset collected from 5G mobile environments. Through experiments, we show that the proposed system is

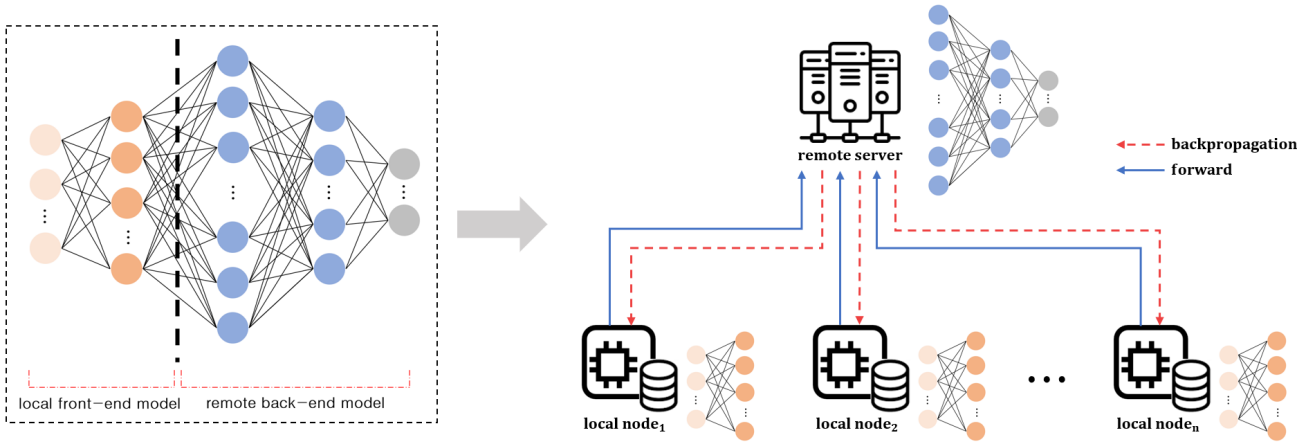

Fig. 1. A basic example of split learning.

suitable to be applied as a network intrusion detection system for distributed environments in the next-generation mobile environment.

The rest of the paper is organized as follows: Section 2 provides a brief review of relevant studies, and Section 3 provides background on split learning and split computing systems. In Section 4, we describe our methodology and the proposed framework. In Section 5, we evaluate the proposed system using datasets collected from real 5G environments, and present the experimental results. Finally, we present concluding remarks and discussion in Section 6.

II. Related work

In the field of artificial intelligence-based NIDS, various studies have been conducted to apply machine learning and deep learning technologies (e.g., [1]-[10]). While these approaches have shown remarkable results, most of them basically considered centralized environments. Recently, to build intrusion detection models while preserving data privacy in distributed environments, federated learning-based approaches have been actively studied. Basically, studies have proposed IDS systems in distributed environments integrating FL with multi-layer perceptron (MLP) and neural network models [11, 12, 13, 14, 15, 16]. In [17] and [18], the authors integrated FL with the Gated Recursive Units (GRU)-based models, and utilized the FedAvg algorithm as the model aggregation. In [19], the authors built a decentralized IDS system combining FL and deep-belief networks, and showed that the proposed method can outperform the conventional machine learning approaches. To detect cyber threats against industrial CPSs, [20] proposed a novel federated deep learning scheme. In [20], the authors developed a FL framework combined with CNN and GRU. To secure network and service management automation, [21] proposed a FL-based anomaly detection system incorporating the ZSM architecture. In [21], the authors built a multi-stage federated learning framework, and showed that the proposed framework can improve the detection performance. Although these studies have shown remarkable results and made it possible to detect network threats while preserving data privacy in a distributed environment, they did not consider the computing power of each node. Different from these results, we proposed the split learning-based intrusion detection system that can train data in a decentralized environment while considering the computing power of each local node and preserving privacy.

III. Background

In this section, we briefly illustrate the concept of split learning, which is a key component of our proposed system. Additionally, we describe the split computing environment that is expected to be realized in the next generation of mobile environment.

A. Split learning

Split learning [22] is a machine learning technique that enables distributed learning in a decentralized environment while preserving privacy, and is designed to address the limitations of traditional Federated Learning, such as the uneven distribution of computational workloads. In Split Learning, the model is partitioned into two distinct components (see Fig. 1): a local front-end model and a remote back-end model. The local front-end model runs on local nodes with limited computational capabilities and performs operations on local data. The remote back-end model, located on a central server, receives intermediate representations or activations from the local front-end models and continues the computation on the global model. This setting ensures that the privacy of sensitive data is preserved, as it remains on the local node and is not transmitted to remote servers.

In terms of model training, the remote back-end model aggregates the intermediate representations from multiple local nodes and computes the current loss value using the aggregated intermediate outputs.[1] The central server then sends the final backpropagated loss value of the back-end model to each node, and each local node updates the weight parameters of the front-end model using the loss value sent by the central server. Simultaneously, the central server updates the weight parameters of the back-end model during backpropagation of the current loss value. This iterative process allows the model to learn from the collective data across the domain while preserving the privacy of local data.

[1]From the perspective of supervised learning, one intermediate output contains the activation results of a specific layer and the corresponding label.

One of the fundamental characteristics of split learning is consistency. That is, split learning ensures that the neural network being trained at the k-th iteration is identical to the neural network if it was trained by one entity (as stated in Lemma 1 in [22]).This property allows the entire computational workload of training a neural network model to be adaptively segmented, taking into account the computing capabilities of participants. In particular, administrators can allocate a relatively small portion of the overall model architecture to local nodes with limited computing resources, and allocate the rest of the back-end model to a central server with sufficient computing resources. In this paper, we utilize the SplitNN model with the aim of splitting the network intrusion detection model in a distributed mobile environment.

B. Split computing in 6G mobile networks

Split computing is a concept expected to develop in 6G mobile networks [24], where computing workloads are split across multiple nodes. The objective behind split computing is to allocate a small portion of the entire computing workload to nodes with relatively limited computational capabilities (e.g., user mobile devices), and allocate the remaining portion (heavy computing workload) to nodes with sufficient computational capabilities (e.g., edge servers, clouds).

Split computing will enable 6G mobile users to experience faster and more efficient application services by leveraging edge servers and/or cloud. In particular, local computations and data pre-processing will be performed on user equipments, reducing the amount of data that should be transmitted to the cloud. This will reduce network latency and increase the efficiency of the overall network system. Furthermore, it will enable users to experience application services faster by allocating burdensome tasks that would have taken a long time and many resources to the edge server and/or cloud. In addition, split computing is expected to improve the reliability and security of 6G mobile networks by utilizing the edge and the cloud. This means that a split computing system will preserve privacy by distributing and processing data on each local node (instead of sending data to a central server), making the computational results more robust and resilient to failures.

With this in mind, we aim to build a distributed learning-based network intrusion detection system, taking into account the split computing environment that is expected to be implemented in next-generation mobile communication networks.

IV. Proposed methdology

We built a split learning-based network intrusion detection system that enables local nodes and an edge computing system to jointly train predictive models. In particular, we considered a scenario where multiple local nodes each hold local data collected from their respective areas, but have limited computational power. Our goal is to build global predictive models that can discriminate network threats across the entire network by learning from all the data distributed across the multiple nodes, while preserving data privacy.

For the predictive model, we combined a deep neural network (DNN) and a convolutional neural network (CNN) with split learning to build intrusion detection systems. In the case of the DNN model, we designed the model architecture to process three hidden layers with 10, 20, and 10 neurons in each layer, and the ReLU function was utilized for activation. The split point for the entire model was specified as the first layer, meaning that the computations from the input layer to the first layer were allocated to the local nodes, while the rest of the back-end layers, from the second layer to the output layer, were allocated to the remote server.

Algorithm 1 Detection model training on node$_i$ with split learning

Input: node$_i$'s dataset $\mathcal{D}^i$, current weights $\mathcal{W}_1$ of the front-end model and $\mathcal{W}_2$, $\mathcal{W}_3$ of the back-end model

Local-side
- Forward propagation
  - $\mathcal{X} = \mathcal{F}(\mathcal{W}_1, \mathcal{D}^i)$
- Output transmission
  - Send $(\mathcal{X}, Label)$ to the edge server

Server-side
- Forward propagation
  - $\mathbf{c} = \mathcal{F}(\mathcal{W}_3, \mathcal{F}(\mathcal{W}_2, \mathcal{X}))$
- Compute gradients
  - $\mathbf{g}_{server} = \mathcal{F}'(\mathcal{W}_2, \mathcal{F}'(\mathcal{W}_3, \mathcal{L}(\mathbf{c}, Label)))$
- Send gradients
  - Send $\mathbf{g}_{server}$ to node$_i$

Local-side
- Compute gradients
  - $\mathbf{g}_{node} = \mathcal{F}'(\mathcal{W}_1, \mathbf{g}_{server})$
- Update $\mathcal{W}_1$
  - $\mathcal{W}_1 = \mathcal{OPT}(\mathcal{W}_1, \eta, \mathbf{g}_{node})$

Output : updated weight $\mathcal{W}_1$

For the CNN model, we designed the architecture to process two convolutional layers and one fully connected layer. Considering the type of network traffic data, we set the convolutional layer to be one-dimensional. The convolutional layers were configured to process 15 convolution filters with a window size of 3 and 10 convolution filters with a window size of 2, and the fully connected layer was configured to have 10 neurons. Additionally, we applied batch normalization after each convolutional layer to reduce the risk of overfitting and the internal covariate shift, and the ReLU function was used as the activation function. Similar to the DNN model, the first layer of the entire model was specified as the split point. Algorithm 1 presents a detailed workflow for training the detection model using split learning on the i-th local node.

Before running the algorithm, the local node can obtain the current weight vector of the front-end model from the final updater. Once node$_i$ gets the current weight vector $\mathcal{W}_1$, it proceeds to train the front-end model by running the algorithm based on the local data $\mathcal{D}^i$ of node$_i$. The local node first feeds the local data $\mathcal{D}^i$ to the front-end model with the current weights $\mathcal{W}_1$, and then sends the results

TABLE I. DATA DISTRIBUTION OF 5G-NIDD

Class	Rows	Weight (%)
Benign	477,737	39.29%
HTTP Flood	140,812	11.58%
ICMP Flood	1,155	0.09%
SYN Flood	9,721	0.79%
SYN Scan	20,043	1.64%
Slowrate DoS	73,124	6.01%
TCP Connect Scan	20,052	1.64%
UDP Flood	457,340	37.61%
UDP Scan	15,906	1.30%
Total	1,215,890	100%

(output of the first layer) and the corresponding labels to the edge server. Subsequently, the edge server trains the back-end model based on the received results and labels. As with the local node, the server first feeds the results of the front-end model to the back-end model that the server maintains, and calculates the loss value to computes the gradients for the current model. The server then sends the last gradients $\mathbf{g}_{server}$ computed from the first layer of the back-end model to the local node node_i. Note that, the server can update the weights of the back-end model immediately after computing the gradients. Finally, node_i computes the gradients for $\mathcal{W}_1$ using the gradients sent by the server with a gradient-based optimization process $\mathcal{OPT}$, and then, updates the front-end model. The above process proceeds sequentially by all local nodes to learn the entire dataset distributed across multiple nodes, and the trained model is utilized as a network intrusion detection model. Note that, although the algorithm presented the process for a single training flow, it can be performed iteratively depending on the size of the local dataset.

From the perspective of the entire system, local nodes are considered as pico base stations that hold network traffic data generated by user equipment in their coverage or even as user equipment itself, and the central server, which aims to build a network intrusion detection system, is located in an edge computing system such as a Multi-Access Edge Computing (MEC) system. In our experiment, we assumed that there are 9 local nodes and a single central server.[2] In this environment, we evaluated our system based on datasets collected in real 5G environments, and demonstrated that the proposed system can be a key technique for network security in the next-generation mobile communication network.

V. EXPERIMENTS AND EVALUATIONS

In this section, we first describe the dataset and present the experimental results along with a comparative analysis.

[2]Although we assumed 9 local nodes for the experiment, it can vary depending on the environment.

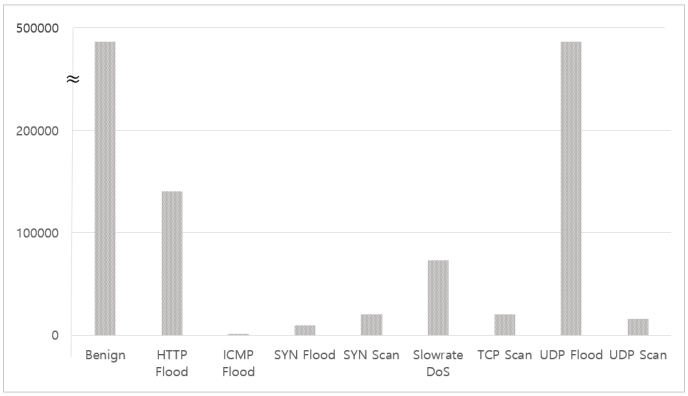

Fig. 2. Histogram of 5G-NIDD by class.

A. Dataset description

In this study, our focus is on building a network intrusion detection system for mobile networks. To evaluate our system, we used a dataset called 5G-NIDD [23]. This dataset was collected in a real 5G environment and is the first dataset generated for research purposes on AI-based network intrusion detection in a 5G environment. 5G-NIDD contains network traffic data for eight attack scenarios and benign behaviors. The attacks can be broadly categorized into two categories: DoS and Port scan. DoS is one of the most common threats that compromise the accessibility of legitimate users, while Port scan is a preliminary step taken to retrieve information about a target before launching an actual attack.

The dataset contains 1,215,890 rows collected from two distinct base stations, and each data point has 25 features.[3] Note that, we excluded any unnecessary features that did not affect detection performance, such as the index and sequence of each row, and encoded the categorical features as one-hot vectors. Consequently, The pre-processed dataset has a total of 91 features, and we applied Z-score normalization to normalize the numerical features. Z-score normalization is a well-known method for scaling numerical features, and we chose to use it as it has been used in [9]. We note that there was no significant difference in model performance between the normalization methods used (e.g., the min-max normalization method). Table 1 presents the distribution of 5G-NIDD, and Fig. 2 shows a histogram that provides visibility into the data distribution. As shown in the table and histogram, 5G-NIDD has severe data imbalance between classes. With this dataset, we evaluate the performance of the proposed model under several scenarios.

B. Experimental scenario

To demonstrate the effectiveness of the proposed system, we created several experimental scenarios by varying the conditions in the data distribution, taking into account the real-world environment. In a distributed system, data originating from a particular region may introduce data bias (e.g., skewness for certain classes) that is not representative of the population. This is because regional data can be influenced by unique social, economic or cultural factors specific to that

[3]5G-NIDD contains various types of data, such as pcap and flow, but we only dealt with datasets processed in flow units.

TABLE II. EXPERIMENTAL RESULTS FOR MULTI-CLASSIFICATION.

	Benign			HTTP Flood			ICMP Flood		
Algorithm	$Precision$	$Recall$	$F1$-score	$Precision$	$Recall$	$F1$-score	$Precision$	$Recall$	$F1$-score
Baseline	99.53% / 99.62%	96.33% / 96.30%	97.91% / 97.93%	99.76% / 99.60%	99.71% / 99.73%	99.73% / 99.67%	100% / 100%	99.65% / 100%	99.82% / 100%
Ideal	99.55% / 99.71%	96.40% / 96.22%	97.95% / 97.94%	99.63% / 99.69%	99.65% / 99.67%	99.64% / 99.68%	100% / 99.31%	100% / 100%	100% / 99.65%
Imbalance	99.47% / 99.58%	96.30% / 96.21%	97.86% / 97.87%	99.64% / 99.53%	99.75% / 99.83%	99.69% / 99.68%	100% / 100%	100% / 100%	100% / 100%
Skewed	99.94% / 99.94%	92.58% / 92.97%	96.12% / 96.33%	99.83% / 99.84%	98.64% / 99.27%	99.23% / 99.55%	100% / 98.29%	99.65% / 99.65%	99.82% / 98.96%

SYN Flood			SYN Scan			Slowrate DoS		
$Precision$	$Recall$	$F1$-score	$Precision$	$Recall$	$F1$-score	$Precision$	$Recall$	$F1$-score
99.87% / 99.71%	99.91% / 100%	99.89% / 99.85%	99.92% / 99.81%	99.76% / 99.58%	99.84% / 99.70%	99.42% / 99.50%	99.55% / 99.26%	99.48% / 99.38%
99.79% / 99.71%	100% / 100%	99.89% / 99.85%	99.86% / 99.88%	99.68% / 99.72%	99.77% / 99.80%	99.35% / 99.38%	99.32% / 99.42%	99.34% / 99.40%
99.75% / 99.79%	100% / 99.91%	99.87% / 99.85%	99.91% / 99.68%	99.38% / 99.76%	99.64% / 99.72%	99.52% / 99.68%	99.32% / 99.11%	99.42% / 99.40%
99.54% / 99.67%	99.30% / 100%	99.42% / 99.83%	99.70% / 99.60%	99.86% / 99.70%	99.78% / 99.65%	97.14% / 98.68%	99.60% / 99.67%	98.35% / 99.18%

TCP Connect Scan			UDP Flood			UDP Scan		
$Precision$	$Recall$	$F1$-score	$Precision$	$Recall$	$F1$-score	$Precision$	$Recall$	$F1$-score
99.64% / 99.84%	99.98% / 99.90%	99.81% / 99.87%	96.30% / 96.27%	99.53% / 99.62%	97.89% / 97.92%	99.89% / 99.77%	99.67% / 99.84%	99.78% / 99.81%
99.72% / 99.88%	99.94% / 99.88%	99.83% / 99.88%	96.37% / 96.19%	99.55% / 99.71%	97.93% / 97.92%	99.89% / 99.79%	99.74% / 99.84%	99.82% / 99.82%
99.90% / 99.78%	99.76% / 99.92%	99.83% / 99.85%	96.26% / 96.19%	99.46% / 99.58%	97.84% / 97.85%	99.22% / 99.92%	99.94% / 99.62%	99.58% / 99.77%
99.87% / 99.92%	98.94% / 99.76%	99.40% / 99.84%	92.82% / 93.16%	99.96% / 99.95%	96.26% / 96.43%	99.97% / 99.64%	99.62% / 99.72%	99.79% / 99.68%

region. This leads to differences between data collected in one area and data collected in another, resulting in imbalanced or skewed datasets. When training an AI model with these biased datasets, the model may not generalize well to the population, as it only trains a narrow subset of the data. This can lead to incorrect predictions and decisions, particularly for underrepresented datasets. Moreover, the model may further develop pre-existing biases and discriminatory views, which may persist negatively in future predictions. In this respect, we considered the following three experimental scenarios to analyze the effectiveness of our proposed model.

- Basically, we assumed a well-distributed data environment as an ideal scenario. In this scenario, we distributed data uniformly for each class at each node. That is, in each sub-dataset grouped into a specific class, the dataset was randomly divided into equal-size pieces (divide the size of the sub-dataset by the number of nodes) and distributed to each node.

- For comparison with the ideal scenario, we considered a data imbalance scenario. Specifically, we configured an environment with different amounts of data at each local node. For this scenario, we randomly assigned a ratio of the data to each local node from the open interval $(0, 1)$ (naturally, we randomly sampled each data point, and set the total ratio to be 1).

- Finally, we considered a skewness scenario. In particular, we assumed that the data collected from each local node was skewed towards a specific class. In the skewness scenario, we configured that there are 9 local nodes, and that each node has only one classes: a benign or a single attack class. Note that, this scenario implies a data imbalance scenario because the original dataset had an imbalance between classes.

C. Experiment result

We divided the entire dataset into two sub-datasets and distinguished them by training and test datasets. In particular, we divided the data randomly at a ratio of 7: 3 in each class for fair evaluation. The total size of the training and test dataset is 851,123 and 364,767, respectively. In the experiment, we first distributed the data to each local node within the training dataset according to the scenarios described in the previous section, and repeated 10 experiments on each model and presented the result of the best performance. To evaluate the performance, we utilized four metrics: $Accuracy$, $Precision$, $Recall$, and $F1$-score. $Accuracy$ denotes the percentage of correctly predicted results and is one of the most common metrics. $Precision$ refers to the fraction of positive values predicted by the trained model that are correct, and $Recall$ denotes the fraction of positive data that are correctly predicted by the trained model. $F1$-score is a metric that combines $Precision$ and $Recall$.

Table 2 presents the experimental results for the multi-classification task, where the former value represents the result of the DNN model and the latter value represents the result of the CNN model (based on the slash symbol). Note that, the results for the baseline shown in the table represent experimental results for models trained in a centralized environment. From the model point of view, the 1-D CNN model converged more stably during the training process in our experiment, but there was no significant difference in performance between the DNN model and the 1-D CNN model. Overall, very high precision and recall values were measured in each class, and relatively low precision results were observed in UDP Flood.

For experiments in the centralized scenario (baseline), the system showed an accuracy of up to 98.32% for both models. In the ideal scenario, the models performed similarly to the baseline models, with an accuracy of up to 98.32%. This means that, in a distributed environment where the data occurs uniformly within the population, split learning approaches can

have similar performance to centralized models. These results can be analyzed as experimental results that demonstrate the aforementioned consistency property of split learning. In the case of the data imbalance scenario, the models showed that the performance of intrusion detection was sufficiently preserved when compared to the baseline and the ideal models (we observed accuracy of 98.24% and 98.26% for DNN and CNN models, respectively). Specifically, we observed that the loss increases dramatically while learning on a particular node with a very small amount of data, but other nodes with sufficient data seemed to solve these problems immediately. As a scenario considering the worst-case environment, we assumed a skewness scenario where the data at each node is biased towards a specific class. Although the models in this scenario were measured to have relatively low accuracy compared to the other scenarios, we always measured the models to be more than 96% accurate. In the classification results for each class, we found that the models misclassified slightly in the Benign and UDP Flood classes, but showed a high overall detection rate in the other classes. Through these experiments, we confirmed that the split learning approach can be applied efficiently to detect network intrusion in a distributed mobile environment.

VI. Conclusion

In this study, we propose a distributed learning-based network intrusion detection system that can train data in a decentralized environment while taking into account the computing power of each local node and preserving privacy. In particular, we leveraged a state-of-the-art learning system, split learning, and combined it with deep learning models, DNN and CNN, to build intrusion detection systems. Furthermore, the proposed system was evaluated in various scenario based on a dataset collected in real 5G environments. Through experiments, we showed that the proposed models can achieve sufficient performance for network threat detection. Specifically, we demonstrated that the models perform similarly to those in centralized environments in a data imbalanced environment, and that detection performance can be preserved even in extreme scenarios. As future work, we will focus on applying generative models to improve the system in data imbalanced and skewed scenarios. In addition, it would be an interesting study to implement robust distributed NIDS models that can resist adversarial attacks that can bypass NIDS.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-00796, Research on Foundational Technologies for 6GAutonomous Security-by-Design to Guarantee Constant Quality of Security)

References

[1] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in Proc. Int. Conf. Signal Process. Commun. Eng. Syst., Andhra Pradesh, India, Jan. 2015, pp. 92–96.

[2] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in Proc. IEEE 15th Int. Conf. Mach. Learn. Appl. (ICMLA), Anaheim, CA, USA, 2016, pp. 195–200.

[3] Y. Imamverdiyev and F. Abdullayeva, "Deep learning method for denial of service attack detection based on restricted Boltzmann machine," Big Data, vol. 6, no. 2, pp. 159–169, Jun. 2018.

[4] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," IEEE Access, vol. 5, 2017.

[5] J. Gao et al., "Omni SCADA intrusion detection using deep learning algorithms," IEEE Internet of Things J., vol. 8, no. 2, Jan. 2021.

[6] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," IEEE Trans. Emerg. Topics Comput. Intell., vol. 2, no. 1, pp. 41–50, Feb. 2018.

[7] C. Ieracitano, A. Adeel, F. C. Morabito, and A. Hussain, "A novel statistical analysis and autoencoder driven intelligent intrusion detection approach," Neurocomputing, vol. 387, pp. 51–62, Apr. 2020

[8] G. Dlamini and M. Fahim, "DGM: A data generative model to improve minority class presence in anomaly detection domain," Neural Comput. Appl., vol. 33, pp. 13635–13646, Apr. 2021.

[9] I. Siniosoglou, P. Radoglou-Grammatikis, G. Efstathopoulos, P. Fouliras, and P. Sarigiannidis, "A unified deep learning anomaly detection and classification approach for smart grid environments," IEEE Trans. Netw. Service Manage., vol. 18, no. 2, pp. 1137–1151, Jun. 2021.

[10] C. Park, J. Lee, Y. Kim, J. G. Park, H. Kim, and D. Hong, "An enhanced AI-based Network Intrusion Detection System using Generative Adversarial Networks," IEEE Internet of Things J., vol. 10, no. 2, 2022.

[11] X. Hei, X. Yin, Y. Wang, J. Ren, and L. Zhu, "A trusted feature aggregator federated learning for distributed malicious attack detection," Comput. Secur. vol. 99, Dec. 2020.

[12] S.A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things intrusion detection: Centralized, on-device, or federated learning?" IEEE Netw. vol. 34 no. 6, pp. 310–317, Sep. 2020.

[13] A.K. Chathoth, A. Jagannatha, and S. Lee, "Federated intrusion detection for IoT with heterogeneous Cohort privacy," ArXiv:2101.09878v1.

[14] Q. Qin, K. Poularakis, K.K. Leung, and L. Tassiulas, "Line-speed and scalable intrusion detection at the network edge via federated learning," 2020 IFIP Networking Conference (Networking), Jun. 2020.

[15] T. Huong, P.B. Ta, D. Long, B. Thang, N. Binh, T. Luong, and K.P. TRAN, "LocKedge: Low-complexity cyberattack detection in IoT edge computing," IEEE Access, vol. 9, pp. 29696 - 29710. Feb. 2021.

[16] V. Rey, P.M.S. Sánchez, A.H. Celdrán, G. Bovet, and M. Jaggi, "Federated learning for malware detection in IoT devices," arXiv preprint arXiv:2104.09994, 2021.

[17] J. Li, L. Lyu, X. Liu, X. Zhang, and X. Lyu, "FLEAM: A federated learning empowered architecture to mitigate DDoS in industrial IoT," ArXiv:2012.06150, 2020.

[18] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A. R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," IEEE 39th International Conference on Distributed Computing Systems, pp. 756–767, 2019.

[19] T.V. Khoa, Y.M. Saputra, D.T. Hoang, N.L. Trung, D. Nguyen, N.V. Ha, and E. Dutkiewicz, "Collaborative learning model for cyberattack detection systems in IoT industry 4.0," IEEE Wireless Communications and Networking Conference, pp. 1–6, 2020.

[20] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for IntrusionDetection in industrial cyber-physical systems," IEEE Transactions on Industrial Informatics, vol. 17, no. 8, pp. 5615 - 5624, Aug. 2020.

[21] S. Jayasinghe, Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "Federated learning based anomaly detection as an enabler for securing network and service management automation in beyond 5g networks," in 2022 Joint European Conference on Networks and Communications & 6G Summit, pp. 345-350. Jun. 2022.

[22] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," Journal of Network and Computer Applications, vol. 116, no. 15, pp. 1-8, Aug. 2018.

[23] S. Samarakoon, et. al. "5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network," arXiv preprint arXiv:2212.01298.

[24] Samsung Research. The next hyper connected experience for all, 2020.