# CAFNet: Compressed Autoencoder-based Federated Network for Anomaly Detection

Abu Saleh Md Tayeen*, Satyajayant Misra*, Huiping Cao*, Jayashree Harikumar†
*New Mexico State University, Las Cruces, New Mexico, USA
Email: {tayeen, misra, hcao}@nmsu.edu
†DEVCOM Analysis Center, WSMR, New Mexico, USA
Email: jayashree.harikumar.civ@army.mil

MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM) | 979-8-3503-2181-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/MILCOM58377.2023.10356377

***Abstract*—Federated Learning (FL) is a promising collaborative training paradigm that utilizes decentralized on-device data. Using supervised learning approaches in FL-based network intrusion detection systems often leads to poor classification performance because of the highly imbalanced data with limited labeled network traffic anomalies from data collected by edge devices. Furthermore, detecting zero-day anomalies/attacks without *a priori* knowledge is difficult. Due to these constraints, unsupervised learning-based methods, such as autoencoders, which only use benign traffic to build the detection model, appear to be the desired choice to identify anomalous network traffic. In this work, we propose a Compressed Autoencoder-based Federated Network (*CAFNet*) framework for network anomaly detection to deal with the labeled data scarcity issue while preserving data owner's privacy and reducing communication overhead. Our framework leverages the latent representation of autoencoders to capture important information in the input features of the distributed network devices and eliminate the transmission of redundant information (weights) during federated training. Our extensive experimental results with three publicly available network intrusion detection datasets show that our proposed framework can significantly lower communication cost up to 65% of the state-of-the-art model compression strategies used in traditional FL as well as achieves attack detection performance comparable to conventional FL framework.**

***Index Terms*—Network anomaly detection, federated learning, autoencoder.**

## I. Introduction

With an increased number of cyber attacks on networks, it has become crucial to find efficient solutions to detect such attacks or anomalies. In recent years, various machine learning (ML) algorithms have been applied to the network security domain for building models to effectively detect network attacks [1]. However, such ML-based solutions are often designed on the unseemly assumption that there is an ample amount of data available in a central host to train the model and such data can be collected from network devices without any privacy concerns. These conventional schemes also consume high network bandwidth and incur long transmission latency due to the explosion of big data generated by distributed network devices.

Researchers [2]–[6] have attempted to address these issues by proposing network Intrusion Detection System (IDS) using Federated Learning (FL) [7]. FL is a collaborative training paradigm that can build a model without centralizing raw data from distributed devices and preserve their privacy. Most of these FL-based IDSes use supervised learning approaches which rely on well-labeled, sufficient, and balanced datasets (*i.e.,* similar number of benign and attack traffic samples) to achieve high detection accuracy. However, in practice, it is expensive, time-consuming, and very difficult to collect and label attack/anomalous traffic data. In addition, devices participating in an FL environment may not experience all types of attacks or may encounter no intrusions at all depending on their location in the network. Therefore, the training set of these devices becomes highly imbalanced resulting in poor performance of the detection models. Moreover, zero-day (*i.e.,* novel) attacks constantly occur and it is hard for any supervised approach to guarantee its detection performance on new/unknown attacks that do not exist in the training dataset. To address this issue, a number of unsupervised Autoencoder (AE)-based approaches have been proposed. These approaches use only benign data during the training for network anomaly detection [8]–[12]. However, these methods are either under a centralized framework which poses the risk of privacy leakage [8]–[10] or do not deal with the communication cost in FL [11], [12].

**Motivation:** Motivated by the above challenges, in this paper, we propose and design an AE-based unsupervised learning method to build a detection model using only benign data, in combination with the FL framework. The newly proposed method is called Compressed Autoencoder-based Federated Network (*CAFNet*) framework for network anomaly detection. Our framework addresses the dataset imbalance issue by waiving the requirement of collecting or labeling attack traffic data for training as well as taking advantage of the privacy preservation and communication reduction properties of FL. To further improve communication efficiency, we capitalize on the latent space representation characteristics of the autoencoders to perform model compression in our framework. The main **contributions** of our work are summarized as follows.

1) We propose a framework to detect anomalous traffic utilizing autoencoders and FL for a networking environment where devices only contain benign data for training.

2) We introduce a new way of reducing communication costs in FL by exploiting the intrinsic characteristics of autoencoders.
3) We conduct experiments on three representative datasets and show that the anomaly detection performance and the communication efficiency achieved by our proposed method is better than the baseline methods.

The remainder of the paper is organized as follows. In Section II, we describe the background knowledge and related works. We present the proposed framework in Section III. In Section IV, we evaluate and discuss the experimental results. Finally, we conclude in Section V.

## II. Background and Related Work

### A. Autoencoder (AE)

An Autoencoder (AE) is one of the classical artificial neural networks which has emerged as a suitable approach to anomaly detection [13]. An AE is considered an unsupervised learning technique as it does not require any class labels to train. The training of the AE is performed by feeding the model with only normal/benign observations. A simple illustration of the architecture of an autoencoder model is shown in Figure 1.

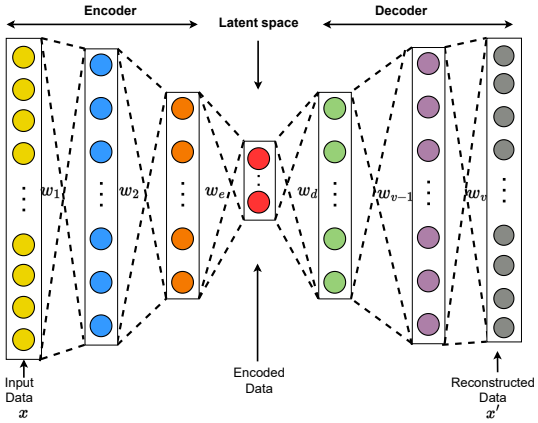


Fig. 1: Architecture of an Autoencoder

In practice, a generic autoencoder consists of two parts: the encoder and the decoder. The encoder and the decoder parts have a symmetrical structure from the middle hidden layer, sometimes called the bottleneck layer. The task of the encoder is to map an input sample, $x_i$, with $q$ dimensions from the input layer to a latent representation, $z_i$, with $q'$ dimensions in the bottleneck layer, where $q > q'$ in general. It compresses the input data to preserve non-redundant information in the latent representation. The task of the decoder is to map the latent representation $z_i$ back into the input space. It attempts to reconstruct input $x_i$ from the latent space by producing $x'_i$ at the output layer.

Let us denote the weight matrices of the hidden layers that connect the encoder and the decoder to the bottleneck layer as $w_e \in \mathbb{R}^{h \times q'}$ and $w_d \in \mathbb{R}^{q' \times h}$ respectively, where $h$ is the number of neurons in the hidden layer immediately before/after the bottleneck layer. Since these parameters, $\{w_e, w_d\}$, of the AE model involve direct connection to the latent space in the bottleneck layer, we name them as *latent representation parameters*. The AE model learns to minimize a loss function with respect to the parameters, $W = (w_1, \cdots w_e, w_d, \cdots w_v)$ using a stochastic gradient descent based training algorithm. Generally, Mean Square Error (MSE) defined in Equation 1 is used as a loss function, $\mathcal{L}$.

$$\mathcal{L}(W; \mathbb{X}) = \frac{1}{n} \sum_{i=1}^{n} (\|x'_i - x_i)\|^2) \tag{1}$$

, where $\mathbb{X} = \{x_1, x_2, \cdots x_n\}$ is the training dataset and $\|x'_i - x_i\|$ is the reconstruction error, $r$.

### B. FL-based IDS

There exist several research works that have utilized FL to design network IDS. Researchers in [2], [3] evaluated a FL-based IDS for IoT devices with NSL-KDD dataset under various real-world scenarios to show that it obtains accuracy comparable to the centralized approach. Zhang *et al.* [4] proposed a DDoS attack detection model based on FL. They used K-means clustering during aggregation and applied a data re-sampling algorithm on the client side to solve the data imbalance problem. Mothukuri *et al.* [5] proposed a FL-based IoT network anomaly detection approach that uses seven gated recurrent unit models, each for a separate window size of the input and a random forest ensembler to combine the predictions. Weinger *et al.* [6] showed the performance degradation problem of traditional FL-based anomaly detection in the context of IoT networks due to class imbalance issue. They proposed five data augmentation schemes to solve this issue.

All of these works employ supervised learning methods to build attack detection models. However, these methods fail to address the data imbalance problem due to the lack of sufficient labeled data and can not provide assurance of high accuracy on unknown attacks. Though few works combined FL and AE-based methods and proposed network IDSes [11], [12] to address these issues, they differ from ours because they do not focus on reducing the communication cost in FL. Our paper's main contribution lies in the exploitation of the AE attributes to improve the communication efficiency in a FL-based network anomaly detection framework.

## III. Design of *CAFNet* Framework

### A. Collaborative Training:

*CAFNet* collaboratively trains an AE model utilizing decentralized benign traffic data from multiple network devices. In particular, this framework consists of two components:

*1) Central Node (CN):* The central node is generally a centralized server in the network with rich storage and computing resources. It sends a global model to all participating nodes and acts as an aggregator of local model updates from them.

*2) Edge Node (EN):* The edge nodes are routers or network security gateways that are equipped with storage and computing power. The ENs use their local traffic data to train the global model and then upload the updated model parameters to the CN. The edge nodes act as clients and can use their local AE models to detect anomalous network traffic. Our framework is illustrated in Figure 2. The collaborative training

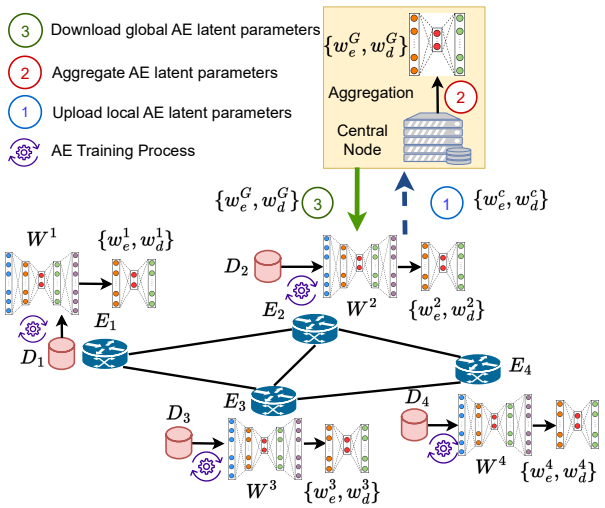


Fig. 2: *CAFNet* framework

algorithm of our framework is given in Algorithm 1 with the code executed at the CN and ENs presented together. In Algorithm 1, $W_t^G$ denotes the global AE model at the $t$-th round, $W^c$ denotes the local AE model of the $c$-th EN, $\{w_{e_t}^G, w_{d_t}^G\}$ are the *latent representation parameters* of the global AE model at the $t$-th round, and $\{w_{e_t}^c, w_{d_t}^c\}$ are the *latent representation parameters* of the local AE model of the $c$-th EN at the $t$-th round. We describe the different phases of this algorithm in detail below.

*1) **Initialization**:* The CN starts the training of *CAFNet* by initiating execution of *CN-Main* function in Step 1. In Step 2 of Algorithm 1, the CN generates the global AE model, $W_0^G$, and initializes its parameters with random values.

*2) **Updating of the global model**:* In each communication round, $t$, the CN first retrieves the *latent representation parameters*, $w_{e_t}^G$ and $w_{d_t}^G$ (defined in subsection II-A) from the current global AE model, $W_t^G$, in Step 4. In Steps 5 to 16 of Algorithm 1, the CN sends these parameters to each of the $|C|$ ENs participating in the training process. After receiving the updated parameters $\{w_{e_{t+1}}^c, w_{d_{t+1}}^c\}$ of the local AE model for all $c \in C$ for communication round $t+1$, the CN aggregates these parameters in Step 7 using *FedAvg* [7]. Finally, in Step 8 of Algorithm 1, the CN updates its *latent representation parameters* of the current global AE model, $W_t^G$ and generates a new global AE model, $W_{t+1}^G$, for the next round, $t+1$.

*3) **Updating of local models**:* Each EN receives the *latent representation parameters*, $w_e^G$ and $w_d^G$, of the global AE model from the CN. As shown in the *EN-Update* function of Algorithm 1, each EN first retrieves the *latent representation parameters*, $\{w_e^c, w_d^c\}$, of its local AE model $W^c$ in Step 10. Second, it replaces these parameters with $w_e^G$ and $w_d^G$ in Step 11 and updates its local AE model $W^c$ with the modified *latent representation parameters* in Step 12. Third, in Steps 13 to 15, each EN trains its local AE model, $W^c$ using mini-batches from its local dataset with a learning rate, $\eta$. Finally, each EN uploads the *latent representation parameters*, $\{w_e^c, w_d^c\}$ of its updated AE model, $W^c$ to the CN in Steps 16 and 17.

**Algorithm 1:** *CAFNet* framework training

1 **CN-Main:**
2 Initialize $W_0^G$
3 **foreach** *round* $t = 0, 1, \cdots$ **do**
4 Get $\{w_{e_t}^G, w_{d_t}^G\}$ from $W_t^G$
5 **foreach** *client* $c \in C$ **do**
6 $\{w_{e_{t+1}}^c, w_{d_{t+1}}^c\} \leftarrow$ `EN-Update`$(c, \{w_{e_t}^G, w_{d_t}^G\})$
7 $\{w_{e_{t+1}}^G, w_{d_{t+1}}^G\} \leftarrow$ `FedAvg`$(\{w_{e_{t+1}}^c, w_{d_{t+1}}^c\}_{c \in C})$
8 $W_{t+1}^G \leftarrow W_t^G \cup \{w_{e_{t+1}}^G, w_{d_{t+1}}^G\}$
9 **EN-Update**$(c, \{w_e^G, w_d^G\})$**:**
10 Get $\{w_e^c, w_d^c\}$ from $W^c$
11 $\{w_e^c, w_d^c\} \leftarrow \{w_e^G, w_d^G\}$
12 $W^c \leftarrow W^c \cup \{w_e^c, w_d^c\}$
13 **foreach** *local epoch* $j$ *from* 1 *to* $E$ **do**
14 **foreach** *batch* $b$ *of size* $B$ **do**
15 $W^c \leftarrow W^c - \eta \bigtriangledown \mathcal{L}(W^c, b)$
16 Get $\{w_e^c, w_d^c\}$ from $W^c$
17 **return** $\{w_e^c, w_d^c\}$

## B. Anomaly Detection

AE-based anomaly detection is accomplished using the reconstruction error, $r$, as the anomaly score. An AE model trained on only normal/benign data learns the data distribution of normal network traffic behavior. It is able to successfully reconstruct the samples that are similar to those in the training dataset resulting in small $r$ values. However, when an anomalous sample is given to the trained AE model, it reconstructs the sample poorly as the model has not seen samples similar to that during the training. As a result, anomalies produce large $r$ values. A fixed threshold value, $\delta$, is obtained from the reconstruction errors of the training data and is used as a decision boundary for detecting anomalous data. The samples that generate $r$ values greater than $\delta$ are classified as anomalous, whereas the ones with $r$ values less than or equal to $\delta$ are classified as normal (benign) samples.

After training is completed, the ENs find out the value of the threshold, $\delta$, and utilize it to decide whether a traffic sample is anomalous or not during the testing stage. To determine the threshold, $\delta$, each EN feeds the network traffic samples from their local training dataset to their trained AE model and calculates reconstruction errors of all samples. The mean value

is calculated from all the reconstruction errors and marked as the threshold value. Thus, each EN has its own threshold.

## IV. Evaluation

### A. Datasets

We evaluated the performance of the *CAFNet* framework on three public network intrusion detection datasets, namely CICDDoS2019 [14], Bot-IoT [15], and UNSW-NB15 [16]. Since *CICDDoS2019* and *Bot-IoT* datasets contain millions of records and *UNSW-NB15* has more attack flow instances than benign flow instances, we applied random undersampling and oversampling methods to get a sample of these datasets.

*1) CICDDoS2019:* This dataset [14] contains over millions of realistic DDoS attack samples in 12 different categories such as DNS, MSSQL, NetBIOS, SNMP, SSDP, and UDP. We took a sample of 250,000 benign flow instances and 250,000 attack flow instances from this dataset for our experiments.

*2) Bot-IoT:* This dataset [15] includes normal traffic flows as well as DoS and DDoS attack flows generated by botnets for different IoT devices, such as smart fridge and smart lights. We chose a sample of 300,000 benign flow instances and 200,000 attack flow instances from this dataset for our experiments.

*3) UNSW-NB15:* In this dataset, Moustafa *et al.* [16] implemented various attack traffic categories such as Backdoors, DoS, Shell-code, and Worms. In our experiments, we used a sample of 60,000 benign flow instances and 40,000 attack flow instances from this dataset.

### B. Experiment Setup

*1) Preprocessing datasets:* To preprocess the datasets, we utilized the following pipeline. We first cleaned it up by removing feature values containing NaN (Not a Number), blanks, and infinity. Second, we removed socket-based features such as 'Flow ID', 'Source IP', etc. to allow the model to learn from the characteristics of the flows itself. Moreover, we removed features with zero variance, *i.e.,* features with constant values and duplicate features. Third, we performed encoding (*i.e.,* convert to float/numeric values) of the categorical features. Fourth, we applied *min-max*-based normalization on the features to eliminate the impact of large variance of the feature values and thus reducing model training time and improving model accuracy. After data preprocessing, we are left with 41, 65 and 59 features for the *CICDDoS2019*, *Bot-IoT*, and *UNSW-NB15* datasets respectively.

*2) Implementation details:* We utilized the *PyTorch* [17] Python library to implement the AE models and the *CAFNet* framework. In our experiments, we applied the stratified $K$-Fold technique to the sampled datasets to perform $K$-fold cross-validation, where one fold is used for testing and the rest are used for training. In our case, $K = 5$. We distributed the training dataset over ten ENs in a network with different proportions, *i.e.,* each EN does not have equal number of training data instances. We also equally divided the testing data among all the ENs.

In our experiments, we used two AE model architectures: *AE1* and *AE2*. Since *UNSW-NB15* dataset is much smaller in size compared to the other datasets, we used a lighter (*i.e.,* less number of hidden layers) AE model architecture, *AE1* only for this dataset to prevent overfitting. In *AE1*, the encoder consists of one hidden layer with 32 neurons and a bottleneck layer with 8 neurons. In *AE2*, the encoder consists of two hidden layers with 32 and 16 neurons respectively and a bottleneck layer with 8 neurons. The decoder parts of the AE models have a symmetrical architecture from the bottleneck layer. We applied Rectified Linear Units (ReLU) activation function on the output of all hidden layers. We used Adam optimizer with learning rate, $\eta = 0.001$ and the Mean Square Error (MSE) as the loss function to train the AE models.

*3) Metrics:* To evaluate the anomaly detection performance of our framework, we used metrics including Accuracy ($Acc$), Precision ($Pre$), Recall ($Rec$), and F1-score ($F1$). Let us denote the number of instances correctly classified as an anomaly/attack with $TP$ (True Positive), the number of instances correctly classified as normal/benign with $TN$ (True Negative), the number of instances incorrectly classified as an anomaly/attack with $FP$ (False Positive), and the number of instances incorrectly classified as normal with $FN$ (False Negative). Accuracy ($Acc$) is the percentage of correctly classified samples, *i.e.,* $Acc = (TP + TN)/(TP + TN + FP + FN)$. Precision ($Pre$) is the ratio of correctly detected attack samples to all detected attack samples, *i.e.,* $Pre = TP/(TP + FP)$. Recall ($Rec$) represents the percentage of correctly classified attack samples, *i.e.,* $Rec = TP/(TP + FN)$ and F1-Score ($F1$) is the harmonic mean of Precision and Recall, *i.e.,* $F1 = 2 \times (Pre \times Rec)/(Pre + Rec)$. To measure the communication efficiency of the the *CAFNet*, we defined the metric for ENs to CN communication cost, $\Gamma = \left(\sum_{i=1}^{|W|} T_i\right) \times |C|$, where $T_i$ is the amount of bits/bytes required to communicate the $i$-th model parameter of the AE model, $W$, and $|C|$ is the number of ENs.

*4) Threshold ($\delta$) selection:* Selecting the threshold, $\delta$ to determine whether a testing sample is anomalous or not is a very important step. As mentioned in Section III-B, each EN computes its own $\delta$ value using the reconstruction errors of its local training dataset. Usually classical statistical methods such as mean and maximum are applied on these reconstruction errors to choose the threshold [9], [10]. In our experiments, we tested with statistical measures including maximum, minimum, mean, median, and different percentiles to find a best-performing threshold value. We found that choosing mean of the reconstruction errors as $\delta$ gives the best trade-off between precision and recall performance metrics for the test dataset.

### C. Baseline Methods

To evaluate our proposed method, *CAFNet*, we compare it with the following baselines.

*1) Non-federated:* In this scenario, all ENs train their AE models with their local dataset without the help of Federated

TABLE I: Performance comparison of all methods on different datasets

| Method | CICDDoS2019 | | | Bot-IoT | | | UNSW-NB15 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pre | Rec | F1 | Pre | Rec | F1 | Pre | Rec | F1 |
| Non-federated | 0.94 ± 0.002 | 0.94 ± 0.017 | 0.94 ± 0.009 | 0.93 ± 0.003 | 0.85 ± 0.026 | 0.88 ± 0.015 | 0.87 ± 0.001 | 0.97 ± 0.003 | 0.92 ± 0.001 |
| Vanilla | 0.93 ± 0.001 | 0.94 ± 0.026 | 0.94 ± 0.013 | 0.93 ± 0.002 | 0.85 ± 0.030 | 0.89 ± 0.017 | 0.87 ± 0.004 | 0.97 ± 0.007 | 0.92 ± 0.002 |
| RM-10 | 0.90 ± 0.003 | 0.86 ± 0.017 | 0.88 ± 0.011 | 0.82 ± 0.037 | 0.57 ± 0.042 | 0.66 ± 0.045 | 0.86 ± 0.002 | 0.93 ± 0.012 | 0.89 ± 0.006 |
| RM-5 | 0.92 ± 0.004 | 0.89 ± 0.022 | 0.90 ± 0.013 | 0.85 ± 0.046 | 0.68 ± 0.073 | 0.74 ± 0.068 | 0.86 ± 0.002 | 0.95 ± 0.015 | 0.91 ± 0.008 |
| PQ-4 | 0.80 ± 0.028 | 0.54 ± 0.027 | 0.62 ± 0.017 | 0.73 ± 0.079 | 0.47 ± 0.149 | 0.55 ± 0.143 | 0.86 ± 0.007 | 0.91 ± 0.023 | 0.89 ± 0.015 |
| PQ-8 | 0.92 ± 0.005 | 0.78 ± 0.019 | 0.84 ± 0.011 | 0.82 ± 0.026 | 0.55 ± 0.059 | 0.64 ± 0.049 | 0.87 ± 0.003 | 0.96 ± 0.004 | 0.91 ± 0.001 |
| PQ-16 | 0.93 ± 0.003 | 0.90 ± 0.033 | 0.91 ± 0.018 | 0.90 ± 0.007 | 0.70 ± 0.019 | 0.78 ± 0.013 | 0.87 ± 0.004 | 0.97 ± 0.011 | 0.92 ± 0.004 |
| ***CAFNet*** | **0.93 ± 0.003** | **0.92 ± 0.012** | **0.93 ± 0.007** | **0.92 ± 0.005** | **0.80 ± 0.045** | **0.85 ± 0.028** | **0.87 ± 0.001** | **0.95 ± 0.006** | **0.91 ± 0.003** |

learning (FL). It is similar to a centralized scenario where each EN is trained separately in a centralized way.

*2) Vanilla FedAvg:* In this scenario, a joint AE model is trained at CN by applying *FedAvg* [7] algorithm to the model parameters received from the ENs. The joint model is then shared with all ENs. This simulates the basic FL scenario without implementing any model compression. As a result, the cost of communicating a model parameter with dimension, $m$ will be $m \times u$, where $u$ is the number of bits required to transmit a real value. We refer to this method as *Vanilla*.

*3) Random mask FedAvg:* This scenario simulates a FL setup that applies *FedAvg* [7] algorithm to the local AE models that are compressed by a *structured update* strategy [18]. We chose a random mask structure for each EN to limit its model updates by setting a certain percentage, $p$ of weight elements to be zero. As a result, the cost of communicating a model parameter with dimension, $m$ will be $(m - \frac{p}{100} \times m) \times u$, where $u$ is the number of bits required to transmit a real value. In this case, higher percentage ($p$) value represents higher compression. We ran our experiments using random masks for all EN models to remove 5% or 10% of the random elements from their weight matrices (model parameters). When $p = 5$, we denote the *Random mask FedAvg* method as *RM-5* and when $p = 10$, we denote it as *RM-10* in our results.

*4) Probabilistic quantization FedAvg:* This scenario is similar to the scenario defined in subsection IV-C3 except that the local AE models are compressed by a *sketched update* strategy [18]. We chose to compress the local AE model updates by probabilistically distributing weights into encoded buckets. For example, consider a vector $V = (v_1, v_2, \cdots v_m)$ representing a model parameter. Let $v_{max} = \max_{1 \leq j \leq m}(v_j)$ and $v_{min} = \min_{1 \leq j \leq m}(v_j)$. The 1-bit or 2-level quantized (compressed) vector, $V'$ of $V$ is generated as follows.

$$v'_j = \begin{cases} v_{max}, & \text{with probability } \frac{v_j - v_{min}}{v_{max} - v_{min}} \\ v_{min}, & \text{with probability } \frac{v_{max} - v_j}{v_{max} - v_{min}} \end{cases} \quad (2)$$

In this case, each EN can encode the vector $V'$ into a bit vector such that $v'_j = 1$ if $v'_j = v_{max}$ and 0 otherwise. Thus, instead of sending $V$ to CN, each EN can only transmit this bit vector and two real values, $v_{max}$ and $v_{min}$. Therefore, if $k$-level quantization is used, the cost of communicating a model parameter with dimension, $m$ will be $(m \times \lceil \log_2 k \rceil) + (2 \times u)$ where $u$ is the number of bits required to transmit a real value. The less the number of levels used for quantization, the more compressed the model will be. In our experiments, we used three different quantization levels where $k \in \{4, 8, 16\}$. We denote *Probabilistic quantization FedAvg* method for each of the $k$ values as *PQ-4*, *PQ-8*, and *PQ-16* in our results.

### D. Results and Discussion

We evaluate our proposed framework, *CAFNet* on two aspects: *model performance* and *communication efficiency*. First, we compare the anomaly detection performance of *CAFNet* with the baseline methods using metrics $Pre$, $Rec$, and $F1$. Second, we compare the communication cost of *CAFNet* and the baseline methods.

*1) Model performance:* In Table I, we present the overall performance of all the methods on the three datasets. In our experiments, we perform 5-fold cross validation and compute the $Pre$, $Rec$, and $F1$ achieved by the local AE models on their test datasets and show the results in $M \pm S$ form where $M$ and $S$ represents mean and standard deviation respectively.

The experimental results in Table I show that our *CAFNet* framework outperforms all the baselines except Vanilla. Compared with the *Vanilla* method, our framework is only 1%, 4%, and 1% lower on F1-score for datasets *CICDDoS2019*, *Bot-IoT*, and *UNSW-NB15* respectively. Since *CAFNet* only communicates the *latent representation parameters* of the local AE models, some encapsulated information encoded in the other parameters gets lost, and thus causes a dip in the F1-score. In terms of recall, *CAFNet* obtains 2% and 14% higher performance compared to the *PQ-16* method and 3% and 18% higher performance compared to the *RM-5* method in detecting attacks for *CICDDoS2019* and *Bot-IoT* datasets respectively.

*2) Communication efficiency:* The major benefit of our *CAFNet* framework is that it significantly reduces the communication overhead when each EN transmits model parameters to the CN and vice versa. That is because each EN and CN only sends the *latent representation parameters* instead of all the parameters of the model. To demonstrate the communication efficiency of *CAFNet*, we compute the cost of EN to CN communication required for all ENs in each federated

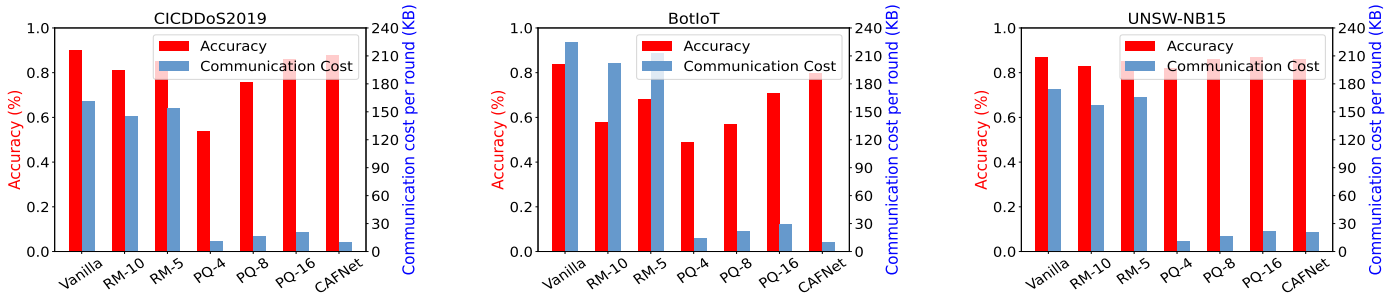


Fig. 3: Comparison of accuracy and communication cost of all methods on different datasets.

method. In Figure 3, we compare, the communication cost, Γ, per communication round during training and the average model accuracy of all methods on the three datasets used in the experiments during testing.

Figure 3 also shows that our proposed method, *CAFNet* achieves 2%, 14%, and 3% higher accuracy than the *PQ-16* method with 9%, 62%, and 50% lower communication cost for *UNSW-NB15*, *Bot-IoT*, and *CICDDoS2019* datasets respectively. Moreover, as shown in Figure 3, our method achieves 88% and 80% accuracy for datasets *CICDDoS2019* and *Bot-IoT* respectively which are very close to the accuracy of the *Vanilla* method while requiring 88% and 95% less communication cost than *Vanilla*. Thus, these results demonstrate the trade-off between the model accuracy and communication cost of our *CAFNet* method, which support compression of data.

## V. Conclusions

In this paper, we propose *CAFNet*, a communication efficient FL-based framework, for network anomaly detection. *CAFNet* adopts the unsupervised learning technique of autoencoders to build anomalous traffic detection models by training them using only on-device benign data from distributed edge devices. A distinct characteristic of *CAFNet* is that it employs the inherent latent space property of autoencoders to shrink the model size, which in fact reduces the communication cost up to 95%. In the future, we plan to explore different variants of autoencoders to improve the performance of our framework.

## Acknowledgment

This work was sponsored by the DEVCOM Analysis Center and was accomplished under Cooperative Agreement Number W911NF-22-2-0001. This work was also partially funded by the U.S. Department of Energy, DE-SC0023392.

## References

[1] J. Lansky, S. Ali, M. Mohammadi, M. K. Majeed, S. H. T. Karim, and et al., "Deep learning-based intrusion detection systems: a systematic review," *IEEE Access*, vol. 9, pp. 101 574–101 599, 2021.

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

[3] S. Kim, H. Cai, C. Hua, and et al., "Collaborative anomaly detection for internet of things based on federated learning," in *2020 IEEE/CIC Int'l Conf. on Communications in China (ICCC)*, 2020, pp. 623–628.

[4] J. Zhang, P. Yu, L. Qi, S. Liu, H. Zhang, and J. Zhang, "FLDDoS: DDoS attack detection model based on federated learning," in *20th IEEE Int'l Conf. on Trust, Security and Privacy in Computing and Communication (TrustCom)*, 2021, pp. 635–642.

[5] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, and et al., "Federated-learning-based anomaly detection for IoT security attacks," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2545–2554, 2021.

[6] B. Weinger, J. Kim, A. Sim, M. Nakashima, N. Moustafa, and K. J. Wu, "Enhancing IoT anomaly detection performance for federated learning," *Digital Communications and Networks*, vol. 8, no. 3, pp. 314–323, 2022.

[7] B. McMahan, E. Moore, D. Ramage, and et al., "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[8] S. Zavrak and M. İskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108 346–108 358, 2020.

[9] K. Yang, J. Zhang, Y. Xu, and J. Chao, "DDoS attacks detection with autoencoder," in *IEEE/IFIP NOMS*. IEEE, 2020, pp. 1–9.

[10] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, and F. Sabrina, "Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset," *IEEE Access*, vol. 9, pp. 140 136–140 146, 2021.

[11] Z. Wang, P. Wang, and Z. Sun, "SDN traffic anomaly detection method based on convolutional autoencoder and federated learning," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 4154–4160.

[12] G. de Carvalho Bertoli, L. A. P. Junior, O. Saotome, and A. L. dos Santos, "Generalizing intrusion detection for heterogeneous networks: A stacked-unsupervised federated learning approach," *Computers & Security*, vol. 127, p. 103106, 2023.

[13] R. Bhatia, S. Benno, J. Esteban, T. Lakshman, and J. Grogan, "Unsupervised machine learning for network-centric anomaly detection in IoT," in *Proc. of the 3rd ACM CONEXT workshop on Big-DAMA for Data Communication Networks*, 2019, pp. 42–48.

[14] I. Sharafaldin, A. H. Lashkari, S. Hakak, and et al., "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Int'l Carnahan Conf. on Security Technology*. IEEE, 2019, pp. 1–8.

[15] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[16] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *MilCIS Conference*. IEEE, 2015, pp. 1–6.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, and et al., "PyTorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, p. 8024–8035, 2019.

[18] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.