Network Anomaly Detection Using Federated Deep Autoencoding Gaussian Mixture Model

Yang Chen, Junzhe Zhang, Chai Kiat Yeo

Network Anomaly Detection using Federated Deep Autoencoding Gaussian Mixture Model*

Yang Chen[0000−0001−8697−003X], Junzhe Zhang[0000−0003−1931−8046], and Chai Kiat Yeo*[0000−0002−7618−1472]

School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798
asckyeo@ntu.edu.sg

Abstract. Deep autoencoding Gaussian mixture model (DAGMM) employs dimensionality reduction and density estimation and jointly optimizes them for unsupervised anomaly detection tasks. However, the absence of large amount of training data greatly compromises DAGMM's performance. Due to rising concerns for privacy, a worse situation can be expected. By aggregating only parameters from local training on clients for obtaining knowledge from more private data, federated learning is proposed to enhance model performance. Meanwhile, privacy is properly protected. Inspired by the aforementioned, this paper presents a federated deep autoencoding Gaussian mixture model (FDAGMM) to improve the disappointing performance of DAGMM caused by limited data amount. The superiority of our proposed FDAGMM is empirically demonstrated with extensive experiments.

Keywords: Anomaly Detection · Small Dataset · Privacy-Preserving · Federated learning · Deep Autoencoding Gaussian Mixture Model · Network Security.

1 INTRODUCTION

Deep learning has been providing a lot of solutions which have previously posed big challenges to the artificial intelligence. It has been deployed in numerous applications such as computer vision, natural language processing and many other domains [1] thanks to the great advancement in computing power and the availability of vast amount of data. Much more complex and advanced algorithms can now be trained [2].

In the cybersecurity domain, anomaly detection is a critical mechanism used for threat detection and network behavior anomaly detection is a complementary technology to systems that detect security threats based on packet signatures. Network anomaly detection is the continuous monitoring of a network for unusual events or trends which is an ideal platform to apply deep learning. Deep anomaly detection [3] has thus seen rapid development such as self-taught learning based

* This project is supported by Grant No. NTU M4082227

deep learning [4] and deep autoencoding Gaussian mixture model (DAGMM [5]. Deep learning significantly improves the model complexity resulting in substantial improvement to the detection accuracy. However, deep learning requires the availability of tremendous amount of data to be well trained and supervised learning not only requires large amount of data but they must also be labelled.

The DAGMM proposed in [5] significantly improves the F1 score compared to other state-of-the-art methods including deep learning methods. It employs dimensionality reduction and feature embedding via the AutoEncoder, a compression network, and then performs density estimation via an estimation network. The entire process is unsupervised and hence no labelled data is needed for training the networks. However, DAGMM still requires a huge amount of data to train its models.

Data availability poses a huge challenge for deep learning system. Compounding the problem is that not every one has amassed huge amount of data and even if they have, the data may not be labelled or they are not to be shared collectively due to privacy and security issues. Herein lies the interest in federated learning, where model parameters instead of training data are exchanged through a centralized master model in a secured manner [6] thereby preserving the privacy of individual datasets as well as alleviating the challenge of limited datasets.

This paper proposes the federated learning assisted deep autoencoding Gaussian mixture model (FDAGMM) for network anomaly detection under the scenario where there is insufficient data to train the deep learning models. FDAGMM can thus improve the poor performance of DAGMM caused by limited dataset and its superiority is empirically demonstrated with extensive experiments. In industry scenarios, the presented solution is expected to solve the problem of lacking training data that each organizations are not willing to share in a centralized mode.

2 Related Work

2.1 Anomaly Detection

Network Anomaly Detection, also called Network Intrusion Detection, has been studied for over 20 years [7]. Network intrusion detection systems are either signature (rule)-based or anomaly-based. The former uses patterns of well-known attacks or weak spots of the system to identify intrusions whereas the latter uses machine learning methods to determine whether the deviation from the established normal usage patterns can be flagged as intrusions [8].

Machine learning methods being applied for network anomaly detection, include genetic algorithms, support vector machines (SVM), Self-organizing map (SOM), random forests, XGBoost, KNN, Naive Bayes networks, etc. However, many suffer low accuracy and high False Positive Rate (FPR). More recently, there are research in deep anomaly detection such as [3] and [4]. The use of deep learning significantly improves the model complexity and results in substantial performance improvement in terms of the various metrics, such as F1 score,

accuracy, precision, score and False Positive Rate (FPR). However, the more complex the model, the more labelled training data it needs to be well trained.

Deep Autoencoding Gaussian Mixture Model (DAGMM) is recently proposed in [5] and it produces good results with no need to label the training data. The model consists of a compression network and an estimation network which are trained end-to-end instead of using decoupled two-stage training and the standard Expectation-Maximization (EM) algorithm. The compression network is an autoencoder that embeds the feature into a low-dimension representation and meanwhile yields the reconstruction error for each input data point, which is further fed into the estimation network which acts as a Gaussian Mixture Model (GMM). It outperforms many state-of-the-art methods in terms of F1 score. However, even though it does not require labelled training data, it demands a large amount of unlabeled data in which normal users do not have or are unwilling or unable to share.

KDDCUP 99 [9] has been the most widely used dataset for the evaluation of anomaly detection methods since it was prepared by [10] in 1999. There are 5 million simulated tcpdump connection records, each of which contains 41 features and is labelled as either normal or an attack, with exactly one specific attack type. It covers attacks falling into the following 4 main categories:

- **DoS attack**: denial-of-service, e.g. smurf;
- **R2L**: unauthorized access from a remote machine, e.g. guessing password;
- **U2R**: unauthorized access to local superuser (root) privileges, e.g. various "buffer overflow" attacks;
- **Probing**: surveillance and other probing, e.g. port scanning.

2.2 Federated Learning

Federated learning is proposed by Konečný et al. [11, 12] to use the availability of privacy sensitive data stored in mobile devices to boost the power of various deep/machine learning models. In typical federated learning (FL), clients, e.g. smart phones, suffer from unstable communication. In addition, their data are unbalanced, Non-IID and massively distributed. These features distinguish FL from conventional distributed machine learning [13, 14].

As shown in Fig. 1, **Federated Learning (FL)** involves two components, i.e. central and local training. K clients indexed by k. The whole process is divided into communication rounds, in which clients are synchronously trained with local local stochastic gradient descent (SGD) on their datasets $\mathcal{P}_k$. In the central server, parameters ω^k come from the local clients, where $k \in S$, and S refers to the participating subset of m clients in each communication round. These updated parameters are then aggregated. [11, 12, 15]

The setting of federated learning follows the principle of focused collection or data minimization proposed in the White House report [16]. In this setting, local models are trained upon data that are stored in the clients. The server does not need training data which contains private information. Only client parameters are sent to the server, and they are aggregated to obtain the central model. After

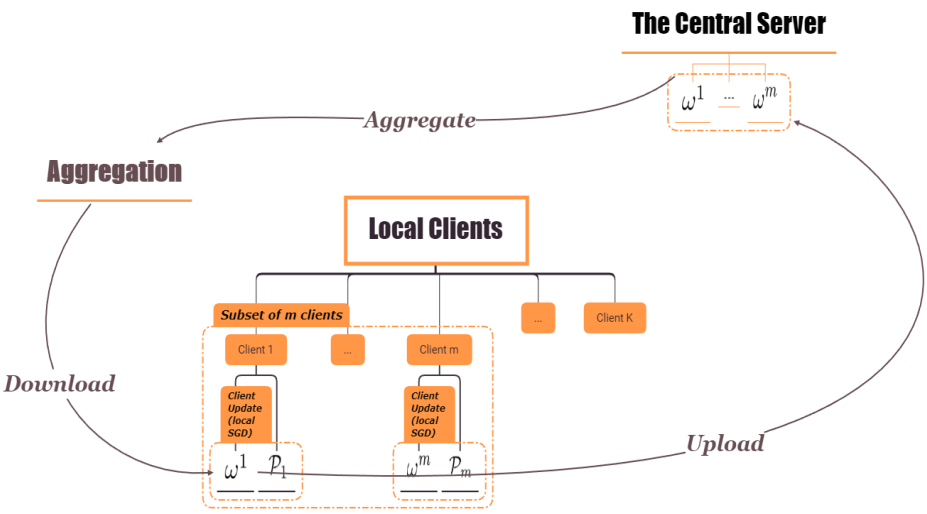


Fig. 1. Overview of Federated Learning.

communication, clients receive the aggregated updates as initial parameters of the subsequent training. Then, their models are retrained on privacy data with local stochastic gradient descent (local SGD) [11, 12, 18, 17].

In the typical FL that is executed on mobile devices, clients suffer from unstable communication. Hence it is reasonable to assume that only a subset of all clients, i.e. the aforementioned participating subset, is ready to get involved in each communication round. However, in our proposed federated deep autoencoding Gaussian mixture model (FDAGMM) for anomaly detection, the assumption does not hold anymore since the clients here commonly refer to companies or organizations with cutting-edge equipment and facilities.

2.3 Deep Autoencoding Gaussian Mixture Model

Two-step approaches that sequentially conduct dimensionality reduction and density estimation are widely used since they well address the curse of dimensionality to some extent [19]. Although fruitful progress has been achieved, these approaches suffer from a decoupled training process, together with inconsistent optimization goals, and the loss of valuable information caused by step one, i.e. the dimensionality reduction. Motivated by these, Zong et al. proposed a **Deep Autoencoding Gaussian Mixture Model (DAGMM)** [5].

As shown in Fig. 2, a *Compression Network* and an *Estimation Network* constitute DAGMM. It works as follows:

– **Dimensionality Reduction**: Given the raw features of a sample $\mathbf{x}$, the compression network which is also a deep autoencoder conducts dimension-

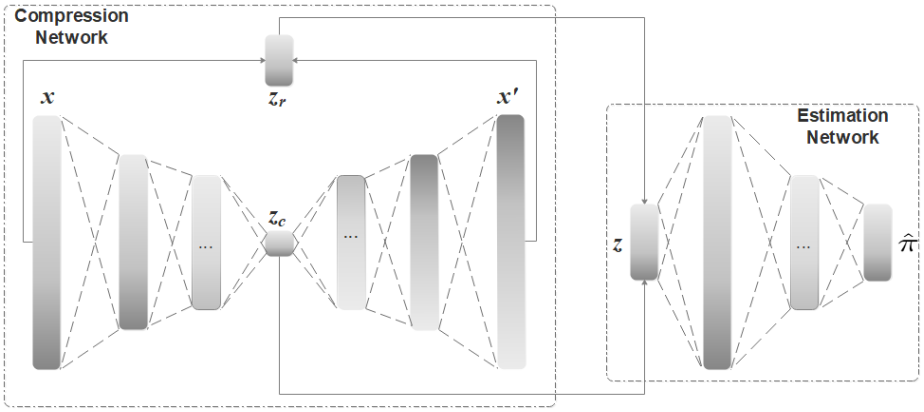


Fig. 2. Overview of Deep Autoencoding Gaussian Mixture Model.

ality reduction to output its low-dimensional representation $\mathbf{z}$ as follows:

$$\begin{aligned} \mathbf{z}_c &= h\left(\mathbf{x}; \theta_e\right) \\ \mathbf{x}' &= g\left(\mathbf{z}_c; \theta_d\right) \\ \mathbf{z}_r &= f\left(\mathbf{x}, \mathbf{x}'\right) \\ \mathbf{z} &= \left[\mathbf{z}_c, \mathbf{z}_r\right] \end{aligned} \tag{1}$$

where θ_e and θ_d are the parameters of the decoder and encoder respectively, $\mathbf{x}'$ is the reconstruction of $\mathbf{x}$ generated by the auroencoder, $\mathbf{z}_c$ is the reduced/learned low-dimensional representation, $\mathbf{z}_r$. $h(\cdot)$, $g(\cdot)$, and $f(\cdot)$ denote the encoding, decoding and reconstruction-error calculation function respectively.

– **Density Estimation**: The subsequent estimation network takes $\mathbf{z}$ from the compression network as its input. It performs density estimation with a Gaussian Mixture Model (GMM). A multi-layer neural network, denoted as $MLN(\cdot)$, is adopted to predict the mixture membership for each sample as follows:

$$\begin{aligned} \mathbf{p} &= MLN\left(\mathbf{z}; \theta_m\right) \\ \hat{\gamma} &= \text{softmax}(\mathbf{p}) \end{aligned} \tag{2}$$

where θ_m corresponds to parameters of MLN, K indicates the number of mixture components, and $\hat{\gamma}$ is a K-dimensional vector for the soft mixture-component membership prediction. Given the batch size N, $\forall 1 \leq k \leq K$, parameter estimation of GMM is further conducted as follows:

$$\begin{aligned}\hat{\phi}_k &= \sum_{i=1}^{N} \frac{\hat{\gamma}_{ik}}{N} \\ \hat{\mu}_k &= \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} \mathbf{z}_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}} \\ \hat{\mathbf{\Sigma}}_k &= \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} \left(\mathbf{z}_i - \hat{\mu}_k\right) \left(\mathbf{z}_i - \hat{\mu}_k\right)^T}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}\end{aligned} \tag{3}$$

where $\hat{\gamma}_i$ is the membership prediction, and $\hat{\phi}_k$, $\hat{\mu}_k$, $\hat{\mathbf{\Sigma}}_k$ are the mixture probability, mean, and covariance for component k in GMM respectively. Furthermore, sample energy can be inferred as:

$$E(\mathbf{z}) = -\log\left(\sum_{k=1}^{K} \hat{\phi}_k \frac{\exp\left(-\frac{1}{2}\left(\mathbf{z} - \hat{\mu}_k\right)^T \hat{\mathbf{\Sigma}}_k^{-1} \left(\mathbf{z} - \hat{\mu}_k\right)\right)}{\sqrt{\left|2\pi\hat{\mathbf{\Sigma}}_k\right|}}\right) \tag{4}$$

where $|\cdot|$ denotes the determinant of a matrix.

Based on the three components, i.e. reconstruction error of autoencoder $L\left(\mathbf{x}_i, \mathbf{x}_i'\right)$, sample energy $E\left(\mathbf{z}_i\right)$, and a penalty term $P(\hat{\mathbf{\Sigma}})$, the objective function of DAGMM is then constructed as:

$$J\left(\theta_e, \theta_d, \theta_m\right) = \frac{1}{N} \sum_{i=1}^{N} L\left(\mathbf{x}_i, \mathbf{x}_i'\right) + \frac{\lambda_1}{N} \sum_{i=1}^{N} E\left(\mathbf{z}_i\right) + \lambda_2 P(\hat{\mathbf{\Sigma}}) \tag{5}$$

3 Federated Deep Autoencoding Gaussian Mixture Model

As discussed in the previous sections, limited data samples lead to the performance deterioration of DAGMM [5]. Therefore, the motivation of the proposed FDAGMM is to address this issue by extending the data sources while preserving the data privacy of the individual clients. Under the framework of federated learning (FL), not only can FDAGMM improve its performance with more data, but privacy is appropriately protected.

The rest of Section 3 is divided into two subsections, namely, *server execution* and *client update*. The two main components of FDAGMM are introduced in the form of pseudo-codes.FDAGMM shares most of the notation as FL except that ω is replaced with θ to be consistent with DAGMM as shown:

$$\theta = \{\theta_{\{e,d\}}, \theta_m\} \tag{6}$$

where the parameters of the autoencoder include those of the encoder and the decoder, i.e. $\theta_{\{e,d\}} = \{\theta_e, \theta_d\}$, and those of the estimation network is denoted as θ_m. Moreover, superscripts and subscripts are adopted to specify client k and communication round t that the parameters belong to, i.e. θ_t^k.

3.1 Server Execution

Algorithm 1 Server Component of FDAGMM

1: **function** SERVEREXECUTION(if_two_phase) ▷ Run on the server
2: initialize θ_0
3: **for** each round $t = 1, 2, ..., T$ **do**
4: **for** each client $k \in \{1, ..., K\}$ **in parallel do**
5: $\theta^k \leftarrow$ ClientUpdate$(k, \theta_t, flag)$
6: **end for**
7: $\theta_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} * \theta^k$
8: **end for**
9: **end function**

Algorithm 1 shows the *Server Execution* that is carried out on the central server. It consists of an initialization operation followed by T communication rounds. In initialization (Line 2), ω_0 is initialized.

Under the FL framework, the training process consists of the communication rounds that are indexed with t (Line 3). Lines 4-6 call sub-function *Client Update* for K clients in parallel. Then in line 7, the aggregation is performed to update θ, which is the parameter of the centre model. n and n_k indicate the number of instances belonging to client k and the total number of all involved samples, respectively.

3.2 Client Update

Algorithm 2 Client Component of FDAGMM

1: **function** CLIENTUPDATE($k, \theta, flag$) ▷ Run on client k
2: $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size N)
3: **for** each local epoch i from 1 to E **do**
4: **for** batch $b \in \mathcal{B}$ **do**
5: $\theta \leftarrow \theta - \eta * \bigtriangledown J(\theta; b)$
6: **end for**
7: **end for**
8: return θ to server
9: **end function**

Client Update (**Algorithm 2**) takes k and θ as its input. k indexes a specific client and θ denotes the parameters of the central model in the current round. E denotes the local epoch. Line 2 splits data into batches, whereas Lines 3-7 train the local DAGMMs by batch with private data stored on each client. η denotes the learning rate; $J(\cdot)$ is the loss function. Its definitions is detailed in Equation **(5)**. Line 8 returns the updated local parameters.

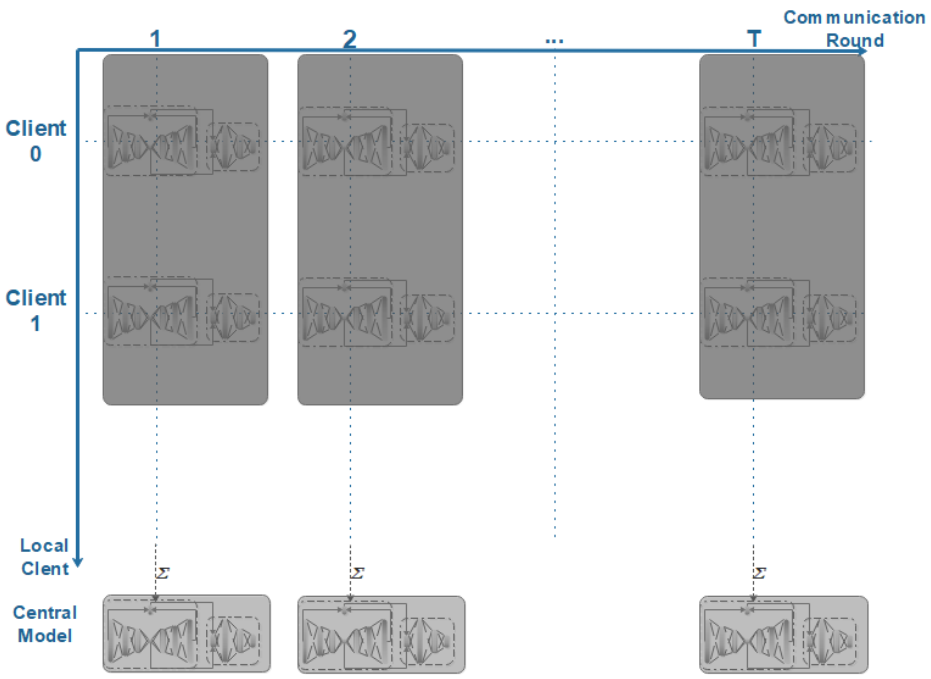


Fig. 3. Aggregation of Federated Deep Autoencoding Gaussian Mixture Model.

Fig. 3 shows an example illustrating the aggregation of FDAGMM. The abscissa and ordinate denote the communication round and the local client respectively. Two local devices, i.e. *Client 0*, *Client 1*, and a *server* are involved. The cross located at (*Clint 0*; T) indicates that *Clint 0* is participating in updating the central model in round T.

4 Experimental Results and Analysis

4.1 Experimental Design

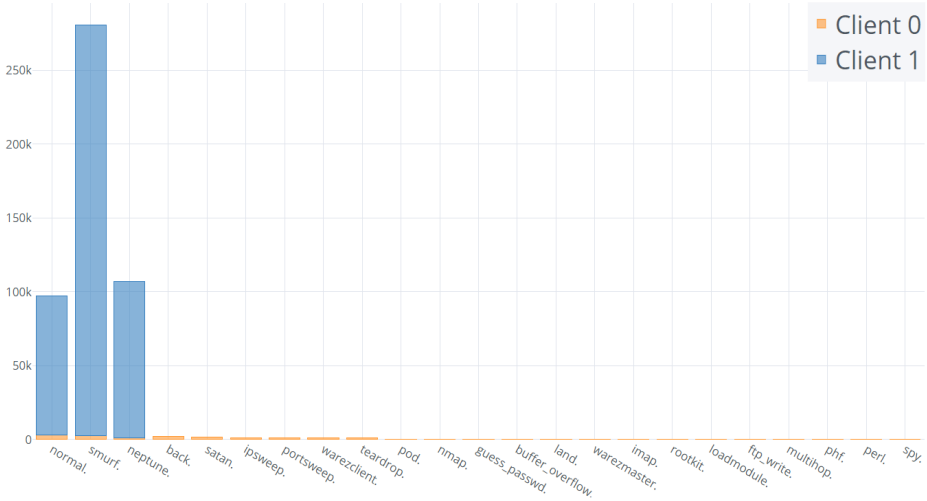
Due to the lack of public datasets for anomaly detection, especially the Intrusion Detection Systems (IDSs), very few of them can be directly adopted in the evaluation of the proposed FDAGMM.

We perform extensive experiments with a public dataset KDDCUP 99, which is the most widely used in the evaluation of various anomaly detection approaches and systems. Table 1 shows the statistics of KDDCUP 99.

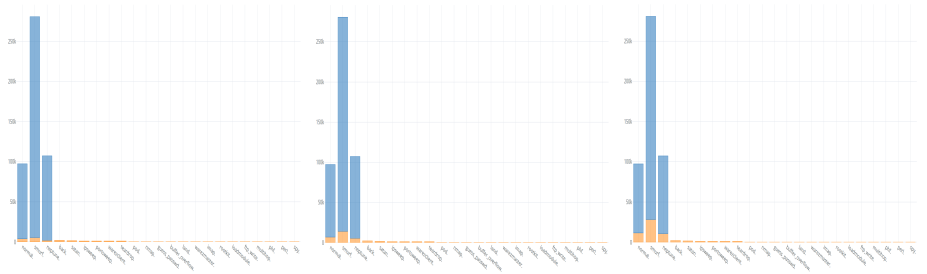
Data Pre-processing Constructing datasets to simulate the private data of the individual clients which fulfill the associated requirements is thus needed for this study. In the FL setting, private datasets stored in clients are Non-IID and

Table 1. Statistics of KDDCUP 99

# Dimensions	# Instances	Anomaly Ratio
120	4898431	19.86%



(a) Scenario 1: Client 0 with 1% common and 100% rare attacks.



(b) Scenario 2: Client 0 with 2% common and 100% rare attacks.

(c) Scenario 3: Client 0 with 5% common and 100% rare attacks.

(d) Scenario 4: Client 0 with 10% common and 100% rare attacks.

Fig. 4. Stacked bar-charts of attack types belong to two clients.

unbalanced. In the experiments, the whole KDDCUP 99 dataset is split into two sets belonging to *Client 1* and *Client 2* through selecting records in the complete KDDCUP. These two clients play the roles of two companies in which Client 1 with limited data asks for help from Client 2 under the framework of FDAGMM.

Attack instances are separated according to their types. Training sets only include half of the attack samples belonging to its client, while test sets consist of both normal and the other half attack instances. The data belonging to *Client 1* are similar in anomaly ratio as the other client. Since there is a very sharp

distinction between rare and common attacks, which is reflected in Fig. 4, *smurf* and *neptune* are included as **Common Attacks** and the rest comprises the **Rare Attacks**. The details are shown in Table 2. The experiments are expected to prove that *Client 1* can improve its performance with the help of *Client 2*.

Table 2. Common and Rare Attacks

	Types
Common Attacks	smurf, neptune
Rare Attacks	satan, ipsweep, portsweep, nmap, back, warezclient, teardrop, pod, guess_passwd, buffer_overflow, land, warezmaster, imap, rootkit, loadmodule, ftp_write, multihop, phf, perl, spy

As shown in Fig. 4 (a), the scenario with *Client 1* holding 1% common and 100% rare (bars in orange) attack instances and *Client 2* holding the rest, i.e. 99% common attacks (bars in blue), is denoted as **Scenario 1**. Those corresponding to the remaining three figures, (b) to (d) are denoted as **Scenario 2**, **Scenario 3** and **Scenario 4** respectively.

Table 3. Default setting of DAGMM

Notion	**Parameter Range**
$\eta^\star$	0.0001
N^*	1024
$\lambda_1^\dagger$	0.1
$\lambda_2^\dagger$	0.005

* Learning Rate
* Batch Size
† Involved in Equation (5)

Parameter Setting on DAGMM For the purpose of a fair comparison, all these experiments adopt the default DAGMM setting as the hyper-parameters of the local models in the proposed FDAGMM. The settings are summarized in Table 3.

4.2 Experiments on KDDCUP 99

The experiments are designed to evaluate the effectiveness of the proposed FDAGMM. According to their attack types, i.e. rare or common, instances making up each training set belong to two clients. Four scenarios with distinct combinations of attacks are considered and illustrated in Fig. 4. Three metrics are

adopted to measure the performance of the compared algorithms. They are **Precision**, **Recall**, and **F1-Score**.

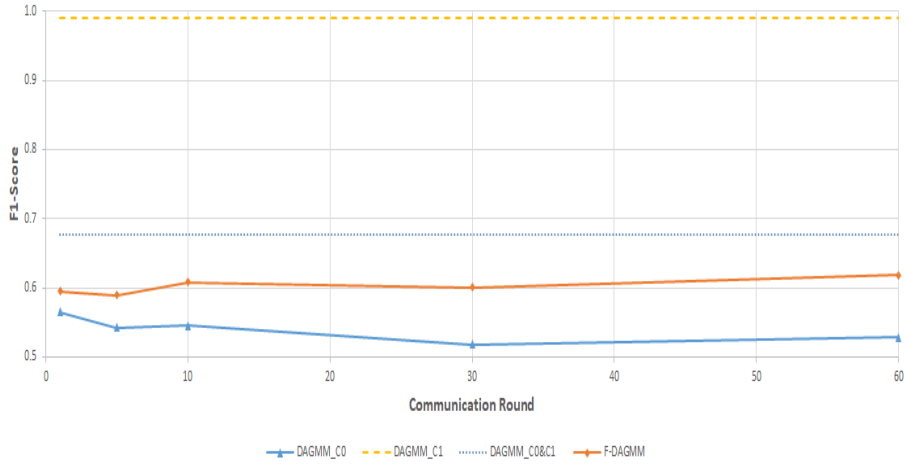
All the experiments are run independently for five times; each time, the random seed is fixed. The average (AVG) values of F1-Score are presented in Fig. 5. Tables 4 and 5 show the AVG and standard deviation (STDEV) of precision and recall values respectively. In the tables, the AVG value is listed before the standard deviation in parentheses.

The names of the algorithms compared in Tables 4 and 5 indicate not only the adopted technique but also the involved training set, i.e. Training set from *Client 0* or *Client 1*.

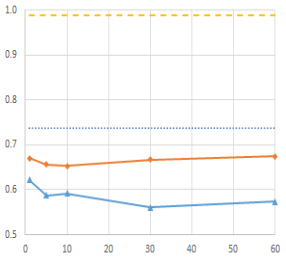
- **DAGMM_C0** employs DAGMM and trains the model with only data from *Client 0*. Test set comprises half of the attack instances belonging to *Client 0* and the corresponding proportion of normal samples.
- **DAGMM_C1** employs DAGMM and trains the model with only data from *Client 1*. Test set comprises half of the attack instances belonging to *Client 1* and the corresponding proportion of normal samples. *DAGMM_C1* thus complements the training on the limited data of *Client 0* to increase the detection performance on *Client 0* without comprising the privancy of its data.
- **DAGMM_C0&C1** employs DAGMM and trains the model with a mixture of the data from two clients. Test set comprises half of the attack instances belonging to *Client 0* and the corresponding proportion of normal samples. *DAGMM_C0&C1 (Ideal Bound* denotes where the performance limit of FDAGMM is. This is under the ideal scenario where both clients are willing to share their data so that DAGMM can be trained to achieve the best performance.
- **FDAGMM** includes two clients, and each employs a DAGMM and trains the model with only data from itself. Test set comprises half of the attack instances belonging to *Client 0* and the corresponding proportion of normal samples. The performance of FDAGMM reflects how much help *Client 0* can receive from *Client 1* under the FL framework. In other words, how close the presented FDAGMM can approach the ideal performance limit, i.e. DAGMM_C0&C1 (Ideal Bound) where there is an abundance of data for training and clients are willing to contribute their data for collective training of the DAGMM.

Table 4. Comparative studies on FDAGMM: Precision.

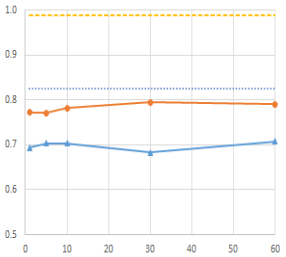
Scenario	DAGMM_C0	FDAGMM	Ideal Bound	DAGMM_C1
Scenario 1	0.397(0.1304)	**0.4532(0.0317)**	0.5366(0.0215)	0.9881(0.0024)
Scenario 2	0.4793(0.1712)	**0.5116(0.0289)**	0.6266(0.0444)	0.9787(0.0073)
Scenario 3	0.6069(0.0872)	**0.6553(0.0403)**	0.7261(0.0612)	0.9832(0.0129)
Scenario 4	0.6735(0.0883)	**0.7447(0.0369)**	0.7583(0.0302)	0.9813(0.0018)



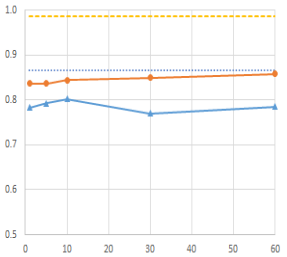
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

Fig. 5. Comparative studies on FDAGMM on KDDCUP 99.

Table 5. Comparative studies on FDAGMM: Recall.

Scenario	DAGMM_C0	FDAGMM	Ideal Bound	DAGMM_C1
Scenario 1	0.9263(0.1434)	**0.9769(0.0459)**	0.9222(0.0422)	0.9952(0.0097)
Scenario 2	0.8166(0.127)	**0.9927(0.0121)**	0.9409(0.0488)	0.9978(0.0044)
Scenario 3	0.8777(0.0698)	**0.9825(0.0284)**	0.9341(0.0482)	0.9984(0.0016)
Scenario 4	0.9237(0.038)	**0.9803(0.0264)**	0.976(0.0334)	0.9987(0.0019)

Based on the results shown in these tables and figures, the following observations can be made:

- The proposed FDAGMM outperforms DAGMM on all metrics, i.e. F1-Score, Precision and Recall for all four scenarios.
- According to its lower STDEV values, FDAGMM's performance is more stable than DAGMM.

– The more Non-IID and unbalanced the data distribution is across clients, the more challenging the scenario tends to be, which is reflected by the blue dotted lines corresponding to DAGMM_C0&C1 in Fig. 5.

5 CONCLUSION

With the help of other clients holding sufficient feature-similar records under the FL framework, we show that the less than satisfactory performance of DAGMM suffering from limited dataset can be addressed and improved using FDAGMM. Empirical studies comparing the performance of the proposed FDAGMM and DAGMM under four distinct scenarios demonstrate the superiority of the FDAGMM in terms of all the associated performance metrics.

This study follows the assumption that all local models adopt the same neural networks architecture and share the same hyperparameters, which implies all the involved data records share the same feature structure. This renders FDAGMM to be less versatile to be deployed to other application domains. In future research, we are going to develop new federated learning assisted DAGMM address the weakness.

References

1. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
2. J. Zhang, S. H. Yeung, Y. Shu, B. He, and W. Wang, "Efficient memory management for gpu-based deep learning systems," *arXiv preprint arXiv:1903.06631*, 2019.
3. R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.
4. A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICST (Institute for Computer Sciences, Social-Informatics and , 2016, pp. 21–26.
5. B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," 2018.
6. H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.
7. F. Y. Edgeworth, "Xli. on discordant observations," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 23, no. 143, pp. 364–375, 1887.
8. C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin, "Intrusion detection by machine learning: A review," *expert systems with applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.
9. M. LLC. (1999) MS Windows NT kdd cup. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/task.html

10. S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the jam project," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2. IEEE, 2000, pp. 130–144.
11. J. Konečný, B. McMahan, and D. Ramage, "Federated Optimization: Distributed Optimization Beyond the Datacenter," *arXiv Prepr. arXiv1511.03575*, no. 1, pp. 1–5, 2015.
12. J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," *CoRR*, vol. abs/1610.0, no. NIPS, pp. 1–5, 2016.
13. C. Ma, J. Konečnỳ, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takáč, "Distributed optimization with arbitrary local solvers," *Optimization Methods and Software*, vol. 32, no. 4, pp. 813–848, 2017.
14. S. J. Reddi, J. Konečnỳ, P. Richtárik, B. Póczós, and A. Smola, "Aide: fast and communication efficient distributed optimization," *arXiv preprint arXiv:1608.06879*, 2016.
15. Y. Chen, X. Sun, and Y. Jin, "Communication-Efficient Federated Deep Learning with Asynchronous Model Update and Temporally Weighted Aggregation," *arXiv preprint arXiv:1903.07424*, 2019.
16. W. House, "Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy," *White House, Washington, DC*, pp. 1–62, 2012.
17. Y. Chen, X. Sun, and Y. Hu, "Federated Learning Assisted Interactive EDA with Dual Probabilistic Models for Personalized Search," in *International Conference on Swarm Intelligence*, 2019, pp. 374–383.
18. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.
19. E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011.