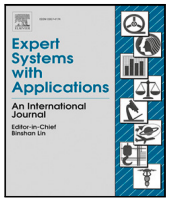
Expert Systems With Applications 234 (2023) 121000

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



# Fed-ANIDS: Federated learning for anomaly-based network intrusion detection systems

Meryem Janati Idrissi [a,*], Hamza Alami [a], Abdelkader El Mahdaouy [b], Abdellah El Mekki [a], Soufiane Oualil [b], Zakaria Yartaoui [b,c], Ismail Berrada [a]

[a] School of Computer Science, Mohammed VI Polytechnic University, Ben Guerir, 43150, Morocco
[b] Modeling, Simulation and Data Analysis, Mohammed VI Polytechnic University, Ben Guerir, 43150, Morocco
[c] National Moroccan Computer Emergency Response Team (maCert), Morocco

ARTICLE INFO

Dataset link: https://github.com/meryemJanatiIdrissi/Fed-ANIDS

*Keywords:*
Network security and privacy
Federated learning
Network intrusion detection
Anomaly detection
Autoencoders

ABSTRACT

As computer networks and interconnected systems continue to gain widespread adoption, ensuring cybersecurity has become a prominent concern for organizations, regardless of their scale or size. Meanwhile, centralized machine learning-based Anomaly Detection (AD) methods have shown promising results in improving the accuracy and efficiency of Network Intrusion Detection Systems (NIDS). However, new challenges arise such as privacy concerns and regulatory restrictions that must be tackled. Federated Learning (FL) has emerged as a solution that allows distributed clients to collaboratively train a shared model while preserving the privacy of their local data. In this paper, we propose Fed-ANIDS, a NIDS that leverages AD and FL to address the privacy concerns associated with centralized models. To detect intrusions, we compute an intrusion score based on the reconstruction error of normal traffic using various AD models, including simple autoencoders, variational autoencoders, and adversarial autoencoders. We thoroughly evaluate Fed-ANIDS using various settings and popular datasets, including USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018. The proposed method demonstrates its effectiveness by achieving high performance in terms of different metrics while preserving the data privacy of distributed clients. Our findings highlight that autoencoder-based models outperform other generative adversarial network-based models, achieving high detection accuracy coupled with fewer false alarms. In addition, the FL framework (FedProx), which is a generalization and re-parametrization of the standard method for FL (FedAvg), achieves better results. The code is available at https://github.com/meryemJanatiIdrissi/Fed-ANIDS.

## 1. Introduction

With the increasing number of attacks on computer networks (Kuypers, Maillart, & Paté-Cornell, 2016), network security has received a lot of attention from security researchers, as well as cybersecurity companies, in their efforts to prevent them. One of the most common security systems used to secure networks is known as Network Intrusion Detection Systems (NIDS) which have known a huge success and have become one of the most widely used security measures to monitor networks for malicious activities.

A NIDS is a software or device which monitors and analyzes all traffic flowing through a network for potential intrusions and notifies the administrator upon detection so that they can be promptly addressed. Traditionally, NIDS can be classified into two primary classes: signature-based (SNIDS) and anomaly-based (ANIDS) systems. The former relies on a database of known signatures to identify attack attempts, while the latter identifies potential anomalies by detecting deviations from a pre-established baseline of normal traffic. On the one hand, SNIDS such as Snort (Caswell & Beale, 2004) and Suricata (Park & Ahn, 2017), can quickly respond to known attacks with fewer false positives. However, they are unable to discover new attacks or previously unseen threats (zero-day attacks) and require frequent maintenance to update them. On the other hand, ANIDS (García-Teodoro, Díaz-Verdejo, Maciá-Fernández, & Vázquez, 2009; Yang et al., 2022; Zhang & Zulkernine, 2006) are more adaptive and efficient in identifying new threats. This comes at the cost of being more vulnerable to false positives due

* Corresponding author.
*E-mail addresses:* Meryem.Janati@um6p.ma (M.J. Idrissi), hamza.alami@um6p.ma (H. Alami), abdelkader.mahdaouy@um6p.ma (A.E. Mahdaouy), abdellah.elmekki@um6p.ma (A. El Mekki), soufiane.oualil@um6p.ma (S. Oualil), zakaria.yartaoui@um6p.ma (Z. Yartaoui), ismail.berrada@um6p.ma (I. Berrada).

https://doi.org/10.1016/j.eswa.2023.121000
Received 1 May 2023; Received in revised form 28 June 2023; Accepted 13 July 2023
Available online 24 July 2023

to the inherent complexity of the techniques often employed for their implementation.

To overcome these issues, this paper investigates how to define a distributed anomaly-based network intrusion detection method that is, on the one hand, able to detect intrusions effectively while incurring low false positive rates. On the other hand, it should provide privacy guarantees to protect sensitive client data. One potential approach could be the use of Deep Learning (DL) models considering their potential to learn complex patterns from raw data. In fact, DL techniques have been effectively employed in various domains, such as computer vision, natural language processing (Chai, Zeng, Li, & Ngai, 2021; Otter, Medina, & Kalita, 2021), and intrusion detection (Elmasry, Akbulut, & Zaim, 2020; Ge, Syed, Fu, Baig, & Robles-Kelly, 2021; Sovilj, Budnarain, Sanner, Salmon, & Rao, 2020). Autoencoders are a specific class of DL models that have shown great promise in network intrusion detection due to their ability to effectively capture and encode complex patterns in data (Al-Qatf, Lasheng, Al-Habib, & Al-Sabahi, 2018; Mirsky, Doitshman, Elovici, & Shabtai, 2018). However, most of the proposed solutions require centralizing all data for training, which is not always feasible for security and privacy constraints (Farahnakian & Heikkonen, 2018; Zavrak & İskefiyeli, 2020). In this context, Federated Learning (FL) (Li, Sahu, Talwalkar and Smith, 2020; McMahan, Moore, Ramage, & y Arcas, 2016) became a prominent distributed learning paradigm for training robust models while preserving the data privacy of the entities belonging to the FL system. FL decouples data collection and model learning by collaboratively training a shared model across multiple entities without moving the data to a central server.

In this work, we propose Fed-ANIDS, a distributed network intrusion detection method based on autoencoders and FL. Fed-ANIDS enables collaborative and secure learning of a global model for network intrusion detection by allowing each entity in the system to learn locally with its own data. To achieve this, Fed-ANIDS utilizes autoencoders to build an anomaly-based system to detect possible attacks. Initially, each entity preprocesses its local data and trains the local model using only normal instances. Then, when all the entities complete the local training, the weights of local models are sent back to the server for aggregation. Finally, when the training is completed and convergence is reached, a threshold selection is performed to find the threshold value, the latter will be used in the detection phase.

We have conducted various experiments with updated and clean versions of USTC-TFC2016 (Wang, Zhu, Zeng, Ye, & Sheng, 2017), CIC-IDS2017 (Sharafaldin, Habibi Lashkari and Ghorbani, 2018), and CSE-CIC-IDS2018[1] datasets. It is worth mentioning that most network datasets commonly used for network intrusion detection suffer from several weaknesses such as pattern redundancy, irrelevant features, packet duplication, mislabeled attack/normal instances, etc. These issues can arise during the labeling process or due to the flow definitions in the tool used to create flow descriptions from raw traffic captures. Thus, we propose to incorporate tools that extract reliable flow features from raw traffic (Engelen, Rimmer, & Joosen, 2021; Liu, Engelen, Lynar, Essam, & Joosen, 2022).

The main contributions of the paper are the following:

1. We propose Fed-ANIDS, an autoencoder-based method for distributed network intrusion detection systems by leveraging FL and anomaly detection. Furthermore, we study the impact of two FL methods, namely FedProx and FedAvg, on the model's performance in the context of NIDS.
2. We use more reliable flow features extracted from USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018 datasets. These features are computed using an updated version of CICFlowMeter,[2] which can handle various challenges related to flow construction, labeling, and attack simulation.
3. We conduct extensive evaluations of Fed-ANIDS using the aforementioned datasets. We also compare our method with Generative Adversarial Network-based models (GAN) that have been utilized for distributed NIDS in the literature (Alhajjar, Maxwell, & Bastian, 2021; Tabassum, Erbad, Lebda, Mohamed, & Guizani, 2022). Moreover, we build and evaluate various scenarios to measure and investigate the generalization performance of Fed-ANIDS with unseen datasets.

The rest of the paper is organized as follows. Section 2 presents various existing works related to NIDS. Section 3 introduces the autoencoder-based distributed learning framework. Section 4 describes the proposed method *Fed-ANIDS* and its different components. Section 5 gives the experimental details and evaluation results. Finally, Section 5 concludes the paper and draws future works.

## 2. Related work

This section provides a state-of-the-art review in the area of FL with a particular focus on intrusion detection systems. Motivated by data privacy and the growing computational power of edge devices, McMahan, Moore, Ramage, Hampson, and Arcas (2017) proposed Federated Learning (FedAvg), an emerging parallelism approach that enables training statistical models directly and locally on end devices. FL is one illustration of the more general approach of "bringing code to data instead of data to code". FL has undergone significant development since then and many approaches have been proposed to tackle the different challenges introduced by FL. One of the major concerns in FL is data heterogeneity. To alleviate the inconsistency caused by training a global model on such data, Li, Sahu, Zaheer et al. (2020) introduced FedProx, a generalized and reparameterized version of FedAvg. FedProx makes a small modification to FedAvg by adding a regularization term to each local objective function. The numerical results show that FedProx is more stable with higher accuracy in heterogeneous networks compared to FedAvg.

The great success that FL has witnessed has led to its adoption in many research areas, in particular, intrusion detection (Agrawal et al., 2021). Several works have been proposed to perform intrusion detection in IoT networks by leveraging FL. For instance, Nguyen et al. (2019) have introduced DÏoT, a self-learning distributed system for detecting anomalous behaviors in IoT networks. The proposed solution utilizes FL to efficiently aggregate behavior profiles. Additionally, DÏoT consists of two main components, a *Security Gateway (SG)*, and an *IoT Security Service (SS)*. On the one hand, SG is responsible for identifying the type of new devices added to the system, as well as performing anomaly detection and detecting compromised devices, in other words, each security gateway locally trains a GRU model for a specific device type and send it to the IoT security service for aggregation. The SS component, on the other hand, supports the SG by maintaining and aggregating device-type-specific models received from all SG. The authors evaluated their approach on 30 IoT devices contaminated with real IoT malware and demonstrated that their system can achieve up to 95.6% true positive rate with zero false alarms. Rahman, Tout, Talhi, and Mourad (2020) proposed an FL-based system for intrusion detection in IoT devices to enable knowledge sharing between peers without losing to privacy issues. The evaluation was conducted on the NSL-KDD dataset considering several data distribution scenarios and a comparison between FL, on-device, and centralized approaches was made. Despite the important difference between FL and centralized learning, yet, FL outperforms on-device learning and is the closest to centralized learning. Similarly, Tian, Chen, Yu, and Liao (2021) proposed DC-Adam, an asynchronous FL anomaly detection approach for IoT systems. To mitigate the inconsistency caused by the gradient delay problem, the authors suggest using Taylor Expansion Formula to lessen the oscillation of the loss function when minimizing it. Moreover, a warm-up procedure for parameter

[1] https://www.unb.ca/cic/datasets/ids-2018.html.
[2] https://github.com/GintsEngelen/CICFlowMeter.

initialization was used to prevent the model divergence under Non-IID data scenarios. For anomaly detection, a denoising autoencoder model was employed, where the reconstruction error was considered as an anomaly score. The proposed approach was evaluated on MNIST (LeCun, Cortes, & Burges, 2010), CIC-IDS2017 (Sharafaldin, Lashkari and Ghorbani, 2018), and IoT-23 (Garcia, Parmisano, & Erquiaga, 2020), and was compared with three other approaches, namely Asynchronous Adam (Asyn-Adam), Asynchronous SGD (Asyn-SGD), and Synchronous Adam (Syn-Adam). Asyn-DC-Adam was proven to converge steadily compared to Asyn-Adam and Asyn-SGD and nearly matches Syn-Adam. Moreover, Asyn-DC-Adam outperforms all the before-mentioned approaches in terms of accuracy, precision, recall, and F1-score (91.65%, 92.92%, and 90.15% F1-score on Mnist, CIC-IDS-2017, and IoT-23 respectively). Despite the benefits of FL approaches especially in IDS, FL models are susceptible to many adversarial attacks leading to their failure. This was shown in Rey, Sánchez Sánchez, Huertas Celdrán, and Bovet (2022), where the authors investigated FL for malware detection in IoT devices for both cases supervised (based on a Multi-Layer Perceptron (MLP) model) and unsupervised (based on an Auto-Encoder model) learning. The authors used N-BaIoT (Meidan et al., 2018a) for evaluation, a dataset that models the traffic of real IoT devices impacted by malware, and compared the proposed framework with centralized and distributed architectures. High accuracy was obtained (up to 99%) for both supervised and unsupervised learning and under several scenarios, this is mainly because N-BaIoT is considered an easy and less complex dataset. However, this accuracy is shown to drop drastically under adversarial attacks, and though the use of robust aggregation functions such as median shows an improvement compared to FedAvg yet insufficient, demonstrating the need for more robust countermeasures.

Several other works have been proposed for FL-based NIDS. In fact, the major challenges are related to data issues, such as data scarcity and data dimensionality issues. The work presented in Zhao, Chen, Wu, Teng, and Yu (2019), tackled the first problem of data scarcity by leveraging the multi-task learning paradigm and the heterogeneity of real-world intrusion datasets, and they proposed MT-DNN-FL, a multi-task deep neural network in FL. On the one hand, the proposed method performs multiple tasks simultaneously, namely, network anomaly detection, traffic recognition, and traffic classification. On the other hand, the adoption of federated learning architecture guarantees user data privacy. The experiments performed on CIC-IDS2017 (Sharafaldin, Lashkari et al., 2018), ISCXVPN2016 (Habibi Lashkari, Draper Gil, Mamun, & Ghorbani, 2016), and ISCX-Tor2016 (Habibi Lashkari, Draper Gil, Mamun, & Ghorbani, 2017) demonstrated the effectiveness of MT-DNN-FL compared to multiple single-task deep neural networks (DNN, k-NN, RF, etc.). Ayed and Talhi (2021) investigated the effectiveness of FL for anomaly detection in NIDS. For this purpose, the CIC-IDS2017 dataset was used, where the proposed experimentation scenario respects the network topology and node characteristics, more specifically, every IP address presented in the dataset was considered as a node in the FL architecture. The authors conducted several experimentation scenarios depending on the client selection strategy and train/test data percentage. The results showed that this method can achieve up to 93% accuracy in some scenarios while preserving data privacy. While in Qin and Kondo (2021), the data dimensionality challenge was addressed. The authors employed FL to build an anomaly-based network intrusion detection through an on-device sequential learning neural network ONLAD (Tsukada, Kondo, & Matsutani, 2020). To tackle the data dimensionality issue, a greedy feature selection algorithm was used to find the set of features that produces the best accuracy according to each device's target attack types. Then, only the devices with the same feature space are gathered and an FL model is created for each group, resulting in multiple global models. Experiments carried out on the NSL-KDD dataset indicate that the best accuracy obtained is 70.4% and an overall improvement is 25.7% Likewise, Tabassum et al. (2022) proposed a privacy-preserving FL-based framework, named FEDGAN-IDS. The latter uses a GAN (Generative Adversarial Network) based model in a distributed setup for two reasons, the first for data augmentation and the second to perform binary and multi-class classification. The proposed framework was evaluated on NSL-KDD, KDD-CUPP99, and UNSW-NB15, and was compared with FED-IDS, an IDS based on FL and simple neural networks. FEDGAN-IDS is shown to outperform FED-IDS on all datasets in terms of accuracy and convergence rate.

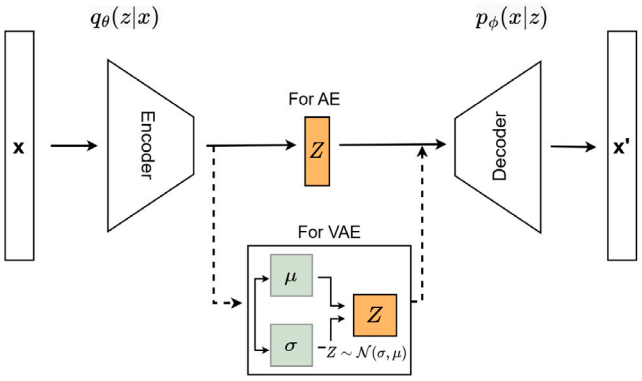



**Fig. 1.** The general architecture of simple AutoEncoders and Variational AutoEncoders.

Several works have been proposed in the area of network intrusion detection using anomaly-based models, but very few have utilized federated learning approaches to address privacy concerns. To the best of our knowledge, none of the works has incorporated autoencoders in a federated setting for NIDS. Our proposed method addresses this gap by leveraging three variants of AEs to improve detection accuracy while maintaining privacy through the use of federated learning. AEs are used for anomaly-based intrusion detection. While FL provides a solution to the challenges of data privacy and security associated with sharing sensitive data between parties.

## 3. Autoencoder-based distributed learning framework

In this section, we introduce a structured approach to securely train an intrusion detection model in distributed settings. This approach consists of two main parts, namely autoencoders and FL. In the following, we first define what autoencoders are and how they work. Then we describe FL and its two most used techniques including *FedAvg* and *FedProx*. Finally, we present the practice that trains an autoencoder-based model across multiple decentralized clients.

### 3.1. Autoencoders

An autoencoder is a specific type of unsupervised neural network that learns an "informative" representation of input data. Let $x \in \mathbb{R}^d$ be the input and $z \in \mathbb{R}^p$ the latent representation of $x$, such that $p \ll d$. Simple AutoEncoders (AE) (Bank, Koenigstein, & Giryes, 2020) encompass two blocks, an encoder, and a decoder. The former is a Neural Network (NN) that learns a latent representation $z$ of $x$. The latter is also a NN that uses the vector $z$ generated by the encoder to reconstruct the input data with minimal loss error, such that $q_\theta(z|x)$ and $p_\phi(x|z)$ are the encoding and decoding distributions, respectively. Variational AutoEncoders (VAE) (Kingma & Welling, 2014) are similar to simple AEs, nevertheless, they map the input vector to a distribution of mean $\mu$ and standard deviation $\sigma$ from which the latent vector $z$ is sampled. Fig. 1 illustrates the general architecture of simple AE and VAE.

Adversarial Autoencoders (AAE) (Makhzani, Shlens, Jaitly, & Goodfellow, 2015) are hybrid models that fuse AEs and Generative Adversarial Networks (GAN) (Goodfellow et al., 2014). It uses the adversarial

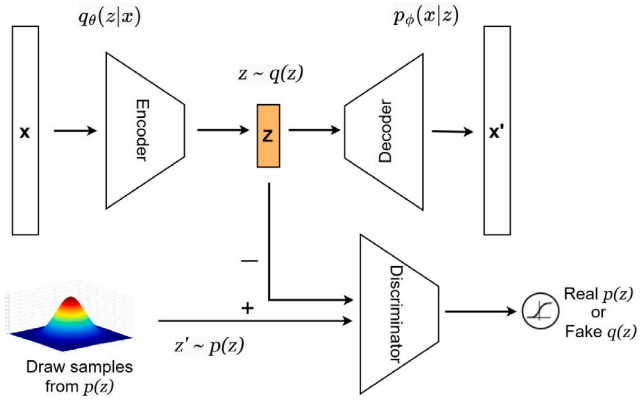



**Fig. 2.** The general architecture of Adversarial AutoEncoders.

loss concept introduced by GAN to improve the regularization of an autoencoder. More specifically, AAE employs adversarial learning in order to impose the latent space to follow a certain distribution that could be a p-dimensional normal distribution $\mathcal{N}(0, I)$. The overview of AAE architecture is illustrated in Fig. 2, where $p(z)$ is the prior distribution we want to impose on $z$, and $q(z)$ is the distribution of the latent variable. The top half of Fig. 2 is a standard autoencoder that plays the role of the generator and $z$ is the generated data. The generator attempts to fool the discriminator into believing that the latent representation is sampled from the pre-chosen distribution. On the other hand, the discriminator $D_\chi(z)$, depicted in the bottom half of the figure, predicts whether a given latent vector is generated by the encoder (fake) or sampled from the predefined distribution (real). Both the autoencoder and the adversarial network are trained jointly with Stochastic Gradient Descent (SGD) in two phases: the *reconstruction phase* and the *regularization phase*. In the reconstruction phase, only the parameters of the encoder and decoder are optimized in order to minimize the reconstruction loss of the inputs, while in the regularization phase, both the discriminator and the generator (encoder) are trained at once. First, the discriminator learns how to distinguish between the real samples (drawn from the prior) and fake samples (generated by the encoder). Then, the discriminator is fixed and the generator is trained to acquire the ability to produce samples following the prior.

### 3.2. Federated learning

Federated learning is a distributed machine-learning approach that permits numerous clients to collaboratively learn a shared model without sharing their data with a central server. In fact, each client optimizes a local model with its own data, and then a global model is aggregated by the server from these current local models. The two popular FL algorithms are FedAvg and FedProx:

- *FedAvg* (McMahan et al., 2017) is a simple algorithm consisting of computing a global model based on the weighted average of the parameters collected from clients. The model is then sent back to the clients for further training. FedAVG is easy to implement and scales well to large datasets and FL networks with a large number of clients.
- *FedProx* algorithm (Li, Sahu, Zaheer et al., 2020) is similar to FedAvg but adds a regularization term to the loss function of each client. The regularization term $\delta$ is a proximal term that penalizes the divergence of the client model parameters from a central point, typically the global model at the previous iteration.

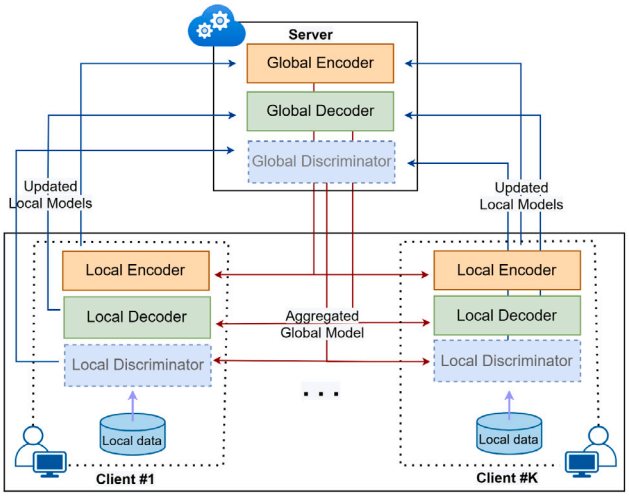



**Fig. 3.** Distributed learning of different variations of autoencoders (simple AE, VAE, and AAE) for NIDS using FL.

### 3.3. Distributed learning using autoencoder-based models

We assume that there are $K$ clients in the network installed in different locations and connected to a central server. Each of which possesses collected benign data $D_k$ that is kept private. We seek to build a machine learning-based NIDS leveraging FL, i.e. privately learn a global model from the network data of all clients for network intrusion detection. The FL training requires protecting the privacy of the clients, while the NIDS should accurately detect attacks with a low rate of false alarms.

We design a distributed NIDS using autoencoders and FL techniques. Each client connected to the central server runs a copy of the global encoder and the global decoder (and the global discriminator for AAE) with his local benign data. Once the local autoencoder is trained, the updated weights are shared with the central server for aggregation. At the detection phase, the global autoencoder aims to report any possible network intrusion. Fig. 3 depicts the architecture of distributed learning using autoencoder-based models for NIDS using FL.

At each training round $t \in E$, a subset $S_t$ of $m = max(C \times K, 1)$ clients are selected at random to take part in the current round such that $C$ is the fraction of clients to be chosen to participate. Each client $k \in S_t$ receives the global model and optimizes it with its local data $D_i$ for $I$ local iterations. The optimization is done using the mini-batch gradient descent technique. Once the training is completed, the client $k$ sends the local weights $\theta_t^i$ (encoder), $\phi_t^i$ (decoder), and $\chi_t^i$ (discriminator) back to the server for aggregation. These steps are repeated until we reach convergence, i.e. the performance of the model gets closer and closer to a specific value. Note that the server acts solely as a coordinator, ensuring that the aggregated model represents the collective knowledge of all participating clients without having any access to the data during training. This ensures privacy protection and prevents the server from gathering any sensitive client data during the training process. Table 1 introduces the main notations used in the remainder of this paper.

## 4. Proposed method

We propose *Fed-ANIDS*, an anomaly-based network intrusion detection method based on FL and autoencoders. Fig. 4 presents the overall architecture of Fed-ANIDS. It consists of four main components: (1) *Global model initialization*, (2) *Local training*, (3) *Model aggregation*, and (4) *Model dissemination*. Each phase is thoroughly explained in the remainder of this section.

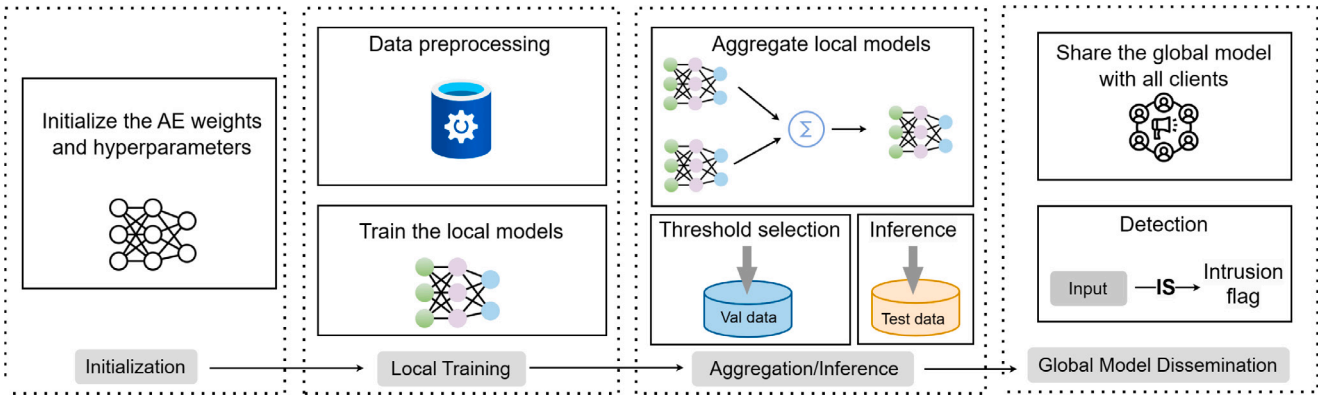



**Fig. 4.** Fed-ANIDS architecture which consists of 4 main components including, global model initialization, local training, model aggregation, and model dissemination.

**Table 1**
Notations used to describe various steps in the remaining of the paper.

| Notation | Description |
|---|---|
| $K$ | All clients/entities in the network |
| $E$ | Total number of communication rounds |
| $I$ | Total number of local iterations |
| $N$ | Size of mini-batch |
| $IS$ | Intrusion score |
| $thr$ | Anomaly detection threshold |
| $\delta$ | The penalty term for FedProx algorithm |
| $D_k$ | Local dataset of client k |
| $\alpha$ | Learning rate of Encoder |
| $\beta$ | Learning rate of Decoder |
| $\gamma$ | Learning rate of Discriminator |
| $p(z)$ | The prior distribution for AAE |
| $\theta_t$ | Encoder parameter at round t |
| $\phi_t$ | Decoder parameter at round t |
| $\chi_t$ | Discriminator parameter at round t |
| $\mathscr{B}$ | The local mini-batch size |

### 4.1. Global model initialization

We assume that we are in a distributed learning network. The central server starts the learning process by initializing the weights of the global model as well as the hyperparameters needed for the training such as learning rates, momentum, $\delta$ value (for FedProx), etc. The global model can be an AE, a VAE, or an AAE. In the case of the AAE model, the server is also in charge of defining a prior distribution $p(z)$. Once the initialization is done, the server shares the model and the hyperparameters with the clients selected in the first round.

### 4.2. Local model training

We consider $K$ clients that collaboratively train a global model for a network intrusion detection task. The clients perform two main tasks, data preprocessing, and local training:

#### 4.2.1. Feature extraction & data preprocessing

Before being fed to the autoencoder-based model, each client ensures that his data is prepared and of quality to be consumed. For each dataset, the publicly available *PCAP*[3] files were utilized and preprocessed according to the works of Engelen et al. (2021) and Liu et al. (2022). First, the *pcapfix*[4] tool was run on all *PCAPs* to repair any possible corrupted or damaged ones. The output *PCAP* files are then run through *reordercap*,[5] a program that sorts packets by timestamp. This step is most helpful in cases where a file has been created by combining frames from more than one source without taking time order into consideration. The resulting files are fixed and ordered *PCAPs* ready to use. Existing NIDS datasets suffer from various pitfalls related to flow construction, labeling, and attack simulation. Therefore, Engelen et al. (2021) and Liu et al. (2022) have proposed an improved version of the CICFlowMeter tool to generate cleaner NIDS datasets. We use the proposed tool to extract 87 statistical flow features from *PCAP* files and save them into CSV files. For CIC-IDS2017 and CSE-CIC-IDS2018 flow labeling, we follow the guidelines[6] of Liu et al. (2022). The extracted features are composed of both numerical and categorical features, hence we convert character data into numeric values (features encoding). Finally, we perform min–max scaling to normalize the data. Fig. 5 illustrates the data preprocessing pipeline.

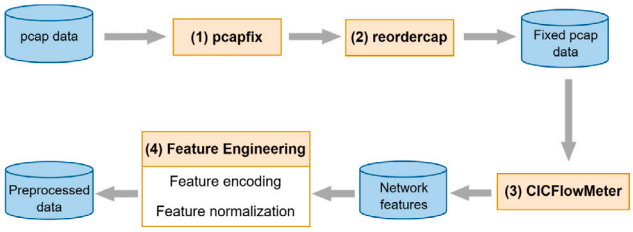



**Fig. 5.** Data preprocessing pipeline.

#### 4.2.2. Local training

If selected, a client downloads the current global state of the autoencoder from the server. The autoencoder is then fed with only normal samples of the client's local data and trained to learn the encoding and the reconstruction of the normal behavior. The mean squared error is minimized between the input $x$ and its reconstruction $x'$. To train the discriminator of the AAE, an isotropic gaussian distribution $\mathcal{N}(\mu, I)$ is used as the imposed prior. Fed-ANIDS uses the *FedProx* algorithm to update the local models, hence a regularization term $\frac{\delta}{2}\|w - w_t\|^2$ is added to the loss function of each client, where $w$ are the parameters of the global model, $w_t$ are the parameters of a local model at round $t$, and $\delta \in [0, 1]$ is a regularizer factor. The aforementioned steps are shown in Algorithm 1 for the client's side.

### 4.3. Model aggregation

Model aggregation is a crucial step in FL. It allows the local models trained on separate devices to be combined into a single global model

[3] Packet Capture, a file format for storing raw network packets.
[4] https://github.com/Rup0rt/pcapfix.
[5] https://www.wireshark.org/docs/man-pages/reordercap.html.
[6] https://github.com/GintsEngelen/CNS2022_Code.

**Algorithm 1** clientUpdate: optimize clients models weights

**Input**: The parameters $\theta$, $\phi$, and $\chi$ for the encoder, decoder, and discriminator resp; the prior distribution; the local normal data $X = [x_0, ..., x_{N-1}] \sim p(x)$; the number of local epochs $I$; the set of mini-batches $\mathcal{B}$ each of size $N$; parameters $\alpha, \beta, \gamma$ for the learning rates for the encoder, decoder, and discriminator resp; the regularizer factor $\delta$ FedProx.
**Output**: Normal or Attack.
**for** $i = 1$ to $I$ **do**
  **for** batch $b \in \mathcal{B}$ **do**
    ▷ Reconstruction phase
    ▷ Minimize the reconstruction loss
    $\mathscr{L}_{rec} = \frac{1}{N}\sum_{i=0}^{N-1}(x_i - x_i')^2 + \frac{\delta}{2}\|\phi - \phi_t\|^2$
    $\phi \leftarrow \phi - \alpha\nabla_\phi \mathscr{L}_{rec}$
    $\theta \leftarrow \theta - \beta\nabla_\theta \mathscr{L}_{rec}$
    **if** AAE **then**:
      ▷ Generate fake samples $z_{fake} \sim q_\phi(z|x)$ by the encoder
      ▷ Draw samples from the prior $z_{real} \sim p(z)$
      ▷ Regularization phase
      ▷ Train the discriminator
      $\mathscr{L}_{dis} = -\frac{1}{N}[\sum_{i=0}^{N-1} log D_\chi(z_{real_i}) + \sum_{i=0}^{N-1} log(1 - D_\chi(z_{fake_i}))] + \frac{\delta}{2}\|\chi - \chi_t\|^2$
      $\chi \leftarrow \chi - \gamma\nabla_\chi \mathscr{L}_{dis}$
      ▷ Train the generator (encoder) to match the prior
      $\mathscr{L}_{prior} = \frac{1}{N}\sum_{i=0}^{N-1} log(1 - D_\chi(z_{fake_i})) + \frac{\delta}{2}\|\chi - \chi_t\|^2$
      $\phi \leftarrow \phi - \alpha\nabla_\phi \mathscr{L}_{prior}$
    **end if**
  **end for**
**end for**
▷ Send $\theta$, $\phi$, and $\chi$ to the server

**Algorithm 2** serverAgg: build global model by aggregating clients models

**Input**: $\theta$, $\phi$, and$\chi$ the parameters for the encoder, the decoder, and the discriminator resp; the $K$ clients; the number of global rounds $E$;
**Output**: $\theta$, $\phi$, and $\chi$
Initialize $\theta_0$, $\phi_0$, and $\chi_0$
**for** each round $t = 0, \dots, E-1$ **do**
  m $\leftarrow$ max(C $\times$ K, 1)
  $S_t \leftarrow$ select a random set of m clients
  ▷ send $\theta_t$, $\phi_t$ and $\chi_t$ to each client k $\in S_t$
  **for** each client k $\in S_t$ **do**
    $\theta_{t+1}^k, \phi_{t+1}^k, \chi_{t+1}^k \leftarrow clientUpdate(w_t)$
  **end for**
  $\theta_{t+1} = \frac{1}{K}\sum_{k\in S_t}\theta_{t+1}^k$
  $\phi_{t+1} = \frac{1}{K}\sum_{k\in S_t}\phi_{t+1}^k$
  **if** AAE **then**
    $\chi_{t+1} = \frac{1}{K}\sum_{k\in S_t}\chi_{t+1}^k$
  **end if**
  Disseminate the updated parameters for the next round.
**end for**

while maintaining data privacy. At the end of each communication round, each participating client sends his local update of the model to the server. Upon receiving all the updates, the server computes the weighted average as the new global weight parameters. The global model is then updated with the weighted average, and the process is repeated until convergence. Algorithm 2 depicts the steps of model aggregation executed by the server.

### 4.4. Threshold selection

Once the training procedure is completed, a score threshold must be computed for the intrusion detection phase. For this purpose, we set a separate validation set upon which we determine a proper threshold for the global model. We should mention that the model performance is heavily reliant on the threshold value. On the one hand, a higher value would result in fewer false alarms but could also result in marking more attacks as normal instances. On the other hand, a lower value would produce more false alarms and mark more normal instances as attacks. We propose to compute the threshold $thr$ following Eq. (1), i.e., the threshold is the sum and standard deviation of $MSE$ over the validation set (Meidan et al., 2018b).

$$thr = \overline{(MSE(D_{val}, \phi_t))} + s(MSE(D_{val}, \phi_t)) \quad (1)$$

The determined threshold is then used at the inference stage. We compare the Intrusion Score (IS), which is the reconstruction loss, with the pre-computed threshold. If the intrusion score is greater than the threshold, then the instance is considered an intrusion; otherwise, the instance is benign.

### 4.5. Model parameters dissemination

Once the training is completed and the convergence is reached, the latest update of the global model is shared with all clients available in the network. When a new instance $X_{new}$ comes into the network, an intrusion score is computed (reconstruction loss). The instance $X_{new}$ is marked as an intrusion if the IS is greater than the threshold, else it is marked as a normal instance.

$$X_{new} = \begin{cases} Anomaly, & \text{if } IS > thr \\ Normal, & \text{otherwise} \end{cases} \quad (2)$$

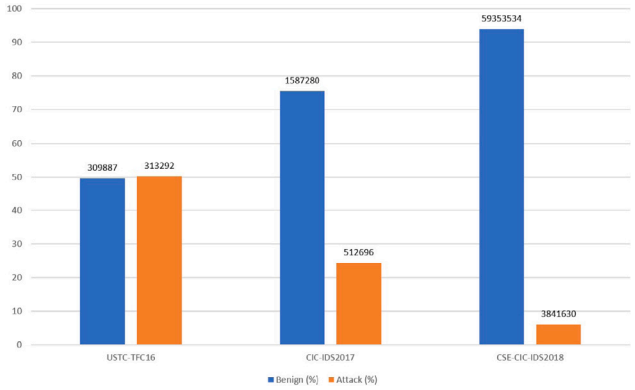



**Fig. 6.** The distribution of benign and attack samples of USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018 datasets.

## 5. Experimental results

In this section, we evaluate the performance of the proposed method *Fed-ANIDS* with various well-known datasets (USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018). We mainly focus on answering the following research questions:

**RQ1:** How does Fed-ANIDS perform in comparison to other baselines (Generative Adversarial Network (GAN), and a Bidirectional GAN (BiGAN)) according to various intrusion detection metrics?
**RQ2:** What is the most efficient distributed algorithm when comparing FedProx and FedAvg?
**RQ3:** What is the performance of Fed-ANIDS when training data comes from various environments, and how does the proposed method perform with previously unseen data?

### 5.1. Datasets

We provide a brief description of the three well-known datasets we used for the Fed-ANIDS performance evaluation.

***USTC-TFC2016*** (Wang et al., 2017) is a prominent dataset that primarily contains two parts. The first part is composed of 10 varieties of malware traffic collected from public websites hosted in a real network environment within the time period 2011 to 2015. The second part contains ten genres of normal traffic collected using IXIA BPS.[7] This dataset comes in *PCAP* format with a total size of 3.71 GB. In this work, we split *USTC-TFC2016* at a 70-20-10 ratio for training, validation, and test sets.

***CIC-IDS2017*** (Sharafaldin, Habibi Lashkari et al., 2018) is a state-of-the-art network intrusion detection dataset. It provides real-world network traffic data ("Benign" and "Attacks") in the *PCAP* format collected during the period of five days (Monday through Friday). In our experiments, we pick the first four days for training and validation (Monday through Thursday) and we test on Friday data.

***CSE-CSE-CIC-IDS2018*** is an improved version of *CIC-IDS-2017* dataset. It was generated and launched by The Canadian Institute for Cybersecurity in 2018/2019.[8] It consists of seven up-to-date attack types including Botnet, Brute-force, Denial of Service and Distributed Denial of Service, Heartbleed, Inside Network Infiltration, and Web Attacks. Since there are two weeks of data in CIC-IDS-2018, we allocate the first week for training and validation, while we preserve the second week for testing.

In real-world scenarios, normal data in network intrusion detection is generally more abundant than attack samples due to the nature of network traffic. Considering the prevalence of normal network traffic, training an anomaly-based intrusion detection system solely on normal data is a practical approach. Therefore, in this work, before partitioning the data over clients, only the normal samples are left in the training and validation sets, while the remaining attack samples are included in the test set. Finally, the training set is equally and randomly partitioned between all the clients in the system, ensuring a fair distribution of normal data for training. Fig. 6 depicts the distribution of benign and attack samples of each dataset.

### 5.2. Experimental settings

In this work, we consider a distributed environment of $K = 10$ clients in total. We fix the client fraction at $C = 0.5$. We set the local iterations at $I = 10$ and the global communication rounds at $E = 30$ epochs for USTC-TFC2016 and CIC-IDS2017 and $E = 10$ for CSE-CIC-IDS2018. For training hyperparameters, we use the validation set to find the best combination that maximizes the performance. For the regularization parameter for FedProx, we tested the values in the candidate set {0.1, 0.01, 0.001, 1e−4, 1e−5, 1e−6}. As for the learning rate and weight decay, we tested the values in the range {0.01, 0.001, 1e−4, 1e−5} and {1e−4, 1e−5, 1e−6} respectively.

The AE architecture adopted in this work consists of two components, namely an encoder and a decoder, which are modeled as two fully-connected layer networks specified by the following number of neurons per layer (87-64-32) and (32-64-87), respectively. Each layer is followed by a ReLU activation function. Similarly, the VAE architecture consists of four layers such that (87-64-64-32) and (32-64-64-87) are the number of neurons per layer for the encoder and the decoder, respectively. For AAE, we model both the encoder and the decoder as three fully-connected layer networks specified by the following number of neurons per layer (87-16-4-2) and (2-4-16-87), respectively. Each layer is followed by a LeakyReLU activation function. Similarly, the discriminator consists of three fully-connected layers specified by the following number of neurons per layer (16-4-2). Each layer is followed by a LeakyRelu activation function except the last layer which is followed by a sigmoid function.

To evaluate the performance of our approach, we consider three standard intrusion detection metrics, F1-score, Accuracy, and False Discovery Rate (FDR). Note that FDR represents the ratio of falsely reported anomalies to total anomalies. We use Python and machine learning libraries such as PyTorch, Numpy, and Pandas to perform all experiments. This work was carried out using the African SuperComputing Center HPC service, supported by Mohammed VI Polytechnic University.[9]

### 5.3. Performance evaluation

To make our evaluations credible, we compare Fed-ANIDS with prior works that have a similar context. Tabassum et al. (2022) introduced a system that leverages AD and FL for intrusion detection. The proposed system is mainly based on GANs (Goodfellow et al., 2014) which are commonly employed across diverse fields, including computer vision and anomaly detection.

According to our proposed method, we implemented two baseline models and evaluated their performances with USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018. The first model is a simple GAN while the second model is a BiGAN (Donahue, Krähenbühl, & Darrell, 2016). The latter has the ability to learn rich representations in applications like unsupervised learning and anomaly detection (Mattia, Galeone, Simoni, & Ghelfi, 2019). Both Autoencoders and GANs are unsupervised learning techniques that can be used for network intrusion detection. The two other major distinctions between our work and Tabassum et al. (2022) are (1) we use the FedProx FL technique, unlike the work of Tabassum et al. (2022) which uses the FedAvg FL technique and (2) we perform all evaluations with cleaner versions of flow features extracted from USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018 datasets, while Tabassum et al. (2022) have conducted their experiments on images generated from NSL-KDD,[10] KDD (Tavallaee, Bagheri, Lu, & Ghorbani, 2009), and UNSW-NB15 (Moustafa & Slay, 2015) datasets. The format of datasets is important in a FL network. Tabular datasets that contain various features extracted from raw traffic are more interpretable, require less storage, and result in a smaller global model size when compared to raw traffic image datasets. Considering that the edge entities in FL systems usually have limited resources, we work with flow features-based datasets.

Table 2 provides the obtained results when evaluating our model with the USTC-TFC2016 dataset. We observe that AAE with the FedProx algorithm yields the best results with the highest F1-score of 99.94% and accuracy of 99.95% and the smallest FDR value of 0.18%, followed by VAE and AE which also perform well. On the contrary, GAN and BiGAN perform poorly, especially BiGAN which induces a high FDR with a low F1-score and accuracy. In addition, FedProx is always on par or better than FedAvg for all models. Table 3 presents the evaluation performance with CIC-IDS2017. We observe that a simple AE trained with FedProx outperforms all other models, with an accuracy that exceeded 93% and the lowest FDR of 1.693%. Lastly, Table 4 gives the scored results when using the CSE-CIC-IDS2018 dataset. VAE is found to be more effective in terms of F1-score and accuracy. The simple AE model achieved the best FDR 0.6% at the cost of a drop in F1 score and accuracy. Overall, the simple AE model is the best-performing model.

For each dataset, at least one of the autoencoder-based approaches with FedProx outperforms other GAN-based models. Thus, our findings suggest that in the context of NIDS, autoencoders are more effective in detecting potential threats compared to GAN-based approaches.

[7] https://www.ixiacom.com/products/breakpoints.
[8] https://www.unb.ca/cic/datasets/ids-2018.html.
[9] https://ondemand.hpc.um6p.ma/.
[10] http://nsl.cs.unb.ca/NSL-KDD/.

**Table 2**
Performance evaluation of Fed-ANIDS with USTC-TFC2016 dataset.

| Metrics | Fed-ANIDS (ours) | | | | | | Baseline | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AE | | VAE | | AAE | | GAN | | BiGAN | |
| | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx |
| F1-score (%) | 99.35 | 99.44 | 99.79 | 99.77 | 99.93 | **99.94** | 94.96 | 96.77 | 61.02 | 72.78 |
| FDR (%) | 0.46 | 0.22 | 0.39 | **0.01** | 0.18 | 0.18 | 9.90 | 8.45 | 5.14 | 15.43 |
| Accuracy (%) | 99.54 | 99.60 | 99.85 | 99.84 | 99.94 | **99.95** | 96.38 | 97.66 | 81.40 | 84.68 |

**Table 3**
Performance evaluation of Fed-ANIDS with CIC-IDS2017 dataset.

| Metrics | Fed-ANIDS (ours) | | | | | | Baseline | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AE | | VAE | | AAE | | GAN | | BiGAN | |
| | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx |
| F1-score (%) | 92.51 | **92.73** | 60.96 | 62.09 | 80.11 | 77.17 | 80.28 | 74.95 | 71.24 | 78.12 |
| FDR (%) | 1.75 | 1.69 | 49.49 | 45.66 | **0.34** | 5.60 | 24.68 | 34.99 | 30.39 | 34.96 |
| Accuracy (%) | 93.36 | **93.54** | 64.34 | 66.56 | 83.94 | 81.62 | 81.90 | 76.28 | 74.97 | 78.63 |

**Table 4**
Performance evaluation of Fed-ANIDS with CSE-CIC-IDS2018 dataset.

| Metrics | Fed-ANIDS (ours) | | | | | | Baseline | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AE | | VAE | | AAE | | GAN | | BiGAN | |
| | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx |
| F1-score (%) | 88.48 | 86.41 | 89.40 | **90.64** | 45.51 | 45.52 | 47.85 | 46.60 | 55.01 | 55.12 |
| FDR (%) | 1.51 | **0.60** | 2.14 | 1.81 | 16.67 | 16.66 | 17.13 | 16.85 | 14.75 | 12.92 |
| Accuracy (%) | 92.90 | 91.08 | 93.72 | **94.48** | 83.34 | 83.34 | 75.58 | 80.77 | 72.95 | 73.36 |

Therefore, solutions for network intrusion detection in real-world scenarios that are based on autoencoders would be more appropriate than GAN-based solutions since they are simple, light, and computationally efficient.

### 5.4. Model generalization for heterogeneous networks

We also evaluate the generalization performance of Fed-ANIDS according to three scenarios using the three previously presented datasets (which are collected from different sources and environments). The first scenario consists of training the model using a subset of USTC-TFC2016 and CIC-IDS2017 datasets. The second scenario builds the model with USTC-TFC2016 and CSE-CIC-IDS2018 datasets. The last scenario optimizes the model with CIC-IDS2017 and CSE-CIC-IDS2018. For each scenario, we evaluate the model with a subset of the testing set of all datasets, such that, for each scenario one of the datasets was never seen before. These scenarios allow us to test the generalization strength of the proposed model under FL in two ways: (1) how well the model generalizes when training with two datasets collected in different environments, and (2) how the model performs with a new dataset never seen before. Considering a network of twenty clients in total, we select the simple autoencoder (AE) as the autoencoder-based model and we split two datasets among ten clients each (e.g. USTC-TFC2016 and CIC-IDS2017) then we train the model for $E = 30$ rounds such that at each round third of the clients are chosen randomly ($C = 0.3$). Once the training is completed, we evaluate the model not only on the two datasets used for training but also on a third one never seen before (e.g. CSE-CIC-IDS2018).

Prior to the generalization evaluation of Fed-ANIDS, Table 5 presents the baseline results of the simple AE model when training separately on each dataset. Tables 6–8 provide the evaluation results for the three scenarios of generalization. Compared to the baseline (Table 5), we observe that in most cases the performance deteriorates in terms of all metrics and the model cannot generalize on seen and unseen datasets. However, the decrease in performance is much larger for some cases than others. For instance, when considering the first scenario (see Table 6), on the one hand, the performance dropped drastically for USTC-TFC2016 despite it being used for training. On the other hand, the model could generalize relatively well on CSE-CIC-IDS2018 which was never seen during training. On the contrary, in the second scenario (see Table 7) the model could not generalize on the CIC-IDS2017 dataset, which was never seen, yet it performed well on USTC-TFC2018 and CSE-CIC-IDS2018 datasets used for training. Lastly, in Table 8, the mode performs well on the CSE-CIC-IDS2018 dataset but achieves poor results on USTC-TFC2016 and CIC-IDS2017. These results can be attributed to the difference in the dataset size, where CSE-CIC-IDS2018 is almost five times bigger than USTC-TFC2016 and CIC-IDS2017. In view of all the obtained results, the FedProx algorithm yields much better results compared to FedAvg, in some cases up to 10% F1-score and accuracy and more than 6% FDR. This can be explained by the fact that the FedProx algorithm tackles the heterogeneity challenge in federated networks.

**Table 5**
Baselines testing results of a simple AE trained on USTC-TFC2016, CIC-IDS2017 and CSE-CIC-IDS2018 datasets separately.

| Metrics | USTC-TFC2016 | | CIC-IDS2017 | | CSE-CIC-IDS2018 | |
|---|---|---|---|---|---|---|
| | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx |
| F1-score (%) | 99.35 | 99.44 | 68.00 | 68.16 | 79.91 | 89.46 |
| FDR (%) | 0.47 | 0.23 | 0.14 | 0.14 | 1.03 | 1.91 |
| Accuracy (%) | 99.54 | 99.60 | 93.72 | 93.79 | 84.55 | 93.01 |

**Table 6**
Testing results of a simple AE trained on USTC-TFC2016 and CIC-IDS2017.

| Metrics | Seen datasets | | | | Unseen dataset | |
|---|---|---|---|---|---|---|
| | USTC-TFC2016 | | CIC-IDS2017 | | CSE-CIC-IDS2018 | |
| | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx |
| F1-score (%) | 71.51 | 78.79 | 58.12 | 66.02 | 64.25 | 65.14 |
| FDR (%) | 41.04 | 31.25 | 0.34 | 0.32 | 0.078 | 0.08 |
| Accuracy (%) | 72.36 | 79.80 | 87.52 | 93.24 | 67.38 | 68.40 |

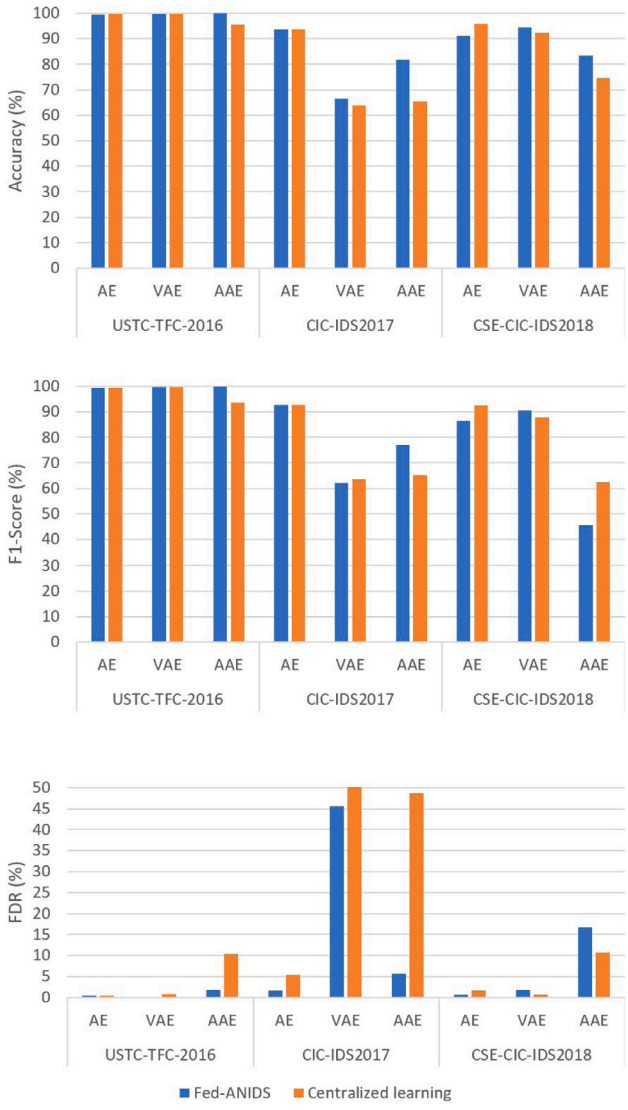



**Fig. 7.** Comparion between Fed-ANIDS and centralized learning.

### 5.5. Distributed vs. centralized learning for anomaly-based NIDS

This section aims to compare the performance of Fed-ANIDS, which is based on FL, with a centralized learning setting. Centralized learning requires all the data to be transferred to a central server for training, meaning that no computation is carried out by the clients. We conducted experiments on the three datasets previously mentioned for each variant of the autoencoder (AE, VAE, and AAE). For the centralized setting, all data was preprocessed by the central entity, which is also the one in charge of training and testing. Fig. 7 illustrates the performance comparison between Fed-ANIDS and centralized learning according to F1-score, accuracy, and FDR. In terms of both the accuracy and the F1-score, the results show that Fed-ANIDS achieves comparable or better performance than centralized learning. Specifically, the centralized performance is generally reached by Fed-ANIDS and sometimes even surpassed. Similarly for FDR, Fed-ANIDS mostly scores lower FDR than centralized learning. This demonstrates the potential of using FL for anomaly-based NIDS that successfully reaches or exceeds the centralized performance while preserving the privacy of clients. Furthermore, the use of FL in such systems also reduces the latency and bandwidth consumption since the data is processed locally on each client's device rather than being sent to a central server.

**Table 7**
Testing results of a simple AE trained on USTC-TFC2016 and CSE-CIC-IDS2018.

| Metrics | Seen datasets | | | | Unseen dataset | |
|---|---|---|---|---|---|---|
| | USTC-TFC2016 | | CSE-CIC-IDS2018 | | CIC-IDS2017 | |
| | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx |
| F1-score (%) | 80.53 | 89.27 | 74.51 | 85.44 | 48.69 | 58.18 |
| FDR (%) | 7.98 | 1.25 | 3.63 | 2.41 | 0.06 | 0.14 |
| Accuracy (%) | 83.72 | 90.70 | 80.16 | 89.95 | 72.81 | 86.76 |

**Table 8**
Testing results of a simple AE trained on CIC-IDS2017 and CSE-CIC-IDS2018.

| Metrics | Seen datasets | | | | Unseen dataset | |
|---|---|---|---|---|---|---|
| | CIC-IDS2017 | | CSE-CIC-IDS2018 | | USTC-TFC2016 | |
| | FedAvg | FedProx | FedAvg | FedProx | FedAvg | FedProx |
| F1-score (%) | 45.47 | 49.83 | 80.09 | 85.87 | 66.41 | 67.68 |
| FDR (%) | 1.31 | 1.18 | 5.28 | 4.87 | 42.07 | 43.93 |
| Accuracy (%) | 71.32 | 80.00 | 86.45 | 91.13 | 69.4 | 69.16 |

## 6. Conclusion and future work

In this paper, we introduce Fed-ANIDS an autoencoder-based approach that incorporates FL and AD for intrusion detection in distributed networks. Fed-ANIDS efficiently enables secure model training and robust intrusion detection by leveraging the benefits of FL and those of autoencoders. We build the proposed method with three variants of autoencoders, namely, simple AE, VAE, and AAE, and we adopt FedProx as the FL algorithm. With a series of experiments and evaluations conducted on three well-known flow-based datasets (USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018), we demonstrate the effectiveness of the proposed method in detecting network intrusions with high accuracy and low false alarms while preserving privacy. Additionally, Fed-ANIDS significantly outperforms other GAN-based models. We also show that the FedProx algorithm adopted in Fed-ANIDS is always on par or better than FedAvg. These results highlight the potential of using autoencoders for large-scale intrusion detection in distributed systems and pave the way to explore other FL algorithms.

Future works may include further research and evaluations of FL-based IDS models with a focus on domain generalization in order to improve their performance across different network domains and achieve better intrusion detection accuracy overall. Moreover, other FL algorithms should be explored and tested for this purpose.

## CRediT authorship contribution statement

**Meryem Janati Idrissi:** Conceptualization, Methodology, Implementation, Data preperation, Writing. **Hamza Alami:** Methodology, Writing, Reviewing, Editing. **Abdelkader El Mahdaouy:** Data preparation, Reviewing. **Soufiane Oualil:** Reviewing. **Zakaria Yartaoui:** Reviewing. **Ismail Berrada:** Supervision, Validation, Reviewing.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Meryem Janati Idrissi reports equipment, drugs, or supplies was provided by Mohammed VI Polytechnic University.

## Data availability

I would like to refer the reader to the following link to access the implementation of the paper alongside all data needed to reproduce the results that appear in our paper, Link : https://github.com/meryemJanatiIdrissi/Fed-ANIDS (github.com).

## References


Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Bhattacharya, S., et al. (2021). Federated learning for intrusion detection system: Concepts, challenges and future directions. *CoRR*, arXiv:2106.09527, URL https://arxiv.org/abs/2106.09527.

Al-Qatf, M., Lasheng, Y., Al-Habib, M., & Al-Sabahi, K. (2018). Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. *IEEE Access*, *6*, 52843–52856. http://dx.doi.org/10.1109/ACCESS.2018.2869577.

Alhajjar, E., Maxwell, P., & Bastian, N. (2021). Adversarial machine learning in network intrusion detection systems. *Expert Systems with Applications*, *186*, Article 115782. http://dx.doi.org/10.1016/j.eswa.2021.115782, URL https://www.sciencedirect.com/science/article/pii/S0957417421011507.

Ayed, M. A., & Talhi, C. (2021). Federated learning for anomaly-based intrusion detection. In *2021 international symposium on networks, computers and communications* (pp. 1–8). http://dx.doi.org/10.1109/ISNCC52172.2021.9615816.

Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. http://dx.doi.org/10.48550/ARXIV.2003.05991, URL https://arxiv.org/abs/2003.05991.

Caswell, B., & Beale, J. (2004). *Snort 2.1 intrusion detection*. Elsevier.

Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, *6*, Article 100134. http://dx.doi.org/10.1016/j.mlwa.2021.100134, URL https://www.sciencedirect.com/science/article/pii/S2666827021000670.

Donahue, J., Krähenbühl, P., & Darrell, T. (2016). Adversarial feature learning. CoRR abs/1605.09782 arXiv:1605.09782, URL http://arxiv.org/abs/1605.09782.

Elmasry, W., Akbulut, A., & Zaim, A. H. (2020). Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. *Computer Networks*, *168*, Article 107042. http://dx.doi.org/10.1016/j.comnet.2019.107042, URL https://www.sciencedirect.com/science/article/pii/S138912861930800X.

Engelen, G., Rimmer, V., & Joosen, W. (2021). Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In *2021 IEEE security and privacy workshops* (pp. 7–12). IEEE.

Farahnakian, F., & Heikkonen, J. (2018). A deep auto-encoder based approach for intrusion detection system. In *2018 20th international conference on advanced communication technology* (pp. 178–183). http://dx.doi.org/10.23919/ICACT.2018.8323688.

Garcia, S., Parmisano, A., & Erquiaga, M. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic. http://dx.doi.org/10.5281/zenodo.4743746.

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, *28*(1), 18–28. http://dx.doi.org/10.1016/j.cose.2008.08.003, URL https://www.sciencedirect.com/science/article/pii/S0167404808000692.

Ge, M., Syed, N. F., Fu, X., Baig, Z., & Robles-Kelly, A. (2021). Towards a deep learning-driven intrusion detection approach for internet of things. *Computer Networks*, *186*, Article 107784. http://dx.doi.org/10.1016/j.comnet.2020.107784, URL https://www.sciencedirect.com/science/article/pii/S138912862031358X.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27: Annual conference on neural information processing systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (pp. 2672–2680). URL https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

Habibi Lashkari, A., Draper Gil, G., Mamun, M., & Ghorbani, A. (2016). Characterization of encrypted and VPN traffic using time-related features. http://dx.doi.org/10.5220/0005740704070414.

Habibi Lashkari, A., Draper Gil, G., Mamun, M., & Ghorbani, A. (2017). *Characterization of Tor Traffic using Time based Features* (pp. 253–262). http://dx.doi.org/10.5220/0006105602530262.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In Y. Bengio, & Y. LeCun (Eds.), *2nd International conference on learning representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, conference track proceedings*. URL http://arxiv.org/abs/1312.6114.

Kuypers, M. A., Maillart, T., & Paté-Cornell, E. (2016). *vol. 30*, *An empirical analysis of cyber security incidents at a large organization. Department of Management Science and Engineering* (pp. 1–22). Stanford University, School of Information, UC Berkeley, http://fsi.stanford.edu/sites/default/files/kuypersweis_v7.pdf, accessed July.

LeCun, Y., Cortes, C., & Burges, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist.

Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, *37*(3), 50–60. http://dx.doi.org/10.1109/MSP.2020.2975749.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, & V. Sze (Eds.), *vol. 2*, *Proceedings of machine learning and systems* (pp. 429–450). URL https://proceedings.mlsys.org/paper/2020/file/38af86134b65d0f10fe33d30dd76442e-Paper.pdf.

Liu, L., Engelen, G., Lynar, T., Essam, D., & Joosen, W. (2022). Error prevalence in NIDS datasets: A case study on CIC-ids-2017 and CSE-cic-IDS-2018. In *2022 IEEE conference on communications and network security* (pp. 254–262). IEEE.

Makhzani, A., Shlens, J., Jaitly, N., & Goodfellow, I. J. (2015). Adversarial autoencoders. *CoRR*, arXiv:1511.05644, URL http://arxiv.org/abs/1511.05644.

Mattia, F. D., Galeone, P., Simoni, M. D., & Ghelfi, E. (2019). A survey on GANs for anomaly detection. *CoRR*, arXiv:1906.11632, URL http://arxiv.org/abs/1906.11632.

McMahan, H. B., Moore, E., Ramage, D., & y Arcas, B. A. (2016). Federated learning of deep networks using model averaging. *CoRR*, arXiv:1602.05629, URL http://arxiv.org/abs/1602.05629.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In A. Singh, & J. Zhu (Eds.), *Proceedings of machine learning research: vol. 54*, *Proceedings of the 20th international conference on artificial intelligence and statistics* (pp. 1273–1282). PMLR, URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., et al. (2018a). N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, *17*(3), 12–22. http://dx.doi.org/10.1109/MPRV.2018.03367731.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., et al. (2018b). N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, *17*(3), 12–22.

Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. *CoRR*, arXiv:1802.09089, URL http://arxiv.org/abs/1802.09089.

Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military communications and information systems conference (MilCIS)* (pp. 1–6). http://dx.doi.org/10.1109/MilCIS.2015.7348942.

Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., & Sadeghi, A.-R. (2019). DÏot: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th international conference on distributed computing systems* (pp. 756–767). http://dx.doi.org/10.1109/ICDCS.2019.00080.

Otter, D. W., Medina, J. R., & Kalita, J. K. (2021). A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(2), 604–624. http://dx.doi.org/10.1109/TNNLS.2020.2979670.

Park, W., & Ahn, S. (2017). Performance comparison and detection analysis in snort and suricata environment. *Wireless Personal Communications*, *94*(2), 241–252.

Qin, Y., & Kondo, M. (2021). Federated learning-based network intrusion detection with a feature selection approach. In *2021 International conference on electrical, communication, and computer engineering* (pp. 1–6). http://dx.doi.org/10.1109/ICECCE52056.2021.9514222.

Rahman, S. A., Tout, H., Talhi, C., & Mourad, A. (2020). Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network*, *34*(6), 310–317. http://dx.doi.org/10.1109/MNET.011.2000286.

Rey, V., Sánchez Sánchez, P. M., Huertas Celdrán, A., & Bovet, G. (2022). Federated learning for malware detection in IoT devices. *Computer Networks*, *204*, Article 108693. http://dx.doi.org/10.1016/j.comnet.2021.108693, URL https://www.sciencedirect.com/science/article/pii/S1389128621005582.

Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. (pp. 108–116). http://dx.doi.org/10.5220/0006639801080116.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization.. *ICISSp*, *1*, 108–116.

Sovilj, D., Budnarain, P., Sanner, S., Salmon, G., & Rao, M. (2020). A comparative evaluation of unsupervised deep architectures for intrusion detection in sequential data streams. *Expert Systems with Applications*, *159*, Article 113577. http://dx.doi.org/10.1016/j.eswa.2020.113577, URL https://www.sciencedirect.com/science/article/pii/S0957417420304012.

Tabassum, A., Erbad, A., Lebda, W., Mohamed, A., & Guizani, M. (2022). FEDGAN-IDS: Privacy-preserving IDS using GAN and federated learning. *Computer Communications*, *192*, 299–310. http://dx.doi.org/10.1016/j.comcom.2022.06.015, URL https://www.sciencedirect.com/science/article/pii/S0140366422002171.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications* (pp. 1–6). http://dx.doi.org/10.1109/CISDA.2009.5356528.

Tian, P., Chen, Z., Yu, W., & Liao, W. (2021). Towards asynchronous federated learning based threat detection: A DC-adam approach. *Computers & Security*, *108*, Article 102344. http://dx.doi.org/10.1016/j.cose.2021.102344, URL https://www.sciencedirect.com/science/article/pii/S0167404821001681.

Tsukada, M., Kondo, M., & Matsutani, H. (2020). A neural network-based on-device learning anomaly detector for edge devices. *IEEE Transactions on Computers*, *69*(7), 1027–1044. http://dx.doi.org/10.1109/TC.2020.2973631.

Wang, W., Zhu, M., Zeng, X., Ye, X., & Sheng, Y. (2017). Malware traffic classification using convolutional neural network for representation learning. (pp. 712–717). http://dx.doi.org/10.1109/ICOIN.2017.7899588.

Yang, Z., Liu, X., Li, T., Wu, D., Wang, J., Zhao, Y., et al. (2022). A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers & Security*, *116*, Article 102675. http://dx.doi.org/10.1016/j.cose.2022.102675, URL https://www.sciencedirect.com/science/article/pii/S0167404822000736.

Zavrak, S., & İskefiyeli, M. (2020). Anomaly-based intrusion detection from network flow features using variational autoencoder. *IEEE Access*, *8*, 108346–108358. http://dx.doi.org/10.1109/ACCESS.2020.3001350.

Zhang, J., & Zulkernine, M. (2006). Anomaly based network intrusion detection with unsupervised outlier detection. *vol. 5*, In *2006 IEEE international conference on communications* (pp. 2388–2393). http://dx.doi.org/10.1109/ICC.2006.255127.

Zhao, Y., Chen, J., Wu, D., Teng, J., & Yu, S. (2019). Multi-task network anomaly detection using federated learning. In *SoICT 2019*, *Proceedings of the tenth international symposium on information and communication technology* (pp. 273–279). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3368926.3369705, URL https://doi.org/10.1145/3368926.3369705.

**Meryem Janati Idrissi** received the M.S. degree in Big Data Analytics & Smart Systems from Sidi Mohamed Ben Abdellah University, Fes, Morocco, in 2018. She is currently pursuing the Ph.D. degree in Data science, Networking and Algorithmic thinking from Mohammed VI Polytechnic University, Benguerir, Morocco. Her current research interests include intrusion detection, federated learning, privacy-preserving.