Understanding What Federated Learning Models Learn: A Comparative Study with Traditional Models

Rafael Teixeira*†, Leonardo Almeida*, Pedro Rodrigues*, Julio Corona*†, Mário Antunes*† and Rui L. Aguiar*†
*Instituto de Telecomunicações, Universidade de Aveiro, Aveiro, Portugal
E-mails: {rafalgteixeira,leonardoalmeida,pedrofrodrigues,jcamejo,mario.antunes,ruilaa}@av.it.pt
†DETI, Universidade de Aveiro, Aveiro, Portugal
E-mails: {rafaelgteixeira,leonardoalmeida,pedrofrodrigues,jcamejo91,mario.antunes,ruilaa}@ua.pt

***Abstract*—Federated Learning (FL) offers a robust framework for training Machine Learning (ML) models across distributed devices while preserving data privacy. However, concerns remain about whether FL-trained models learn in the same way as their centralized-data counterparts. This paper examines the impact of FL on model behavior, utilizing Explainable AI (XAI) techniques and correlation metrics to compare feature importance. By applying two XAI metrics across four public datasets and evaluating multiple FL strategies, we assess how closely federated models align with traditionally trained ones. Although models often achieve similar classification performance (Matthews's Correlation Coefficient (MCC) $>$ 0.9), we observe significant differences in attribution patterns, especially in async approaches, with correlation scores below 0.7 Spearman's Rank Correlation Coefficient (SRCC) on several occasions and below 0.4 SRCC for the worst scenario. These findings suggest that identical performance does not always imply equivalent learning. Furthermore, since these findings were obtained under Independent and Identically Distributed (IID) settings, it is expected that under non-IID settings, the results might be even worse, underscoring the need for explainability-driven validation tools to ensure the reliability, fairness, and trustworthiness of FL models in practice.**

***Index Terms*—XAI, Federated Learning, 6G**

I. Introduction

Federated Learning (FL) is emerging as a pivotal approach to enable privacy-preserving Machine Learning (ML) in distributed environments. Unlike traditional centralized-data training methods, FL enables models to be trained across multiple devices or nodes without transferring raw data [1], thereby addressing growing concerns regarding data ownership, security, and regulatory compliance, such as the General Data Protection Regulation (GDPR).

However, while FL offers clear advantages in terms of privacy and scalability, it also introduces significant challenges. Heterogeneous data distributions across clients can impair model convergence and performance. Additionally, the decentralized nature of FL complicates the evaluation of model fidelity compared to traditionally trained counterparts.

This work proposes a methodology to assess the similarity between FL and centrally trained models using Permutation Importance (PI) and Partial Dependence Variance (PDV), two Explainable Artificial Intelligence (XAI) techniques. This study analyzes if FL models assign comparable significance to input features, focusing on the effects of federation on training rather than proposing new methods.

We validate our approach using 6G network applications, where FL is a cornerstone technology [2]. Experiments focus on network slicing and intrusion detection, representing real-world FL deployment scenarios with privacy and infrastructure constraints.

The remaining sections of the paper are organized as follows: Section II introduces the reader to explainability assessment and FL. Section III provides a detailed description of the experimental configuration used in this study, and Section IV presents the results obtained. The results are discussed in Section V, and the conclusions and future work are drawn in Section VI.

II. Background

This section provides the necessary background for XAI in subsection II-A and ML model training in subsection II-B.

A. Explainable AI

XAI algorithms address transparency and interpretability in black-box ML models [3]. They are classified based on timing (*ante-hoc* for pre/during training explanations and *post-hoc* for after training explanations), model specificity (model-specific or model-agnostic), and explanation scope (local for individual predictions, global for the entire model) [4].

This work utilizes model-agnostic XAI approaches providing global model explanations and generalizable results. Metrics like Local Interpretable Model-agnostic Explanations (LIME), SHapley Additive exPlanations (SHAP), GradCAM, or Integrated Gradients were excluded due to their focus on individual explanations or image-based classifications. Thus, PI and PDV were selected.

PI [5] is a model-agnostic technique measuring feature importance by assessing the increase in prediction error after permuting a feature's values, indicating its significance.

Similar to PI, PDV [6] is a model-agnostic technique that computes global feature importance or feature interaction of

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

a pair of features. It relies on Partial Dependence to analyze the marginal effect of features on a ML model's predicted outcome, summarized by a single positive number.

These metrics were chosen due to their use in 5G/6G and FL [7], [8]. While state-of-the-art, both PI and PDV make assumptions for feature extraction; and since there is no way to evaluate which is correct when applied to Deep Neural Networks (DNNs), in this study we will consider the results from both techniques.

B. Training methods

In this section, we provide an overview of model training from the perspective of leveraging distributed resources to train models and reducing training costs. Consequently, the overview will not cover different ML models or optimization algorithms used for parameter optimization during training.

Traditional learning is the simplest form of training. It trains a model using a single computing unit and the complete training set. Since it only uses a single computing unit, it is limited to vertical scaling, which can become very expensive. Despite that, this approach has no communication overhead, as there is no network communication between computing units. Given that there is no overhead, this is the best approach if the model can be trained on time.

Distributed learning builds upon traditional learning to enable it to be horizontally scaled. This is achieved by leveraging multiple computing units within the same computer or distributing them across various computers connected via high-speed links [9].

In distributed learning, a central server, known as a parameter server, manages training and stores the dataset, distributing subsets to computing units. Training algorithms differ in communication methods, aiming to minimize data exchange while maintaining learning efficiency. Approaches fall into two categories: model parallelism and data parallelism.

Model Parallelism splits the model either by layers (vertically) or within layers (horizontally), useful when a single computing unit cannot handle complete training. However, it requires high communication, as units exchange intermediate values twice per example.

Data Parallelism splits training data across multiple computing units, increasing sample processing throughput and reducing batch time. The two main approaches—centralized and decentralized optimization—differ in how model parameters are updated.

In centralized optimization, the parameter server updates the model by collecting gradients from all computing units for each batch processed. In contrast, in decentralized learning, the computing units can execute multiple training epochs. Decentralized optimization has gained popularity due to its reduced communication requirements. Regardless of the approach, the training can be performed synchronously or asynchronously.

All previous approaches assume a centralized dataset, but this isn't always feasible, especially with sensitive or personal data that may require anonymization.

Federated Learning addresses this by enabling model training without centralizing data, reducing the risk of data leaks.

That said, FL can still be seen as a special case of data parallelism, meaning that any approach under that category can be applied in FL. The most common approach in FL is the decentralized synchronous approach, with the various implementations changing how the workers' weights are combined according to their data distributions.

III. Research Metodology

Although FL and XAI have been used together in 5G/B5G [1], [7], [8], to the best of the authors' knowledge, the impact that FL can have on the final model has yet to be analyzed. In this section, we describe the experiments conducted to determine whether models trained with FL and traditional learning identify similar patterns in the data. The code used to obtain the results is available on GitHub [1].

The experiments were proposed in a scenario that replicates an FL environment using a network of 59 nodes—58 worker nodes for training and one central parameter server for aggregation. Each node ran as a Virtual Machine on Ubuntu 24.04, hosted across three servers: a GIGABYTE R120-T32-00 (48 cores, 64 GB RAM), a HUAWEI Kunpeng 920 7260 system (128 cores, 384 GB RAM), and an Intel Xeon E5-2620 v4 system (16 cores, 256 GB RAM). Figure 1 details the worker capabilities.

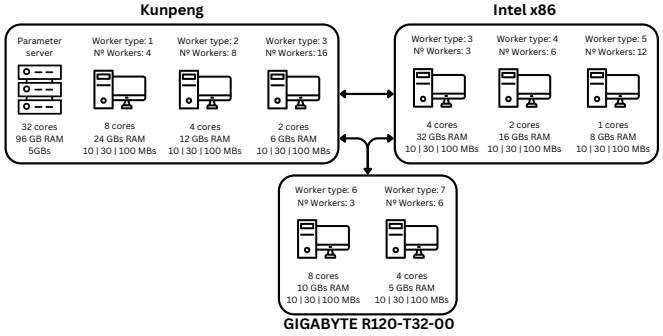

Fig. 1. Description of the virtual machines used in the experimentation environment from the three different servers. The description of the machines mentions the number of CPU cores, the amount of RAM, and the available bandwidth.

The diverse servers and workers simulate a heterogeneous environment with varying computing capabilities. The three servers, connected via a 20 Gbit cabled link for ample bandwidth, evenly host the 58 worker nodes.

A. Datasets and models

Four datasets were selected to guarantee the validity of the results and their generalization: two based on network slice attribution and two for intrusion detection systems. These datasets represent two tasks that will benefit from FL due to their data privacy constraints and distributed nature.

[1] github.com/rgtzths/xai_fl

1) Network Slice Attribution: The datasets considered for network slicing were the Slicing5G [10] and NetSlice5G.

The objective of the first dataset [10] is to classify a request according to one of three slices (Ultra Reliable Low Latency Communications (URLLC), enhanced Mobile Broadband (eMBB), or massive Machine Type Communications (mMTC)) given a set of resource requirements and device characteristics.

The raw dataset is composed of eight input features and one output. Seven of the input features are categorical, so they were transformed with one hot encoding as a preprocessing step. The only exception is the *time* feature, which remained unchanged. After preprocessing, the dataset consisted of 60 features and 466,739 examples. The examples were then shuffled and partitioned into two subsets: 80% for training and 20% for testing.

Furthermore, the training set was divided into two subsets: 80% for the actual training and 20% for validation. This results in a ratio of 64/16/20 for training, validation, and testing. The final dataset consists of 298,712 examples for training, 74,679 for validation, and 93,348 for testing.

The model implemented was based on the original model proposed for the dataset and is a simple neural network with four layers.The first layer contains 16 nodes and uses the Rectified Linear Unit (ReLU) activation function. After that, there is another layer with eight nodes with an ReLU activation function and a layer with four nodes and the tanh activation function. Finally, the output layer consists of three nodes and employs the softmax activation function.

The second dataset, NetSlice5G, is publicly available on Kaggle[2] and comes pre-divided into training and testing sets. However, since the testing set lacked the expected labels for the inputs, it was discarded, as accuracy metrics could not be derived from it. This dataset shares features similar to the previous one, but differs in that the categorical features have already been preprocessed using one-hot encoding. The only additional step was to divide the data using the same ratio as before, resulting in 20,212 training examples, 5,054 for validation, and 6,317 for testing.

The model used for this dataset was proposed in [11]. The model consists of an input layer with 16 nodes, a second layer that is dense with 73 nodes and features a ReLU activation function, and a third layer with a dropout rate of 0.5. Finally, the output layer is dense with three nodes and a softmax activation.

2) Intrusion Detection System: For Intrusion Detection System (IDS), two datasets were considered: the IoT-Device-Network-Logs (IoT-DNL) and the UNSW. The IoT-DNL is a dataset tailored for network-based IDS initially introduced by Sahil Dixit [12] and sourced from Kaggle[3]. This open-source dataset is designed explicitly for evaluating anomaly-based IDS in wireless environments. The dataset is flow-based and labeled, focusing on Internet of Things (IoT) devices. It has undergone preprocessing to suit the requirements of network-based IDS.

The dataset comprises a total of 477,426 instances, each with 14 variables. Upon analyzing the correlation between the features and the normality feature (the label), it was possible to conclude that frame.number and frame.time are highly correlated with each other but do not have any correlation with the label. Therefore, it was decided to remove these two features from the dataset and retain the remaining ones.

The UNSW-NB15 dataset is a synthetically generated dataset created by the Australian Centre for Cyber Security (ACCS). This dataset combines real, modern, regular network traffic with contemporary, synthesized attack activities. Comprising over 2 million records with 49 features, the UNSW-NB15 dataset is designed to train binary and multi-class classification models, as each malicious activity is labeled with one of the nine attack categories. To avoid complex preprocessing procedures, in this study, we used the NF-UNSW-NB15[4] preprocessed dataset published in [13] by the University of Queensland.

Since the datasets were already preprocessed, feature scaling and dataset division were the only steps applied. The dataset division followed the same ratio as the network slicing datasets, resulting in 1,912,220/381,940 training examples, 382,444/76,388 validation examples, and 478,055/95,486 testing examples for the UNSW and IoT-DNL, respectively.

Regarding the models used, the model was the one used in [11]. The model consists of four hidden layers, each with 64 neurons, and a dropout layer with a dropout rate of 0.1 between each hidden layer. The activation function used in the hidden layers is the ReLU function, and the softmax is used in the output layer.

As for the UNSW-NB15 dataset, in [14], the authors use 3 different DNNs, including Multi-layer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN). All models performed well and achieved similar results, so we used the MLP model, the simplest and fastest model to train. The model consists of 3 hidden layers with 128, 96, and 64 neurons, respectively, and a dropout layer with a dropout rate of 0.25 after the hidden layers. The activation function used in the hidden layers is the ReLU function, and the softmax is used in the output layer.

B. Federated learning algorithms

This study assesses the impact of FL on model explainability by examining four FL algorithms. Two use decentralized optimization: one with synchronous communication [15] and another with asynchronous [16]. The other two follow centralized optimization, also with synchronous [17] and asynchronous [9] communication. These approaches differ in how model updates are shared and aggregated.

The hyperparameters used on the FL algorithms are worth mentioning. The models trained with FL were trained using 58 workers with Independent and Identically Distributed (IID)

[2]kaggle.com/datasets/amohankumar/network-slicing-in-5g

[3]kaggle.com/datasets/speedwall10/iot-device-network-logs

[4]espace.library.uq.edu.au/view/UQ:ffbb0c1

data distribution, and for single-host training, we utilized the full capacity of the parameter server. Every approach used a learning rate of 0.001 and a batch size of 128. The number of local epochs for the decentralized approaches was one, and the updated bound for the decentralized asynchronous approach was 0.2. All traditionally trained models were trained until convergence, and the FL ones were trained until they achieved the same validation Matthews's Correlation Coefficient (MCC).

C. Model comparison

In this subsection, we will focus on how the similarity between XAI results of FL approaches was analyzed. Since the focus of the work is not on the results obtained by a model but on the concordance of the XAI results between models trained using traditional and federated approaches, after computing the XAI metrics, we used the Pearson Correlation Coefficient (PCC) [18] and Spearman's Rank Correlation Coefficient (SRCC) [19] to compare each FL-trained model's feature importance with the traditionally trained ones.

PCC measures the linear correlation between two variables, X and Y. It takes values between +1 and -1, where +1 indicates total positive linear correlation, 0 denotes no linear correlation, and -1 signifies total negative linear correlation. The PCC equation is presented in Equation 1.

$$PCC(x,y) = \frac{\sum(x_i - \bar{x}) \times (y_i - \bar{y})}{\sigma_x \times \sigma_y} \tag{1}$$

SRCC is a non-parametric measure that assesses the monotonic relationship between two variables, X and Y. Like PCC, SRCC also ranges from +1 to -1, where +1 represents total positive monotonicity, 0 denotes no monotonicity, and -1 signifies total negative monotonicity. Unlike PCC, SRCC considers the rankings or order of data points rather than their actual numerical values, making it more robust to nonlinear relationships. The SRCC equation is presented in Equation 2.

$$\text{SRCC}(X,Y) = \frac{\text{cov}(\text{R}(X), \text{R}(Y))}{\sigma(\text{R}(X))\sigma(\text{R}(Y))} \tag{2}$$

Since PCC and SRCC measure correlation between variables, a high correlation between the output of the XAI technique for a FL approach and the single training will mean that the underlying model is similar and a low correlation will mean that the model learned different patterns and is considerably different.

The values obtained from the XAI techniques were calculated using 10% of the test dataset. To ensure that this data follows the same distribution as the test dataset, we employed a stratified sampling approach.

IV. Results

This section provides an analysis of the final model's performance for each learning approach in each dataset, as well as a correlation analysis of the feature importance of the trained models.

Since all the considered datasets are classification datasets, they all use the same performance metrics. We considered accuracy, F1-Score, and MCC as relevant metrics. The accuracy and F1-score were chosen because they were also used by other models analyzing the datasets, and MCC was used because it tends to penalize misclassifications more severely than F1-Score. The metrics results for the validation and test sets of the six datasets are presented in Table I.

TABLE I
PERFORMANCE RESULTS OF THE VARIOUS MODELS FOR THE SIX DATASETS.

Model	Validation dataset		Test dataset	
	F1 score	MCC	F1 score	MCC
Slicing5G				
Traditional Training	1.000	1.000	1.000	1.000
Centralized Sync	1.000	1.000	1.000	1.000
Centralized Async	1.000	1.000	1.000	1.000
Decentralized Sync	1.000	1.000	1.000	1.000
Decentralized Async	1.000	1.000	1.000	1.000
NetSlice5G				
Traditional Training	1.000	1.000	1.000	1.000
Centralized Sync	1.000	1.000	1.000	1.000
Centralized Async	1.000	1.000	1.000	1.000
Decentralized Sync	1.000	1.000	1.000	1.000
Decentralized Async	1.000	1.000	1.000	1.000
IOT-DNL				
Traditional Training	0.997	0.997	0.998	0.997
Centralized Sync	0.999	0.999	0.999	0.999
Centralized Async	0.953	0.947	0.955	0.949
Decentralized Sync	0.997	0.996	0.997	0.997
Decentralized Async	0.997	0.996	0.997	0.996
UNSW				
Traditional Training	0.987	0.846	0.987	0.848
Centralized Sync	0.989	0.850	0.988	0.85
Centralized Async	0.983	0.796	0.982	0.798
Decentralized Sync	0.986	0.837	0.986	0.839
Decentralized Async	0.984	0.805	0.983	0.808

Analyzing the table, it is clear that every training approach could achieve similar performance, as the differences in the various metrics are usually lower than 0.01. The only exceptions were the centralized async approach, which performed worst in the IoT-DNL and UNSW datasets, and the decentralized async approach for the UNSW dataset.

As explained in Section II, the PDV and PI methods give the user vital insights regarding the importance each feature has on the model's output.

One way to analyze the similarity between the models is to compare their outputs. As mentioned in Section III, this analysis was developed using two correlation metrics, the PCC and SRCC. Table II presents the correlation results obtained for every dataset.

Looking at the results, it is safe to say that the models obtained by the different FL approaches are very similar to those obtained using the traditional approach, as they have a high correlation (greater than 0.85). The only exceptions were the Centralized Async approach, which achieves notably lower correlations in various configurations, and the Decentralized Async approach, which achieves lower correlations for SRCC

TABLE II
CORRELATION RESULTS OF THE VARIOUS MODELS WITH THE TRADITIONALLY TRAINED ONE FOR THE SIX DATASETS. (BOLD VALUE PRESENTS CORRELATIONS BELOW 0.7)

Tabular Datasets

Model	PI		PDV	
	PCC	SRCC	PCC	SRCC
Slicing5G				
Centralized Sync	**0.699**	**0.593**	0.771	**0.622**
Centralized Async	0.881	0.767	0.906	0.768
Decentralized Sync	0.928	0.779	0.881	0.753
Decentralized Async	0.935	0.571	0.842	**0.530**
NetSlice5G				
Centralized Sync	0.997	**0.505**	0.998	**0.356**
Centralized Async	0.998	0.909	0.999	0.926
Decentralized Sync	1.000	0.894	1.000	0.903
Decentralized Async	0.995	0.896	0.997	0.875
IOT-DNL				
Centralized Sync	0.941	0.955	0.898	0.936
Centralized Async	0.77	0.709	**0.649**	**0.591**
Decentralized Sync	0.978	0.973	0.979	0.964
Decentralized Async	0.967	0.955	0.944	0.945
UNSW				
Centralized Sync	0.989	0.926	0.989	0.821
Centralized Async	0.990	0.823	0.990	**0.624**
Decentralized Sync	0.987	0.910	0.987	0.881
Decentralized Async	0.975	0.905	0.975	0.799

in the UNSW dataset, as well as in several configurations of the Slicing5G dataset.

V. DISCUSSION

Considering the results presented in the previous section, several aspects merit an extended discussion, such as the tendency of Async approaches to have lower correlation values. This section presents an in-depth analysis of these aspects.

To better understand the results presented in Table II, it is necessary to understand the complexity of the datasets considered in the experiments. The Slicing5G and NetSlice5G are the easiest to solve, as the models usually solve it within a few epochs, and the models' architectures are very simple. After that comes the IoT-DNL, which requires more epochs, does not achieve an MCC of 1, and has a considerably larger network. The hardest dataset to solve is the UNSW, as the achieved MCC is lower even when more epochs are available and a considerably more expressive neural network is trained.

Considering the difficulty of learning from the datasets, several reasons can lead to the results presented. The results from the Slicing5G/NetSlice5G datasets can result from multiple independent feature combinations being able to solve the dataset. For example, for the Slicing5G, the "Packet Delay Budget (Latency) $<$ 300ms" and "Packet Delay Budget (Latency) $<$ 10ms" features have 0.68 and -0.68 correlation values with the label, and for the NetSlice 5G the "IoT" and the "LTE/5G" present 0.91 and -0.91. This means that the FL approaches can arrive at different valid solutions, as one approach can value the positive correlation and the other the negative, and still solve the classification task correctly.

This is usually not the case in the more complex datasets, and the models eventually converge in the same direction. For example, in the UNSW dataset, only "MIN_TTL" correlates close to 0.7, followed by the "MAX_TTL" at 0.55. Since there are no equivalent positive and negative correlations in these datasets, the optimization path becomes clearer.

However, when considering the asynchronous versions of the FL approaches, given that they require more epochs to converge, the faster workers create models that are biased towards their partial datasets. This has been shown in previous works considering heterogeneous workers [20], where asynchronous approaches show considerably different behavior from synchronous approaches. The reason for this behaviour is perfectly depicted in the example in Figure 2, where although both worker 1 and 2 tend to increase the value w_1, worker 1 tends to reduce the value of w_2. Nonetheless, since worker 1 can produce more updates, the global mode becomes biased towards the data distribution of worker 1.

Furthermore, since the Asynchronous versions update the model without waiting for every worker, in some cases, the updates submitted by the workers were calculated for previous models. This is also depicted in Figure 2, where the green circles represent the intended model update, which can be significantly different from the actual update. When these two issues are compounded, the model update can lead to an undesired result, as is the case with the second update of worker two, where the intended update aimed to increase slightly w_1 while maintaining w_2, but was applied on a model that required an increase in w_2 and a slight decrease in w_1 to achieve the desired model.

Consequently,

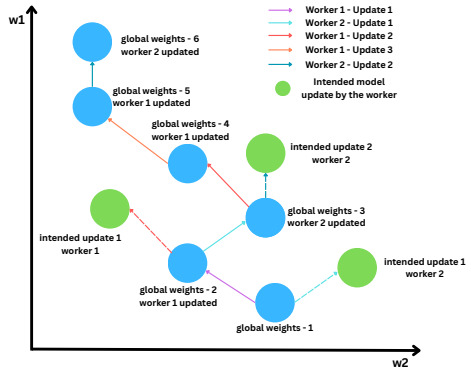

Fig. 2. Example of a centralized asynchronous update scheme where worker 1 produces more updates than worker 2.

These issues related to asynchronous communications can justify why, even with the same performance, the models differ more than their synchronous counterparts. This is more evident in the centralized asynchronous version, as global model updates occur more frequently, leading to the second issue being more prevalent.

Another crucial aspect to discuss regarding these results is that they were obtained using a random and uniform division of the dataset among workers. This ensures a best-case

scenario for the FL approaches, as the workers will have access to samples from every label. In reality, most FL scenarios will face workers with considerably imbalanced datasets that might not have examples from various labels. This is usually referred to as a non-Independent and Identically Distributed (non-IID) setting.

Although it is known that models usually struggle to converge in a non-IID setting [21], there is also no analysis regarding the similarity between the model learned in that setting and the traditional one. Furthermore, the evidence shows that besides impacting performance, non-IID data also affects model fairness [21]. Considering the results and the evidence in the state-of-the-art, the differences between models in non-IID settings are expected to be even higher.

That said, XAI methods must be developed for FL to ensure model integrity, security, and fairness, as this is the only approach to create trust in the FL training process.

VI. Conclusion

The main takeaway from this work is that even when models reach high performance, the patterns learned in FL can differ considerably from those obtained through traditional training. The four FL strategies explored here represent the main training approaches in the field, and the datasets used provide a meaningful basis for explainability analysis.

Our results indicate that both the complexity of the dataset and the communication strategy between models can cause models to drift, leading them to learn new patterns while still maintaining similar performance metrics. At the same time, some FL approaches may introduce bias toward workers who participate more frequently, pointing to the need for mechanisms that ensure fairer training.

The lack of alignment across models, even in cases of strong performance, highlights the importance of discussing FL's applications in sensitive areas such as healthcare. In such settings, guaranteeing model reliability is crucial, especially since FL-trained models may differ substantially from those trained using traditional methods.

Looking ahead, future work will evaluate FL under non-IID data conditions, where divergences from traditional learning are expected to be even more pronounced. Expanding the experiments to include a larger and more diverse set of workers will also provide further insights into how FL models generalize in practice.

Acknowledgment

This study was partially funded by the PRR – Plano de Recuperação e Resiliência and by the NextGenerationEU funds at University of Aveiro, through the scope of the Agenda for Business Innovation "NEXUS: Pacto de Inovação – Transição Verde e Digital para Transportes, Logística e Mobilidade" (Project nº 53 with the application C645112083-00000059); and by the European Union/Next Generation EU, through Programa de Recuperação e Resiliência (PRR) [Project Nr. 29: Route 25 (02/C05-i01.01/2022.PC645463824-00000063)];

References

[1] R. Teixeira, G. Baldoni, M. Antunes, D. Gomes, and R. L. Aguiar, "Leveraging decentralized communication for privacy-preserving federated learning in 6g networks," *Computer Communications*, vol. 233, p. 108072, 2025.

[2] G. Flagship, "6g white paper on edge inteligence," 2021. https://5g-ppp.eu/wp-content/uploads/2021/05/AI-MLforNetworks-v1-0.pdf Accessed: 2023-09-15.

[3] Y. Wu, G. Lin, and J. Ge, "Knowledge-powered explainable artificial intelligence for network automation toward 6g," *IEEE Network*, vol. 36, no. 3, pp. 16–23, 2022.

[4] I. Kök, F. Y. Okay, O. Muyanlı, and S. Özdemir, "Explainable artificial intelligence (xai) for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14764–14779, 2023.

[5] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 10 2001.

[6] B. M. Greenwell, B. C. Boehmke, and A. J. McCarthy, "A simple and effective model-based variable importance measure," 2018.

[7] A. Renda, P. Ducange, F. Marcelloni, D. Sabella, M. C. Filippou, G. Nardini, G. Stea, A. Virdis, D. Micheli, D. Rapone, *et al.*, "Federated learning of explainable ai models in 6g systems: Towards secure and automated vehicle networking," *Information*, vol. 13, no. 8, p. 395, 2022.

[8] J. L. Corcuera Bárcena, P. Ducange, F. Marcelloni, G. Nardini, A. Noferi, A. Renda, F. Ruffini, A. Schiavo, G. Stea, and A. Virdis, "Enabling federated learning of explainable ai models within beyond-5g/6g networks," *Computer Communications*, vol. 210, pp. 356–375, 2023.

[9] M. Langer, Z. He, W. Rahayu, and Y. Xue, "Distributed training of deep learning models: A taxonomic perspective," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 12, pp. 2802–2818, 2020.

[10] A. Thantharate, R. Paropkari, V. Walunj, and C. Beard, "Deepslice: A deep learning approach towards an efficient and reliable network slicing in 5g networks," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, (New York, NY, USA), pp. 0762–0767, IEEE, 2019.

[11] R. Teixeira, G. Baldoni, M. Antunes, D. Gomes, and R. Aguiar, "Leveraging decentralized communication for privacy-preserving federated learning in 6g networks," *Computer Communications*, vol. 233, p. 108072, 01 2025.

[12] D. K. K. Reddy, J. Nayak, B. Naik, and G. S. Pratyusha, "Deep neural network–based security model for iot device network," in *Deep Learning for Internet of Things Infrastructure*, pp. 223–243, CRC Press, 2021.

[13] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Networks and Applications*, vol. 27, p. 357–370, Nov. 2021.

[14] R. A. Khamis and A. Matrawy, "Evaluation of adversarial training on different types of neural networks in deep learning-based idss," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, (Montreal, QC, Canada), pp. 1–6, IEEE, 2020.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, PMLR, 04 2017.

[16] S. Zhang, A. Choromanska, and Y. LeCun, "Deep learning with elastic averaging sgd," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, (Cambridge, MA, USA), p. 685–693, MIT Press, 2015.

[17] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," 2017.

[18] J. Benesty, J. Chen, Y. Huang, and I. Cohen, *Pearson Correlation Coefficient*, pp. 1–4. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[19] P. Sedgwick, "Spearman's rank correlation coefficient," *BMJ*, vol. 349, 2014.

[20] R. Teixeira, L. Almeida, M. Antunes, D. Gomes, and R. L. Aguiar, "Efficient training: Federated learning cost analysis," *Big Data Research*, vol. 40, p. 100510, 2025.

[21] S. Amiri, A. Belloum, E. Nalisnick, S. Klous, and L. Gommans, "On the impact of non-iid data on the performance and fairness of differentially private federated learning," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 52–58, 2022.