Communication Costs Analysis of Unsupervised Federated Learning: an Anomaly Detection Scenario

Mirko Nardi[1,2], Lorenzo Valerio[2] and Andrea Passarella[2]

[1]Scuola Normale Superiore, Piazza dei Cavalieri 7, Pisa, 56126, Italy
[2]IIT-CNR, Via Giuseppe Moruzzi 1, Pisa, 56124, Italy

Abstract

The rapid growth of distributed data across edge devices has prompted the development of decentralized machine learning techniques, such as Federated Learning (FL), to address privacy and data transfer concerns. Only a few recent works have focused on unsupervised FL approaches compared to their supervised counterparts, with the consequence that many aspects of these solutions, e.g., the communication cost, have not been thoroughly investigated. In this paper, we analyse the communication cost associated with unsupervised federated anomaly detection, focusing on a proposed method where clients are grouped into communities based on inlier patterns and subsequently train autoencoder models in a federated fashion. Our analysis quantifies the communication overhead introduced by the federated learning process and compares it to traditional centralized approaches for anomaly detection. We also explore potential trade-offs between communication cost, privacy, and model performance. Our findings reveal that the unsupervised federated approach can achieve a significant reduction in communication cost (up to 83.33%) with comparable performance, by selecting better-suited models. Furthermore, the adjustments we implement render the methodology independent of dataset size, offering notable privacy benefits and competitive accuracy performance, making it highly effective in industrial scenarios with large local datasets and a moderate number of clients.

Keywords

federated learning, unsupervised, anomaly detection, communication cost analysis

1. Introduction

The capacity to obtain valuable insights from Big Data through AI techniques is a key component for the advancement of Industry 4.0 [1]. In this scenario, robots near production lines gather, process, and utilize data generated during their operations. These robots have built-in computing and communication capabilities for AI tasks such as identifying potential failures or product defects and coordinating with one another.

The prevalent AI paradigm is centralized, where data gathered by edge devices are transmitted to the cloud for AI model training. However, due to the explosive growth of data generated at the edge and heightened concerns about data privacy and ownership [2], there is a shift towards relocating AI processes to the edge, transitioning the paradigm towards a more distributed or decentralized approach.

Federated learning (FL)[3] has arisen as an attractive method for training machine learning models without the need to share raw data among different clients. This decentralized framework maintains data privacy, reduces communication overhead, and facilitates more scalable training. Challenges in federated learning encompass managing heterogeneous data distributions among clients[4] and devising efficient solutions.

This study concentrates on unsupervised federated learning, specifically targeting the enhancement of federated anomaly detection for mobile edge devices. We examine and refine a federated anomaly detection method [5] that employs global information to boost performance while minimizing communication overhead. Our approach is not only suitable for preserving privacy and addressing network resource constraints, but also designed for utilizing tiny ML models on individual nodes.

Unlike the reference work, our focus lies on analyzing communication costs and selecting suitable models to attain competitive results with reduced overhead. By opting for the appropriate model architecture on the Fashion-MNIST dataset, we successfully improve the methodology's performance, achieving an 83.33% reduction in communication cost. This makes it a highly effective solution for industrial settings involving large local datasets and a moderate number of clients, offering a robust alternative to conventional centralized methods.

2. Related Works

Federated Learning (FL) is a distributed learning framework that optimizes computing power and data management on edge devices, and has quickly advanced since

Ital-IA 2023: 3rd National Conference on Artificial Intelligence, organized by CINI, May 29–31, 2023, Pisa, Italy
mirko.nardi@sns.it (M. Nardi); lorenzo.valerio@iit.cnr.it (L. Valerio); andrea.passarella@iit.cnr.it (A. Passarella)
0000-0002-8689-7976 (M. Nardi); 0000-0001-5574-7847 (L. Valerio); 0000-0002-1694-612X (A. Passarella)

its introduction [3]. FL provides a modern and more effective evolution to traditional distributed paradigms [6, 7, 8, 9, 10] and has been deployed in various use-cases by major service providers [11, 4, 12].

Unsupervised FL has been relatively less studied, with a few notable works in the field. For instance, Van [13] first introduced unsupervised representation learning in a federated setting, but without considering typical distributed setting issues. Zhang et al.[14] addressed the inconsistency of representation spaces and misalignment of representations in unsupervised FL. Tzinis et al.[15] proposed an unsupervised FL approach for speech enhancement and separation, utilizing a small portion of supervised data to boost the main unsupervised task. In [16], Luo and colleagues presented a collaborative system of autoencoders for distributed anomaly detection, but with local models used for inference only. Mothukuri et al. [17] proposed a FL-based anomaly detection approach for intrusion identification and classification in IoT networks. Lubana et al.[18] proposed Orchestra, a globally consistent clustering technique for unsupervised FL that guarantees good generalization performance. Han et al. [19] introduced FedX, an unsupervised FL framework that employs a two-sided knowledge distillation with contrastive learning, improving performance on five unsupervised algorithms.

3. Reference Methodology and Cost Model

We use the distributed anomaly detection framework presented in [5] as our reference, revisiting its methodology and analyzing communication costs.

3.1. Methodology Description

We study a distributed learning system with clients M and data distributions C, where $|C| \leq |M|$. Each client obtains a fraction d of its samples from $C_{out} \in C \setminus C_{in}$, with $C_{in} \neq C_{out}$, and the remaining $(100-d)\%$ from $C_{in} \in C$. Typically, $d \in [5\%, 15\%]$ is considered realistic and is commonly used in anomaly detection contexts [20]. The methodology's goal is to create consistent groups of clients and perform standard federated learning within those groups.

In the subsequent sections, we revisit the two-phase structure that constitutes the entire process, (for more details, please refer to the reference work [5]).

3.1.1. Preprocessing Phase: Group Identification

This phase aims to enable clients to join a group with the same (or similar) majority class C_{in}. Clients train a lightweight anomaly detection model on local data, and pairs of clients exchange models to classify their local data. If nodes have similar inlier/outlier ratios using each other's models, they share the same inlier class and should be in the same group.

An undirected graph is generated using candidate groups from each client, and a community detection algorithm is applied to identify groups of nodes for the upcoming standard FL step.

3.1.2. Federated Learning Outlier Detection

After the first phase, k groups (or communities) $G_0, \ldots, G_k$ are formed. For each group, federated learning is initiated using autoencoders as models. Autoencoders are suitable because they naturally fit the FL framework and can effectively be used in AD tasks. The Federated Averaging (FedAvg) protocol is used for FL [3]. At the end of each communication round, the trained autoencoder is shared among the clients of the same group.

3.2. Cost of Decentralized Anomaly Detection

3.2.1. Phase I

Firstly, each client trains a model on its local data and exchanges it with the other clients. Given $|M| = n$ clients and letting S_m be the size of a single local model, the total communication cost for exchanging models between all pairs of clients can be estimated as $n(n-1) \times S_m$. Clients then share pairwise association information, adding a cost of $n(n-1) \times S_r$, with $S_r = 1$ bit.

Clients share candidate groups with a single client, who builds the graph, runs the community detection algorithm, and sends back the community information. A predefined policy selects the client for this task, like the one with the lowest ID.

Assuming the average candidate group size is G_m and each client ID size is S_{id}, the total communication cost for sharing candidate groups to the selected client can be estimated as $(n-1) \times G_m \times S_{id}$. G_m can vary depending on the performance of the first step. In the worst case, G_m is equal to the number of clients, i.e., $|G_i| = n$. In the ideal case, G_m is equal to the average size of the real groups, where each group is a set of nodes sharing the same data distribution. Assuming that the average number of nodes in each group is $p \geq 1$, it holds that in the ideal case $G_m = p = n/|C|$.

In our experiments, we observed that G_m typically aligns with this ideal condition, demonstrating our methodology's effectiveness in grouping clients with similar data distributions. After the community graph has been computed, an additional cost of $(n-1) \times G_m \times S_{id}$ is required for sending community information back to

the clients. Therefore, the overall communication cost for the first phase can be summarized as:

$$P_1 = n(n-1)S_m + n(n-1)S_r + 2(n-1)\frac{n}{|C|}S_{id} \quad (1)$$

The first phase's communication cost is dominated by the quadratic term $n(n-1)$. As the number of clients grows, this impacts the cost significantly. The model size, S_m, also becomes a dominant component in the cost compared to other information types. Thus, the first phase's communication cost can be expressed as $O(n^2 S_m)$, emphasizing the model size's importance in the overall cost.

3.2.2. Phase II

In the second phase of the methodology, each group (community) $G_0, \dots, G_k$ starts a federated learning instance using a corresponding model $U_0, \dots, U_k$ (they all have the same architecture). The communication cost analysis in this phase involves local model updates, model aggregation, and global model update distribution. For simplicity, we firstly consider a single group and an external aggregator. Each client trains their model on local data and computes updates. The size of these updates depends on the model architecture, and we denote it as S_u. Clients share local updates with the aggregator, which combines these updates using the Federated Averaging algorithm. The communication cost for sending local model updates is $|G_i| \times S_u$ for each group G_i, where $|G_i|$ is the number of clients in group i. Afterward, the aggregator distributes the global model update to all clients in the group, with a communication cost of $|G_i| \times S_u$.

The total communication cost for the second phase is the sum of the costs for all groups. Given r communication rounds in each FL instance and assuming there are k groups, the formula for the total communication cost in the second phase is:

$$P_2 = r \sum_{i=0}^{k} 2|G_i|S_u \quad (2)$$

Considering that the sum over $|G_i|$ counts all the clients in the system, and that selecting a client within the group as the aggregator slightly improves the communication cost (i.e., that client does not need to send its update), we can rewrite the total communication cost formula for the second phase as follows:

$$P_2 = 2r(n-k)S_u \quad (3)$$

Where n is the number of clients. Asymptotically, the communication cost in the second phase is linear with respect to the number of clients and the size of the local model updates.

3.3. Cost of Centralized solution

On the other hand, in a centralized solution, all clients send their local datasets to a central server for processing, and the server trains a global model on the entire dataset D. The communication cost in the centralized approach is linearly dependent on the dataset size. Clients send their entire local datasets, D_i, to the server, resulting in a communication cost of $\sum_{i=1}^{n} |D_i| \times S_d$, where S_d represents the size of a single data point.

Recalling that the cost of the proposed decentralized solution is quadratic in the number of clients and linear in the local model size, the choice between the proposed methodology and a centralized solution depends on the specific requirements and priorities of a given application. For instance, when dealing with large datasets, the proposed methodology is likely more efficient not only in terms of privacy of data, but also in terms of communication. When, instead, data are very small in size, a decentralised solution may generate higher traffic than a centralised one. In such cases, it is crucial to utilize relatively light models in the decentralized approach to further optimize communication costs and ensure its effectiveness in comparison to the centralized solution.

4. Experimental Results

We evaluate the proposed methodology using the fashion-MNIST [21] dataset and focus on the communication aspect. We follow the experimental setup in [5], with minor variations.

The fashion-MNIST dataset has ten classes ($|C| = 10$). We ensure that clients have numerically balanced and disjoint datasets. We set $p = 9$, representing the number of clients within the same data distribution (class). The ideal partitioning we aim to find consists of $k = |C| = 10$ groups with p clients each. We only consider the case of $p = 9$ for this evaluation.

The models used here and in the reference work are presented in Table 1. All the number of parameters reported for the models refer to the input of MNIST-like datasets, i.e., 784 dimensions.

In contrast to our prior work [5], in the first phase, we replace OC-SVM [22] with a small convolutional autoencoder to break the dependence on the data size (thus reducing the communication cost), while maintaining satisfactory performance (in terms of detection accuracy). Notably, the OC-SVM model does not have a fixed number of parameters, as it depends on the support vectors found during the training process; hence, the number of parameters is proportional to the local dataset D_i. In the second phase, we transition from a flat autoencoder to a more extensive convolutional autoencoder, which enables better representation of the underlying structure

Table 1
Models tested

Phase	Model	Configuration	Number of Parameters
1	OC-SVM	RBF kernel, $\nu = 0.1$	$\propto D_i$
1	Convolutional Autoencoder (5 layers)	See appendix A	≈ 4.000
2	Flat Autoencoder	[64,32,64]	≈ 100.000
2	Convolutional Autoencoder (7 layers)	See appendix A	≈ 28.000

in image data and enhances communication efficiency as well as generalization in the federated learning context.

4.1. Group Detection

In this section, we present the experimental results of the group detection process for the Fashion-MNIST dataset. The results shown here concern only to the convolutional models; similar findings regarding communities were observed in our prior work[5], albeit at a higher communication cost.

Let $m_{C_i,j}$ be the j-th client with majority class C_i; we define I_{C_i} as the ideal set of clients having the same majority class C_i, e.g., $I_0 = m_{0,0}, \dots m_{0,p-1}$. In table 2, we observe that most of the communities found for fashion-MNIST consist of clients with the same majority class, such as G_0 with I_6, G_1 with I_8, and so on. An exception is given by G_6, which is formed by clients with majority classes I_2, I_0, I_3, and I_4. This indicates that there is a higher degree of similarity between the clients' data distributions in these majority classes.

Table 2
Community detection for Fashion-MNIST

Community ID	Members
G_0	I_6
G_1	I_8
G_2	I_1
G_3	I_5
G_4	I_7
G_5	I_9
G_6	$I_2 \cup I_0 \cup I_3 \cup I_4$

4.2. Federated Outlier Detection

We compare our implementation with the same methodology and test protocol as in our prior work [5]. For instance, we take two baselines as reference: (i) local, where clients only train on local data; and (ii) `ideal`, in which a client $m_{C_i,j}$ uses the model trained through federated learning on the set of clients I_{C_i}, i.e., the set of the clients sharing the same majority class. The test samples for each client are randomly sampled from the fashion-MNIST test set, following the same inlier/outlier classes and the ratio of the corresponding client.

The results presented in Table 3 show the average AUC-ROC scores across clients having the same inlier class. Our proposed configuration outperforms the `local` baseline in all inlier classes, indicating that the federated outlier detection approach effectively leverages global information to improve the model's performance. Furthermore, our results are consistently close to the performance of the `ideal` baseline, demonstrating the potential of the methodology to achieve near-optimal results.

This positive performance trend is also evident for clients that are trained within the same federation. For instance the large federation of clients with classes 0, 2, 3, and 4 demonstrates competitive performance compared to the `ideal` baseline, where a federated instance is run for each ideal group.

In our prior work [5], mean AUC-ROC values of 0.714, 0.761, and 0.772 were obtained using the OCSVM and the flat autoencoder model shown in Table 1. These models require higher communication overhead compared to the models we propose. Here we achieves comparable results with lighter models, as demonstrated by the mean AUC-ROC of 0.649, 0.728, and 0.740 in Table 3. This highlights the efficiency and effectiveness of proper model selection, as it reduces communication overhead while maintaining competitive performance.

4.3. Communication Costs

To calculate the actual communication costs for the two phases of the methodology, we can use Eq.(1) and Eq.(2). We will consider each parameter to be represented using 32-bit floating-point numbers. Therefore, the size of a model (S_{id} or S_u) is determined by the number of its parameters multiplied by 4 bytes.

In our experiment with 90 clients, each client ID (i.e., S_{id}) can be represented using a 2-byte unsigned integer. Regarding OCSVM, the size of the model is linearly dependent on the size of the client's local dataset D_i. On average, in our test, a trained OCSVM model has a size of $0.3D_i$, where D_i is approximately 1/90 of the fashion-MNIST train set. The fashion-MNIST train set is about 47.04 MB in total, resulting in an average D_i of approximately 0.522 MB.

Table 3
AUC-ROC mean per inlier class for Fashion-MNIST. Last line refers to the mean AUC-ROC values of the reference work [5] where different models are used.

Inlier	Local	Our Method	Ideal
0	0.672	0.664	**0.753**
1	0.950	0.946	**0.953**
2	0.485	0.670	**0.688**
3	0.713	0.726	**0.748**
4	0.583	0.726	**0.732**
5	**0.601**	0.592	0.599
6	0.512	0.704	**0.707**
7	0.849	**0.873**	0.869
8	0.395	**0.567**	0.550
9	0.734	**0.809**	0.800
Std Dev	0.012	0.013	0.014
Mean	0.649	0.728	**0.740**
Mean [5]	0.714	0.761	0.772

For the second phase, we perform the computation considering 6 groups consisting of 9 clients each and 1 group with 36 clients. We assume 10 communication rounds (r = 10).

The final costs for the tested models are presented in Table 4. With respect to the models used in [5], we achieve a significant reduction of approximately 83.33% in communication cost, with comparable performance, by selecting better-suited models. More importantly, we are no longer dependent on the dataset size, as is the case in the reference work and when centralizing the data. It is worth noting that the cost of data centralization in our example appears moderate (the fashion-MNIST train set is around 45.04 MB); however, local datasets in an industrial context can be enormous. Therefore, by using the right models, the methodology can be highly effective.

Table 4
Communication cost of the models tested

	Phase	Comm. Cost (MB)
This work	1	128.2
	2	201.6
Our prior work	1	1256.4
	2	720
Centralized		45.04

5. Conclusions and Future Work

In this paper, we analyzed a decentralized and unsupervised anomaly detection solution [5] from the perspective of communication costs. We derived closed formulas to estimate the costs and discussed model choices for efficient anomaly detection in terms of accuracy and communication cost.

Our results demonstrated that the proposed methodology could detect similar communities to the reference work while significantly reducing communication costs by approximately 83.33%, with comparable performance. Importantly, the communication cost reduction is independent of the dataset size, unlike in the reference work or centralized data approaches.

The methodology is well-suited for distributed industrial applications with a moderate number of clients, as its communication cost is quadratic with respect to the number of clients ($O(n^2 S_m)$). By using smaller models relative to the dataset size, this approach is more convenient than centralizing the data and is highly effective in various contexts, particularly when centralized data storage is impractical or restricted by privacy constraints.

In future work, we aim to explore the potential of other models and techniques to further optimize communication costs (e.g., remove the quadratic complexity with respect to the number of clients) while maintaining high accuracy in decentralized anomaly detection. We also plan to investigate the applicability of our methodology to a broader range of industrial scenarios and datasets, such as video or IoT sensor data. Additionally, the development of adaptive communication strategies to dynamically adjust the number of communication rounds or model selection based on the current network conditions or dataset properties could be an interesting avenue for further research.

Acknowledgments

This work has been partly funded under the H2020 MARVEL (grant 957337) and CHIST-ERA SAI (grant CHIST-ERA-19-XAI-010) projects. The work of A. Passarella has been partly funded by PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI", funded by the European Commission under the NextGeneration EU programme.

References

[1] M. Khan, X. Wu, X. Xu, W. Dou, Big data challenges and opportunities in the hype of industry 4.0, in: 2017 IEEE International Conference on Communications (ICC), IEEE, 2017, pp. 1–6.

[2] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, Z. Lin, When machine learning meets privacy: A survey and outlook, ACM Computing Surveys (CSUR) 54 (2021) 1–36.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial intelligence and statistics, PMLR, 2017, pp. 1273–1282.

[4] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan, et al., Towards federated learning at scale: System design, arXiv preprint arXiv:1902.01046 (2019).

[5] M. Nardi, L. Valerio, A. Passarella, Anomaly detection through unsupervised federated learning, arXiv preprint arXiv:2209.04184 (2022).

[6] K. Ota, M. S. Dao, V. Mezaris, F. G. B. D. Natale, Deep learning for mobile multimedia: A survey 13 (????) 1–22. URL: https://dl.acm.org/doi/10.1145/3092831. doi:10.1145/3092831.

[7] K. Chahal, M. S. Grover, K. Dey, A hitchhiker's guide on distributed training of deep neural networks (????). URL: http://arxiv.org/abs/1810.11787. arXiv:1810.11787.

[8] T. Ben-Nun, T. Hoefler, Demystifying parallel and distributed deep learning: An in-depth concurrency analysis (????). URL: http://arxiv.org/abs/1802.09941. arXiv:1802.09941.

[9] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, J. S. Rellermeyer, A survey on distributed machine learning 53 (????) 30:1–30:33. URL: https://doi.org/10.1145/3377454. doi:10.1145/3377454.

[10] T. Tuor, S. Wang, K. K. Leung, K. Chan, Distributed machine learning in coalition environments: overview of techniques, in: 2018 21st International Conference on Information Fusion (FUSION), IEEE, 2018, pp. 814–821.

[11] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, F. Beaufays, Applied federated learning: Improving google keyboard query suggestions, arXiv preprint arXiv:1812.02903 (2018).

[12] W. A. Group, Federated learning white paper v1, ???? URL: https://aisp-1251170195.cos.ap-hongkong.myqcloud.com/fedweb/1552917186945.pdf.

[13] B. van Berlo, A. Saeed, T. Ozcelebi, Towards federated unsupervised representation learning, in: Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking, 2020, pp. 31–36.

[14] F. Zhang, K. Kuang, Z. You, T. Shen, J. Xiao, Y. Zhang, C. Wu, Y. Zhuang, X. Li, Federated unsupervised representation learning, arXiv preprint arXiv:2010.08982 (2020).

[15] E. Tzinis, J. Casebeer, Z. Wang, P. Smaragdis, Separate but together: Unsupervised federated learning for speech enhancement from non-iid data, arXiv preprint arXiv:2105.04727 (2021).

[16] T. Luo, S. G. Nagarajany, Distributed anomaly detection using autoencoder neural networks in WSN for IoT, Technical Report, 2018. doi:10.1109/ICC.2018.8422402. arXiv:1812.04872.

[17] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, G. Srivastava, Federated-learning-based anomaly detection for iot security attacks, IEEE Internet of Things Journal 9 (2021) 2545–2554.

[18] E. S. Lubana, C. I. Tang, F. Kawsar, R. P. Dick, A. Mathur, Orchestra: Unsupervised federated learning via globally consistent clustering, arXiv preprint arXiv:2205.11506 (2022).

[19] S. Han, S. Park, F. Wu, S. Kim, C. Wu, X. Xie, M. Cha, Fedx: Unsupervised federated learning with cross knowledge distillation, in: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXX, Springer, 2022, pp. 691–707.

[20] C. C. Aggarwal, Outlier Analysis, Springer International Publishing, 2017. doi:10.1007/978-3-319-47578-3.

[21] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv:1708.07747 (2017).

[22] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, et al., Support vector method for novelty detection., in: NIPS, volume 12, Citeseer, 1999, pp. 582–588.

A. Model Architecture

1. C2D(16,3,2,R) → C2D(8,3,2,R) → C2DT(8,3,2,R) → C2DT(16,3,2,R) → C2D(1,3,S)
2. C2D(32,3,R) → MP2D(2) → C2D(32,3,R) → MP2D(2) → C2DT(32,3,2,R) → C2DT(32,3,2,R) → C2D(1,3,S)

C2D denotes Conv2D, C2DT denotes Conv2DTranspose, MP2D denotes MaxPooling2D, R denotes ReLU activation, and S denotes Sigmoid activation. The numbers indicate filter counts and kernel sizes. The number after a comma in a Conv2D or Conv2DTranspose layer represents the stride.