International Revenue Share Fraud Prediction on the 5G Edge Using Federated Learning

Luís Ferreira[1,2], Leopoldo Silva[1], Francisco Morais[1], Carlos Manuel Martins[3], Pedro Miguel Pires[3], Helena Rodrigues[2], Paulo Cortez[2*] and André Pilastri[1]

[1]EPMQ - IT Engineering Maturity and Quality Lab, CCG ZGDV Institute, Guimarães, Portugsn-article-revised2.texal.
[2]ALGORITMI Center, Department of Information Systems, University of Minho, Guimarães, Portugal.
[3]WeDo Technologies, Braga, Portugal.

*Corresponding author(s). E-mail(s): pcortez@dsi.uminho.pt;
Contributing authors: luis.ferreira@dsi.uminho.pt; leopoldo.silva@ccg.pt; francisco.morais@ccg.pt; carlos.mmartins@mobileum.com; pedro.mpires@mobileum.com; helena@dsi.uminho.pt; andre.pilastri@ccg.pt;

Abstract

Edge Computing and Multi-access Edge Computing (MEC) are two recent paradigms of distributed computing that are growing due to the rise of the fifth-generation (5G) of broadband cellular networks. The development of Edge Computing and MEC architectures involves the hosting of applications close to the end-users, allowing: an improved privacy, given that critical data is not shared with other systems; a reduced communication latency; an improved application speed; and a more efficient energy use. However, many applications are challenged by Edge Computing and MEC. In the case of Machine Learning (ML) applications, there can be privacy rules that do not allow data to be shared among distinct edges. Additionally, the devices used to train ML models might present lower computational capabilities than traditional computers. In this work, we present a Federated ML architecture that uses decentralized data and light ML training techniques to fit ML models on the 5G Edge. Our system consists of edge nodes that train models using local data and a centralized node that aggregates the results.

As a case study, an International Revenue Share Fraud (IRSF) task is addressed by considering two real-world datasets obtained from a commercial provider of Telecom analytics solutions. We test our architecture using two iterations of a Federated ML method, then compare it with a centralized ML model that is currently adopted by the provider. The results show that the Federated Learning decentralized approach produces an excellent level of class discrimination and that the main models maintain the performance across two rounds of decentralized training and even surpass the existing centralized model. After validating the results with the Telecom provider, we have built a prototype technological architecture that can be deployed in a real-world MEC scenario.

Keywords: 5G Networks, Edge Computing, Federated Learning, Machine Learning, Multi-access Edge Computing

1 Introduction

The fifth-generation (5G) of broadband cellular networks not only represents an evolution from the previous generation (4G) but is also a new technological standard that aims to achieve improvements in the quality of service by offering, for instance, higher throughput and low latency. These improvements present a challenge and, in response, a new computing paradigm has emerged: Edge Computing. Edge Computing technology moves computational resources to the devices at the edge of a communication network. Computations that were typically made in the cloud or at the end-user devices are now made on edge devices. This move allows the desired high throughput and low latency but also diminishes the data transfer between end-user devices and the cloud, allowing the data to be kept on the edge, near the end-user devices [1].

This way, Edge Computing technology offers the possibility of hosting applications close to the users, thus reducing communication latency, and improving application performance and energy efficiency [2]. In particular, Multi-access Edge Computing (MEC) provides Cloud Computing and Information Technology capabilities within the Radio Access Network (RAN), at the edge of mobile communication networks, close to mobile subscribers [3]. MEC technology offers a distributed environment for the provision of applications and services, as well as for content processing and storage, in the proximity of mobile users.

This paper proposes a framework to detect International Revenue Share Fraud (IRSF) on the Edge. In this framework, we instantiate a ML application on the edges and on a central module, using a common ML model to be used on all the edges and Federated Learning to perform decentralized training on the edges. IRSF is one of the most prevalent frauds among the mobile phone network. Due to how international agreements among telecommunication companies are made and also for privacy concerns, this fraud is difficult to avoid or backtrack, making it a popular choice for malicious actors [4, 5].

This framework also aims to cope with the novel data protection and privacy regulations [6], which introduced strict rules regarding the possible usages of costumer data and limitations on how that data may be transferred among locations and business partners.

Our work is part of the R&D project "Opti-Edge: 5G Digital Services Optimization at the Edge", which is being developed by a leading provider of analytics solutions for the Telecom industry. The goal of the Opti-Edge project is to develop a solution capable of running ML algorithms to detect IRSF in a MEC scenario. In this scenario, typically there are a group of base stations where phone calls occur and that generate data records. For privacy reasons, each base station must be capable of detecting fraudulent calls without sharing data with other base stations, while consuming the least possible amount of computational resources. The proposed Federated ML framework can deal with the training and deployment of ML models within a MEC scenario using decentralized data and lighter ML algorithms.

For the experimental results, we used real-world IRSF data from one of the software company's clients to evaluate our architecture. We simulate the application of the proposed architecture during two rounds of decentralized training using three edge nodes and one centralized location. Then, we analyze the results and compare them with a pure centralized ML model that was previously developed by the software company. Moreover, we monitor the performance of the ML models when tested on more recent data. After validating the results with the Telecom company, we designed a prototype technological architecture that can be deployed in a real-world MEC scenario.

This work consists of a rather extended version of our previous work [7] and that includes several new elements. Firstly, we describe the IRSF task in more detail and perform an updated survey of the state-of-the-art works regarding application of Federated Learning in Edge Computing architectures (Section 2). Secondly, the IRSF data is presented with more depth (Section 4.1). Thirdly, we analyze the proposed federated ML architecture using an additional and larger IRSF dataset. Fourthly, the new Section 6 discusses the technologies that were used to implement a demonstrator prototype of the proposed Federated Learning solution.

The paper is organized as follows. In Section 2, we present the related work. Next, Section 3 describes the proposed ML architecture for Federated Learning at the Edge. Section 4 describes the used datasets, ML frameworks, and the algorithm used to aggregate the ML models. Section 5 shows and discusses the experimental results. Section 6 details the technological architecture of the prototype demonstration that was built using the proposed ML architecture. Finally, Section 7 presents the main conclusions and future work directions.

2 Related Work

2.1 5G and Edge Computing

5G offers the potential to create new applications, business models, and to improve quality of life through almost instantaneous data communication and high transmission rates, low communication latency, and massive connectivity for new applications, including autonomous vehicles, smart cities, smart homes, or Industry 4.0 [8]. To reach 5G goals, the Edge Computing paradigm has emerged and, with it, the extension and transference of cloud capabilities to the edge of the network are enabled. This allows computationally-intensive tasks and data storage to happen near the end-user equipment and within the RAN, increasing the quality of service requirements, including low latency and high throughput, needed by the applications running on that equipment [1, 9].

Several technologies have emerged to support Edge Computing and to define a new Cloud Computing paradigm that breaks through the centralized architecture and alleviates the constraints that are faced by the centralized cloud paradigm [2]. Of these emerging technologies, the MEC is of particular interest to this work due to its use and orientation to the telecommunications sector. MEC has several benefits for Mobile Network Operator (MNO), Application Service Provider (ASP), and other participants in this sector. MNOs can allow access to the RAN to third-party vendors for the deployment of their applications and services, ASPs may profit from a MEC-enabled infrastructure that offers low latency and high bandwidth along with ease of scalability, and also end-users can experience performance improvements through offloading techniques [9]. The particular MEC technology was standardized by the European Telecommunications Standards Institute (ETSI) and the Industry Specification Group (ISG) and it is recognized by the European Public-Private Partnership on 5G Infrastructure as a key emerging technology for 5G networks [3].

2.2 International Revenue Share Fraud

Telephony networks and, in particular, mobile networks are subjected to many types of fraud, including toll evasion frauds, retail billing frauds and revenue share fraud [10]. The last one is specifically known as IRSF and it consists of a relevant fraud type that is addressed by this work.

According to the Communications Fraud Control Association 2021 fraud loss survey [11], IRSF is still the most reported fraud scheme with losses totaling $6.69B. IRSF takes many forms and evolves rapidly, making it one of the favorite fraudster schemes [4]. It usually requires coordination and cooperation among several types of fraudsters that can range from individual fraudsters to providers of International Revenue Premium Numbers. Among its forms, the wangiri[1] scam is the most used [11] and it is depicted in Fig. 1. Under this scheme, the fraudster calls the victim, lets the call ring once and then drops

[1] Wangiri is a Japanese word meaning 'one ring and drop'

the call. When and if the victim calls back, the call is redirected by a fraudster operator to the premium number and the victim incurs the high costs of the premium call. The call profit is then shared among fraudsters [12].

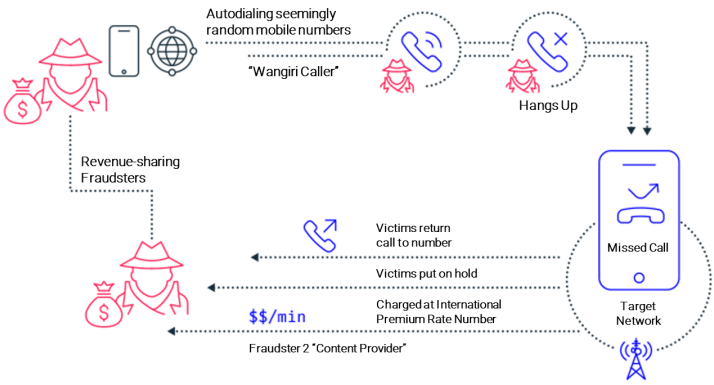

Fig. 1 The wangiri call scam (adapted from [12]).

Within our knowledge, there is scarce research related with the detection of IRSF using ML algorithms. In [13], IRSF was approached as an anomaly detection task by using an Isolation Forest algorithm fed with input features that are only available at the start of the call. However, this approach resulted in a large rate of false positives and an overall accuracy of 45% to detect an IRSF call. Another study [14] performed a comprehensive study on the nature of International Premium Rate Number (IPRN) and IRSF. Firstly, the authors defined a set of features based on relations between known test IPRNs and the origin and destination numbers of the calls. Then, using these features in conjunction with a supervised Random Forest algorithm, a data-driven model was developed, obtaining a 98% accuracy with a 0.28% false positive rate in detecting fraudulent calls. Yet, the study relied on a set of known IPRNs that tend to dynamically evolve over time. Thus, the proposed approach requires a continuous IPRN database collection, that changes regularly and needs to be kept up to date, an issue that is also present in the features. One interesting aspect studied in [14] is that the IRSF scam can happen when considering only certain operators or when the call is initiated from a particular geographic point, since some types of IRSF frauds rely on the routes that the call follows from origin to destination. This means that, at times, the classification of a call as a scam call depends on the context in which it happens, which can increase the difficulty of the ML learning task.

Federated Learning is a modern technique that trains ML models using local data that is not shared. Then, it aggregates the local models to generate a global model that contains all the local models knowledge. It presents itself as a potentially good choice to solve this context problem, as each local model is trained under its own context.

2.3 Federated Learning

In the past decade, diverse studies have approached ML in Edge Computing and MEC scenarios to optimize the planning of the architecture [15] or develop operational intelligence [16]. Several of these works are based on centralized approaches that train the ML models in a single location using all the data generated from the edge devices.

In the last few years, there has been an attempt to train models without sharing the data between edge devices, using Federated Learning concepts, which were the works we analyzed in more depth. Table 1 summarizes the related works that use Federated Learning in Edge Computing scenarios in terms of the following columns: **Ref.** – the study reference; **ML** – which ML techniques were used in the study (we adopt the DL acronym to refer to shallow and deep structures of neural architectures); **FL** – if the work uses a Federated Learning; **HC** – if the approach explicitly considers the hardware constraints of the edge devices for ML training and inference; **EC** – if the study is applied to an Edge Computing scenario; **MEC** – if the study is applied to a MEC scenario; **5G** – if the work considers 5G; **Telecom** – if the approach is applied to the Telecom domain; and **FL Approach** – brief explanation of the Federated Learning approach.

Most of the analyzed works are from 2020 and 2021 (seven studies from each year), which confirm that Federated Learning and Edge Computing are trending research topics. The majority of the related works are related to MEC architectures, even though some studies consider other paradigms of Edge Computing (e.g., Vehicular Networks [17]). Similar to our approach, most studies use Deep Learning models, even though distinct neural network architectures are used among these works (e.g., Convolutional Neural Networks, Long Short-Term Memory). Only five works do not use Deep Learning, using instead Deep Reinforcement Learning [18–20], classical ML algorithms [21], and optimization methods [22]. Around half (11) of the studies consider the hardware constraint of the edge devices in the study. However, contrary to this work, none of the studies use ML frameworks especially designed for edge devices (e.g., TensorFlow Lite). Of the 20 surveyed works, only one considers 5G networks [20]. In addition, none of the works are applied to the telecommunications domain. Regarding the research topics, seven works use existing Federated Learning techniques (e.g., Federated Averaging) to develop ML applications from decentralized data. Other works propose new Federated Learning frameworks, for example, developing new techniques for model aggregation (e.g., [23, 24]) or to prevent network attacks (e.g., [25]).

3 Proposed Architecture

This paper is part of the R&D project "Opti-Edge", developed by a leading Portuguese software company in the area of telecommunications. One of the expected results of the project is the development of a ML architecture that

Table 1 Summary of the related work (Federated Learning in Edge Computing scenarios).

Year	Ref.	ML	FL	HC	EC	MEC	5G	Telecom	FL Approach
2018	[26]	DL	✓		✓				Application of existing FL techniques
2019	[27]	SVM; LR K-means; DL	✓	✓	✓	✓			New FL approach (control local and updates)
2019	[19]	DRL	✓		✓	✓			New FL approach (optimize caching)
2019	[21]	GBDT	✓		✓	✓			New FL approach (asynchronous FL)
2019	[18]	DRL	✓	✓	✓				Application of existing FL techniques
2019	[22]	Opt.	✓	✓	✓	✓			Application of existing FL techniques
2020	[23]	DL	✓	✓	✓				New FL approach (aggregation technique)
2020	[28]	DL	✓		✓				New FL approach (selective aggregation)
2020	[29]	DL	✓		✓	✓			Application of existing FL techniques
2020	[30]	DL	✓		✓	✓			Application of existing FL techniques
2020	[31]	DL	✓	✓	✓	✓			Application of existing FL techniques
2020	[32]	DL	✓		✓				New FL approach (gradient sparsification)
2020	[33]	DL	✓	✓	✓	✓			New FL approach (model partition technique)
2021	[20]	DRL	✓	✓	✓	✓	✓		Application of existing FL techniques
2021	[25]	DL	✓		✓				New FL approach (prevent network attacks)
2021	[34]	DL	✓	✓	✓	✓			New FL approach (two levels of aggregation)
2021	[35]	DL	✓	✓	✓	✓			FL literature review
2021	[24]	DL	✓		✓	✓			New FL approach (aggregation technique)
2021	[36]	DL	✓	✓	✓	✓			New FL approach (model compression)
2021	[17]	DL	✓	✓	✓				New FL approach (client selection algorithm)
2022	this work	DL	✓	✓	✓	✓	✓	✓	Application of existing FL techniques (5G and Telecom domain)

DL - Deep Learning; DRL - Deep Reinforcement Learning; FL - Federated Learning; GBDT - Gradient Boosting Decision Trees; LR - Linear Regression; Opt. - Optimization; SVM - Support Vector Machines

allows the company to train and deploy ML models at the edge, responding to a set of realistic assumptions made by the company.

First, to ensure data privacy, the data (e.g., phone calls) must not be shared between edge nodes, meaning that each edge must train ML models only with its local data. Second, the computation effort should be minimal, by using lighter ML models. This assumption is also related to sustainability concerns, to reduce the carbon footprint of the data centers. Third, the latency for the inference should be low, since the company needs to quickly detect fraudulent behaviour. Lastly, the fourth assumption is related to reliability, since if one of the edges is compromised the others should be able to keep responding.

To tackle these challenges of project Opti-Edge, we propose an architecture that uses concepts of Federated Learning and lighter ML algorithms. The proposed architecture consists of two main components: the edge nodes and a centralized location (Fig. 2).

3.1 Edge Nodes

In the proposed architecture, the main steps made by each edge node are inference (generate predictions) and local training. Fig. 3 represents the internal operation of the edge node (the left side is related to the inference task and the right side is related to the training task). At every moment, there is a common ML model that is shared across all edge nodes (copies of the same model). This model is a TensorFlow Lite model, a format used to reduce inference time and disk space occupation. This model is used every time there is the need to make predictions. In contrast with the inference task, the training

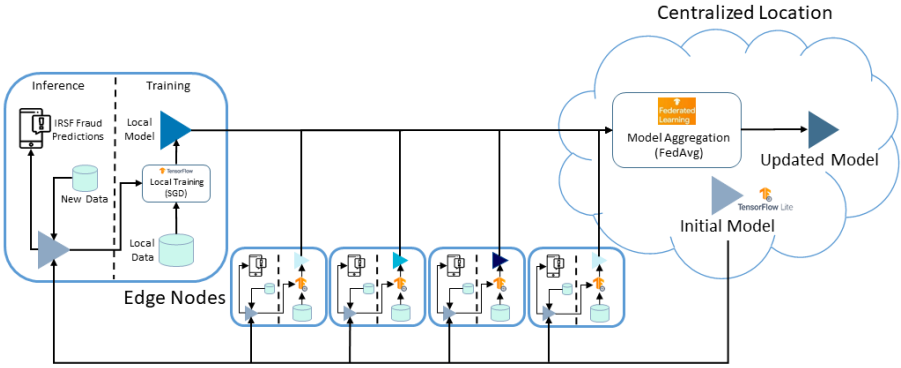

Fig. 2 The proposed ML architecture for Federated Learning at the edge (adapted from [7]).

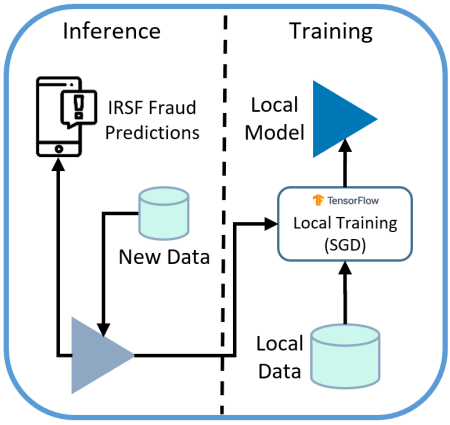

Fig. 3 Representation of an edge node (adapted from [7]).

of new ML models is typically more sporadic. When there is a need to update the main ML model, each edge node uses its local data to train a model.

3.2 Centralized Location

The proposed architecture also identifies a centralized location (e.g., cloud), which performs the tasks of aggregation and sharing of the main model. The aggregation phase is responsible for updating the main ML model. Since the learning step is decentralized (i.e., every edge node trains using its local data), this step is used to aggregate all local models into a single model. The aggregation technique is Federated Averaging (FedAvg) [37], widely used in the context of Federated Learning. This technique is explained in more detail in Section 4.3. The sharing step is usually processed right after the aggregation. The sharing consists of sending a copy of the new aggregated model to each edge node, replacing the previous model.

4 Materials and Methods

4.1 Data

For the practical experiments, we used data provided by the analytics software company, related to IRSF. The analyzed data consists of two datasets that contain phone call records from one of the company's clients. The first dataset included one month of phone calls (from July 1, 2019, to July 31, 2019), corresponding to 78,174 records. The second and larger dataset contains a total of 1,948,504 records related with four months of data (from August 1, 2020 to November 30, 2020). All data is anonymized to safeguard the companies and their customers. Both datasets have a similar structure, with the same 35 column attributes. Since most columns were either constant or empty, we opted to remove these non informative attributes from both datasets, resulting in final datasets that contain the 14 attributes described in Table 2.

Table 2 Description of the adopted IRSF attributes (adapted from [7]).

Attribute	Description	Type	Example
EVENT_TYPE	Type of event that originated the call	Categorical	MOC
SERVED _ENTITY_ID	ID of event	Categorical	352123456784
IMSI	International Mobile Subscriber Identity	Categorical	270123456784999
IMEI	International Mobile Equipment Identity	Categorical	15412345678179
A_NUMBER	Call sender phone number	Categorical	352123456765
B_NUMBER	Call receiver phone number	Categorical	352123456717
START_DATE	Start date of event	Date	20190703092145
DURATION	Duration of event	Numerical	2
CALL_TYPE	Type of phone call	Categorical	Voice
CHARGE_AMOUNT	Amount charged during call	Numerical	3.73
CELL_ID	Cellphone identifier	Categorical	9998
ROAMING_INDICATOR	Usage of roaming ($\in \{0, 1\}$)	Categorical	1
COUNTRY_CODE	Receiver country (ISO 3166-1)	Categorical	351
APN	Classification of call between "normal" or "fraud" ($\in \{0, 1\}$)	Categorical	1

The data included a target column (APN) that classifies a call between "normal" and "fraud", thus this attribute is modeled in this work assuming a supervised learning binary classification. Fig. 4 shows the distribution of the two classes for both datasets. As is usual within the fraud domain [14], in both datasets most phone calls were considered "normal", with only a small percentage of records being considered "fraud". For the first batch of data, around 15.71% of the records were labeled as fraud (around 12,000 calls). The second batch of data was more unbalanced, containing only 2.48% of "fraud" records (around 48,000 calls).

Table 3 shows the number and percentage of missing values and unique values for each attribute. For the only column that presented missing data (COUNTRY_CODE), we replaced the missing values with zero, which is assumed as a numeric code value for the "unknown" level. Given that most

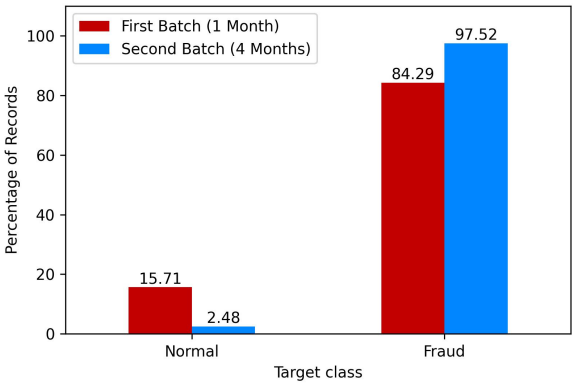

Fig. 4 Distribution of the target classes (Normal and Fraud) for the two datasets.

columns were categorical (which are not directly handled by numeric based ML algorithms), we opted to encode all categorical attributes into numerical values by using a Label Encoding, which transforms the values of the attribute into integers (between 1 and the number of levels of that attribute). In cases where a new category that was not present in the train data appears on test data, we encoded these values to zero, corresponding to an "unknown" category. We chose this technique because it is easy to implement and it does not generate new columns, resulting in data-driven models that are faster to be trained and that require less memory. It should be noted that most categorical attributes have a high cardinality (e.g., IMSI), thus using an one-hot encoding would heavily increase the number of inputs of the ML models.

Table 3 Missing values and unique values for the IRSF datasets.

Attribute	1st Batch (1 month)				2nd Batch (4 months)			
	Missing Values		Unique Values		Missing Values		Unique Values	
	No.	%	No.	%	No.	%	No.	%
EVENT_TYPE	0	0	4	<1	0	0	4	<1
SERVED _ENTITY_ID	0	0	7725	10	0	0	25248	1
IMSI	0	0	9599	12	0	0	32632	2
IMEI	0	0	2318	3	0	0	6430	<1
A_NUMBER	0	0	4673	6	0	0	13791	<1
B_NUMBER	0	0	4687	6	0	0	13775	<1
START_DATE	0	0	-	-	0	0	-	-
DURATION	0	0	-	-	0	0	-	-
CALL_TYPE	0	0	3	<1	0	0	3	<1
CHARGE_AMOUNT	0	0	-	-	0	0	-	-
CELL_ID	0	0	20	<1	0	0	20	<1
ROAMING_INDICATOR	0	0	2	<1	0	0	2	<1
COUNTRY_CODE	42196	54	28	<1	985583	51	28	<1
APN	0	0	2	<1	0	0	2	<1

4.2 ML Framework

The base framework for the development of the experiments was TensorFlow, an open-source ML library for graph-based numerical computing [38]. In particular, we used two modules of the TensorFlow API: TensorFlow Lite and TensorFlow Federated. TensorFlow Lite is a 'lighter' implementation of TensorFlow, developed especially for resource-constrained devices (e.g., smartphones, IoT devices). TensorFlow Lite allows the development and deployment of Deep Learning modules on edge architectures. It is compatible with Linux and also mobile operating systems, such as Android and iOS. TensorFlow Federated is the TensorFlow module focused on ML models trained with decentralized data. This module was developed to work within Federated Learning scenarios, where a centralized model is shared across many clients.

4.3 Federated Training and Aggregation

One of the main decisions when applying Federated Learning is the approach used to unify the local models trained with local (i.e., decentralized) data, similar to this project where one of the requirements is that there is no sharing of data between clients. The aggregation approach we used in this paper was the FedAvg algorithm proposed in [37]. The FedAvg algorithm assumes the existence of one common model that is shared across several clients. Having a copy of the common model, each client applies the Stochastic Gradient Descent (SGD) method using its local data. In practice, to generate a local model, each client uses the current model and its local data as input. During the local training, the client typically applies one step of SGD to the main model, resulting in a slightly different model.

The aggregation phase of FedAvg generates a single model based on the local models. This aggregation is similar to a weighted average of the weights of each model, using Eq. (1).

$$w_{t+1} \leftarrow \sum_{k=1}^{n} \frac{n^k}{n} w_{t+1}^k \tag{1}$$

In this equation, w_{t+1} represents the new main model, n^k is the number of records using during the training of model of client k, n is the total number of records (sum of all local models) and w_{t+1}^k represents the weights of the local model of client k. A representation of this technique is shown in Fig. 5.

4.4 Initial Model Architecture

One of the assumptions of FedAvg is that during the Federated Learning iterations the architecture of the ML model does not change, only the weights are updated. Given the constraints of the project, we designed a neural network with few layers and neurons. In particular, it consists of a multilayer perceptron network with four hidden layers, ReLU activation functions in

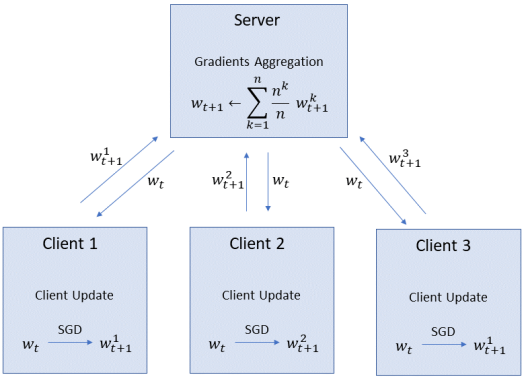

Fig. 5 Representation of the FedAvg technique.

all processing neurons except the output one, which assumes a logistic function (Fig. 6). The models are executed during 100 epochs, assuming also an

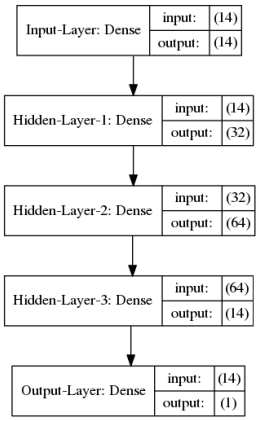

Fig. 6 Architecture of the neural network used for the initial model (adapted from [7]).

early stopping procedure that considered three epochs. We used binary cross-entropy as loss function, the Adam optimizer and we computed the Area Under the Curve (AUC) as the validation and evaluation metric.

The AUC measure is based on the Receiver Operating Characteristic (ROC) analysis [39]. When a binary classifier outputs a decision score $d_i \in [0, 1]$, the class can be interpreted as "positive" if $d_i > K$, where K is a decision threshold, otherwise it is considered "negative". With the obtained class label predictions, there will be True Positives, True Negatives, False Positives, and False Negatives. The ROC curve shows the False Positive Rate or one minus the specificity (x-axis) versus the True Positive Rate or sensitivity (y-axis), for all possible threshold values. The AUC measure has two main advantages. First, the quality values are not affected by unbalanced data (which is our

case). Second, the AUC values can be easy interpreted as [40]: 0.5 – random classifier, 0.7 – reasonable; 0.8 – very good; 0.9 – excellent; 1.0 – perfect. Fig. 7 exemplifies one of the ROC curves related to the predictive results (Table 4). For this ROC curve, two distinct thresholds were selected. The more sensitive and lower threshold is colored in green, while the more specific and higher threshold point is colored in red.

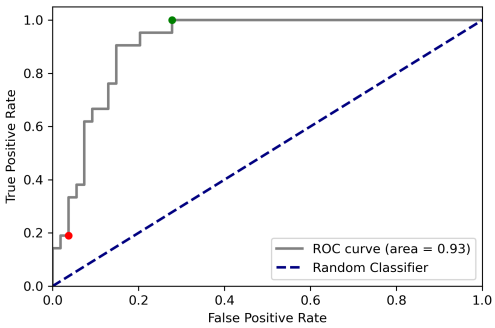

Fig. 7 Example of a ROC curve related to the predictive experiments (shown in Table 4).

5 Results

5.1 Experimental Evaluation

For the computational experiments we simulated, using three nodes as edges and one node as the central node, the proposed Federated Learning architecture to detect IRSF (described in Section 3). We performed two Federated Learning experiments, using the two provided datasets, including one month and four months of data. For each dataset, we used the first half of the data (ordered by timestamp) to train an initial model inside the centralized node, using the neural network architecture described in Section 4.4. Then, we shared the obtained models (in TensorFlow Lite format) across the three edge nodes.

During two iterations of local training and aggregation, we used the second half of each dataset to train local models on all edge nodes, send each local model to the cloud, aggregate the local models, and, lastly, share the new main model across all edges. For each round of the local training, the first 75% of the data was distributed among the edges, separating the records by phone codes (e.g., +351). The remaining 25% of data for each round was used as a common test dataset on the central node, to evaluate the predictive performance of the ML models. Fig. 8 shows the division strategy used for each dataset.

We note that we focused our experimental evaluation exclusively on the predictive performance of the ML models, in particular on the decentralized approach. We did not evaluate latency (e.g., to send the models to the central node) since we considered that those values were not significant for these simulated experiments. Given that the TensorFlow Lite models are rather

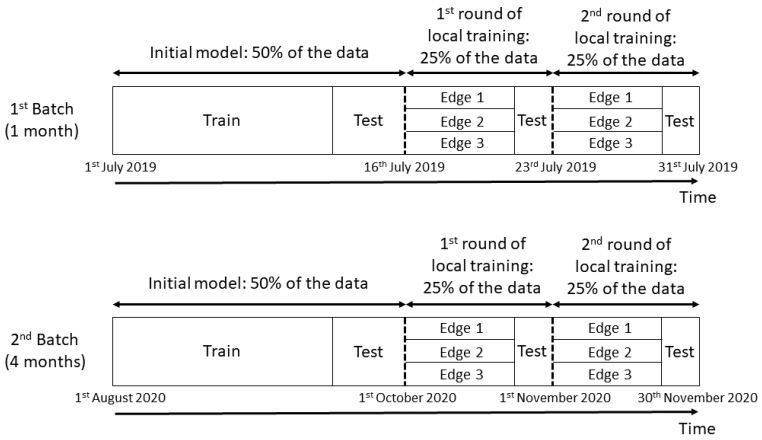

Fig. 8 Division of the dataset between rounds of local training (adapted from [7])

small (around 100 KB) and the transfer of the models between the edge nodes and the central node in our environment was less than ten milliseconds, even in an extreme scenario where we train the local models every day, this would imply spend less than one second per day.

5.2 Local Training Results

Table 4 shows the obtained results for the conducted experiments. For each of the two used datasets, the table details the obtained test set AUC across the three Federated Learning steps: the initial round, the first round of local training, and the second round of local training. For each Federated Learning round, the obtained AUC includes the scores both for the local models (trained on the edge nodes) and for the main model (**Main**), which resulted from the aggregation of the local models. Since the two initial models (one for each dataset) were trained with centralized data, we did not include the local model results for that round. In each round, the AUC represents the result obtained on the test set for that round, meaning that, for each round, the test set was the same for all models (local and main). In Table 4, we also included the score of a Random Forest model that was developed by the company using a pure centralized approach. This ML model is used for comparison purposes since it is currently used by the software company for detecting IRSF.

For the first dataset (one month of data), the results show that during the two rounds of local training, the obtained AUC values on the test set can be considered of excellent quality (always higher than 0.96), both for the local models and for the main models (which aggregate all local models of that round). Moreover, during the two rounds of local training, the main model was able to maintain the AUC value obtained by the initial model, which was trained using only centralized data. Another interesting result is that for the first and second rounds of local training, the performance of each of the three local models (one for each edge node) was slightly lower than the performance

Table 4 Results obtained by the local models and the main models during the Federated Learning iterations (adapted from [7]).

Dataset	Round	Measure	Model Used to Predict			
			Edge 1	Edge 2	Edge 3	Main
1st Batch (one month)	Initial Model Round	AUC	-	-	-	0.980
	Local Training (1st Round)		0.974	0.977	0.973	0.981
	Local Training (2nd Round)		0.976	0.964	0.975	0.980
2nd Batch (four months)	Initial Model Round	AUC	-	-	-	0.980
	Local Training (1st Round)		0.934	0.969	0.969	0.979
	Local Training (2nd Round)		0.969	0.966	0.972	0.980
Company model (Random Forest)		AUC	-	-	-	0.932

of the main model. Those results might be explained by the fact that the test sets for each round include phone calls from the three edge nodes. Thus, it is not expected that a local model can perform as well as the main model. However, the maximum predictive difference between a local model and the respective main model was less than 1 *percentage point (pp)*.

The second dataset (which considered a larger period and 25 times more records) produced consistent predictive results when compared to the first batch of data. The obtained AUC values were also of excellent quality, with the minimum value of 0.934 obtained by one of the local models. The performance of the main model was also consistent during the local training rounds, maintaining the value of the initial model. Similar to the first dataset results, the four months dataset always presented higher AUC values for the main model when compared to the local models of that round. However, for the second dataset, the differences between the local models and the respective main model were higher. The maximum difference between a local model for the first round of local training and the main model was 4.5 *pp* and for the second round was 1.4 *pp*. While the maximum differences for the one-month data were smaller, this might be due to the fact that the second dataset had significantly more records than the first dataset.

Nonetheless, the obtained results are a good indicator that the Federated Learning process helps the ML model to maintain its performance when predicting the existence of IRSF, even when considering different intervals for the local updates. This capability is especially important in the telecommunication domain since the ML models typically require many updates. This need is mainly due to the fact that Telecom companies frequently generate a huge amount of data (e.g., phone calls) and that new fraud methods are constantly being developed by the fraudulent agents [4, 14].

Comparing the obtained results with the Random Forest model, it is possible to verify that for all local training rounds of both datasets, the initial model and the other main models present a higher AUC than the baseline model, with an average difference of 4.8 *pp*. However, a word of caution is necessary. Our main goal with this work was to assess the performance of the proposed ML approach, by comparing the performance of the local models with the main models, since the value of the application is that the learning is performed in a decentralized manner, keeping the data processing local to the edges and respecting data privacy constraints made by the company. It is worth noticing that the Random Forest model might have used a different time period during its training. Also, unlike our approach, this process used different preprocessing procedures and was developed with a centralized dataset. Thus, the comparison of our approach with the baseline prediction model only serves the purpose of demonstrating that the decentralized models can achieve AUC values that can match the company's current production model, even though we cannot make a direct comparison. Nevertheless, this is another indicator of the potential of the use of Federated Learning at the edge within the domain of telecommunications.

5.3 Model Degradation Analysis

Additionally, after the previous experiment we evaluated the performance degradation of the main models. To do that, we used every main model available to predict on the test data in each Federated Learning round. This means that, for the initial round we only used the initial model, since it was the only model available; for the first local training round, we used the main model from this round and also the initial main model; for the second local training round, we used all three main models. Our main goal was to monitor the performance degradation of the main models and the need for model updates.

The obtained results are presented in Table 5, which confirm that the performance of the main models decreases over time, when the models are applied to more recent data. It should be noted that the differences are rather small (maximum difference of less than 1 *pp*), which can be justified by the small interval of the datasets (one month and four months). As other works that apply ML to the Telecom domain show, the performance of the models can stay stable for a few months and then drastically drop. For example, in [41] the authors apply ML algorithms to predict churn for the Telecom domain, with some of the algorithms maintaining almost perfect Accuracy (around 100%) during six months and then dropping to around 10% on the seventh month. These performance deterioration patterns are common within the Telecom domain, due to a continuous "arms race" between the fraudsters and the anti-fraudsters, where the anti-fraudsters introduce a new technology to block frauds, which forces the fraudsters to develop new fraudulent techniques and so on. Nevertheless, the results obtained in this article help support the need for constant updates of the ML models, which can be achieved with Federated Learning when the data is decentralized and there are privacy constraints.

Table 5 Results obtained by previous main models on new data (adapted from [7]).

Dataset	Round	Measure	Main Model Used to Predict		
			Initial Round	1st Round	2nd Round
1st Batch (one month)	Initial Model	AUC	0.980	-	-
	Local Training (1st Round)		0.979	0.981	-
	Local Training (2nd Round)		0.977	0.978	0.980
2nd Batch (four months)	Initial Model	AUC	0.980	-	-
	Local Training (1st Round)		0.979	0.979	-
	Local Training (2nd Round)		0.971	0.979	0.980

6 Demonstrator Technological Architecture

The experimental evaluation in Section 5 confirms that the Logical Architecture proposed in Section 3 has value as a distributed architecture for detecting IRSF while assuring data privacy. To verify that that architecture was viable in a real-life environment, a prototype demonstration was built using a selection of technologies agreed upon with the telecommunications company.

It is important to notice that, on this paper scope, this prototype was not subjected to performance tests, and some realist assumptions were made. In particular, regarding latency, it was considered that due to the nature of edge technologies and since the inference model is closer to the end user, the use latency of said model is lower than the use latency of a model executing on a remote cloud location.

Regarding the communication latency that happens during model transference between the edges and the cloud, and vice-versa, we consider that this step will happen sparsely and thus latency is not considered an issue. First it is expected that this latency is lower enough to be irrelevant within the update frequency of the models (currently, at most once a day). Secondly, the current solution, used by the Telecom company, is based on a centralized solution that requires the edge data to be transferred to the cloud and the volume of the data, from our observations, is, at least, two orders of magnitude higher than the volume of the generated inference models, meaning that the proposed solution should have a much lower communication latency than the current solution.

The Federated Learning architecture presented in Section 3 was implemented, as a prototype demonstration, in a MEC based on Open Network Edge Services Software (OpenNESS). OpenNESS is a software toolkit for optimizing and performing edge platforms, integrating and managing network

applications and functions with cloud-like agility across any type of network[2]. This implementation was based on OpenNESS clusters, using one cluster as cloud platform and others as edges. The technological architecture for this implementation is presented in Fig. 9.

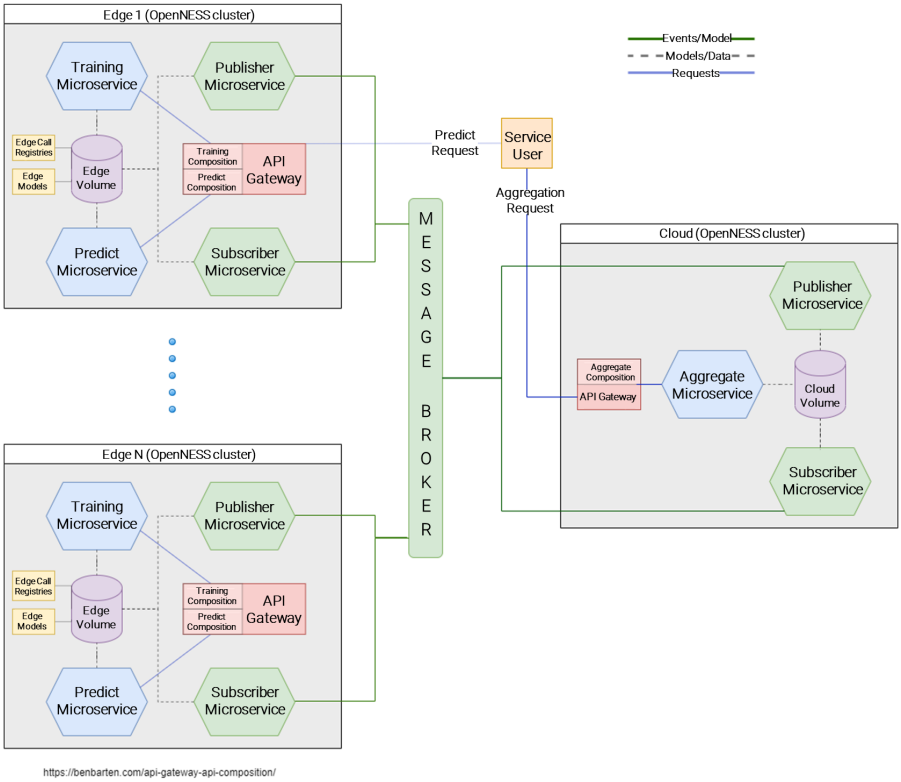

Fig. 9 Technological implementation of the proposed Federated Learning architecture.

The implementation follows a micro-service architecture [42] where, in the cloud, an aggregation micro-service is kept while on the edges the predict and training micro-services are both kept, all following a service per container pattern. For the interaction among the micro-services, a one-to-many interaction style is used, which is based on an asynchronous publish/subscribe service, ensured by a message broker RabbitMQ and by publishing and subscribing micro-services on each edge and cloud.

The one-to-many interaction style decouples the aggregation, predict and training micro-services keeping the dependencies among them to a minimum. For such architecture, scaling, failure handling, or integration tasks of new services are simplified. In particular, any resource-adequate node can substitute the central node and new configurations may be experienced. In the future, this

[2] https://www.openness.org/docs/doc/overview

solution may be integrated with the operators MEC platforms that provide sophisticated service instantiation and orchestration among MEC nodes.

For the service client interaction with the application, a one-to-one synchronous interaction style is used, based on RESTful APIs implemented by an API gateway. This API gateway also implements a service composition that orchestrates the interactions between the micro-services. The needed data storage for each service, both on the edges and the cloud, follows a shared database pattern, where on each edge and on the cloud there is a data volume for all the micro-services running on that edge or cloud.

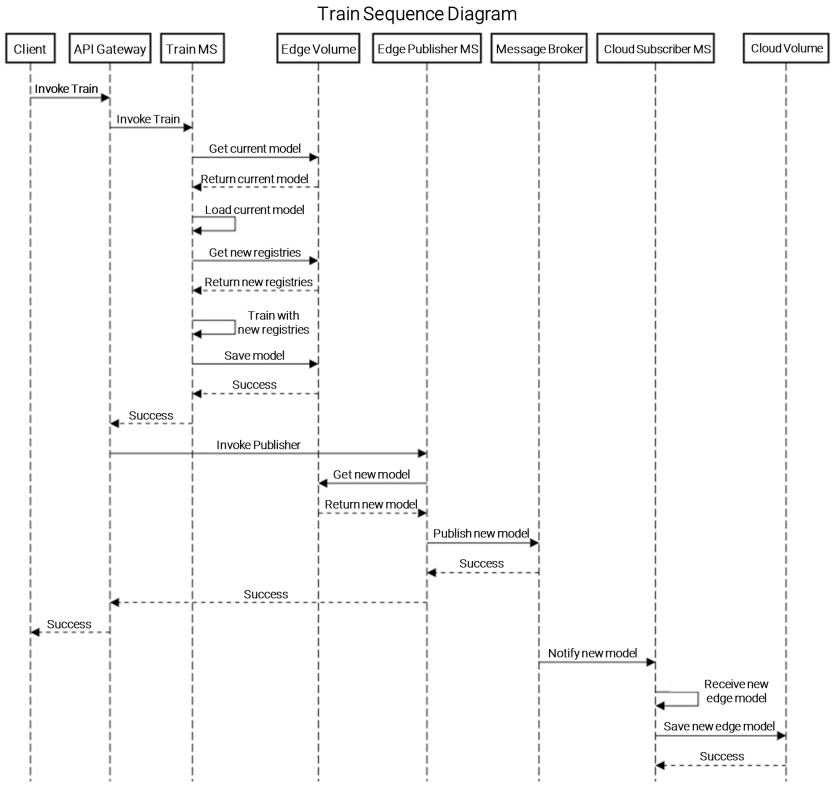

Fig. 10 Sequence diagram for the training service.

The training micro-service follows the sequence diagram of Fig. 10. This service is invoked regularly by a client service and starts by fetching the model used by the prediction service from the edge volume. This model is then trained with the new registries that are present on that edge and saved to the edge volume. Once the training process finishes, the Edge Publisher publishes the new model on the message broker, which in turn, notifies the Cloud Subscriber micro-service that saves the model to the cloud storage.

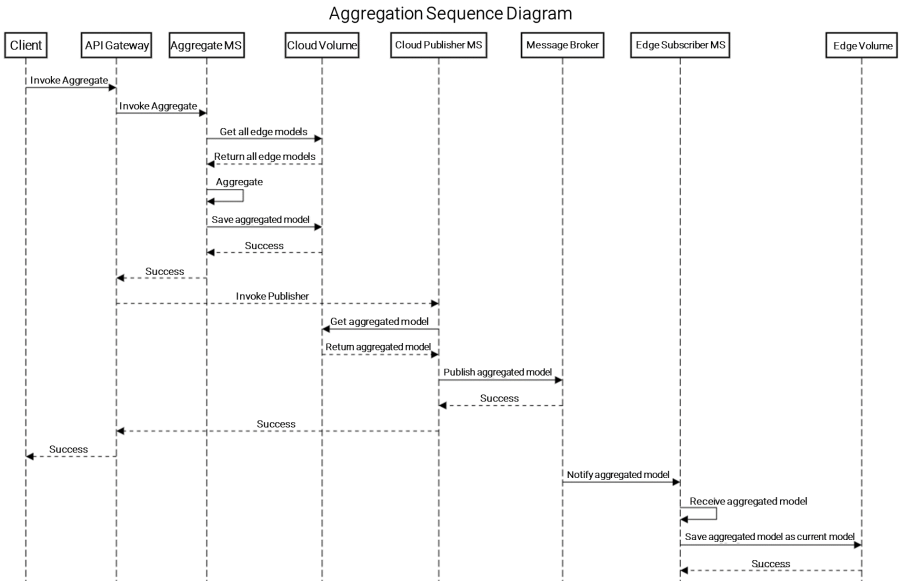

Fig. 11 Sequence diagram for the aggregation service.

The aggregation service follows the sequence diagram of Fig. 11. This service is invoked regularly by a client service and starts by fetching all edge models from the cloud volume, the models that were previously received from the training service. These models are then aggregated and the resulting model is saved to the cloud storage. The cloud publishing service then publishes this model to the message broker that notifies all the edges. The edges then save this model to the storage and this model will be used by the predicting service.

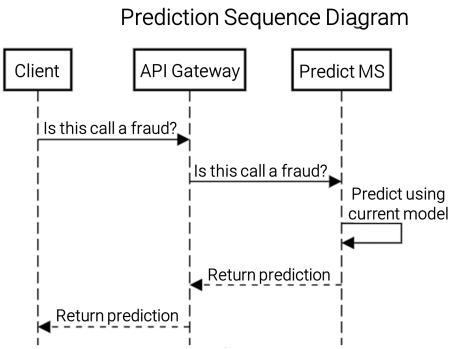

Fig. 12 Sequence diagram for the prediction service.

The prediction service follows the sequence diagram of Fig. 12. This service is invoked whenever a client needs to predict if a call is a fraud call or not. Once it is invoked, the predict micro-service uses the last main model to predict if the call is a fraud or not.

7 Conclusions

With the rise of Edge Computing and 5G networks, companies are able to design and host applications closer to the data sources, lowering latency and increasing performance and efficiency. One of the primary issues of Edge Computing for ML applications is the need to perform the training using decentralized data, which is usually accomplished via Federated Learning approaches. In the last few years, there have been many works that use Federated Learning concepts in Edge Computing scenarios. For the Telecom domain, one of the main applications of ML is related to fraud detection, such as IRSF. However, within our knowledge there are no research works that predict IRSF in a decentralized manner, nor considering 5G networks.

In this work, we propose a framework that can train and update ML models using decentralized data. The framework was developed for the Opti-Edge R&D project, which includes a major provider of analytics solutions for the Telecom domain. Within this domain, typically there are several edge nodes that cannot share data between each other or with the cloud (e.g., for confidentiality reasons). Thus, our proposed framework uses Federated Learning to aggregate models that were trained with different data.

To evaluate the framework, we simulate two rounds of federated training and aggregation using two datasets provided by the Telecom analytics company concerning the IRSF fraud: the first dataset considering one month of data (about 78,000 records); the second dataset including four months of calls (almost 2,000,000 records). Then, we compare the obtained results for both datasets with a baseline model, developed before this work using centralized data. The results show that the decentralized approach (which used Federated Learning) produces an excellent level of class discrimination (always higher than 0.93 of AUC) and it can maintain its performance across two rounds of local training (with weekly or monthly periodicity). In effect, the proposed framework received very positive feedback from the software company. After validating the results with the company, we build a prototype technological architecture to deploy the proposed framework in a real-world MEC scenario.

In future work, we wish to further evaluate the framework with more datasets from the telecommunication domain and consider longer periods of time (e.g., one year), to verify the consistency with our results and to better evaluate model degradation during the model updates. Moreover, we intend to apply concepts of Automated Machine Learning (AutoML) and Neural Architecture Search (NAS) to automate the design of the neural network architectures for the initial ML models. We also intend to perform a sensitivity analysis on the aggregation frequency of the models, since we recognize that in certain scenarios that require a high model update frequency, the communication latency may become an issue. Additionally, we intend to evaluate the robustness of the Federated Learning approach regarding attacks to the local edges (e.g., verify if the attacks propagate to the main models). Finally, we aim to develop a new Federated Learning architecture that does not require either a centralized module in the cloud or a previously trained initial model.

Acknowledgments

This work was executed under the project Opti-Edge: 5G Digital Services Optimization at the Edge, Individual Project, NUP: POCI-01-0247-FEDER-045220, co-funded by the Incentive System for Research and Technological Development, from the Thematic Operational Program Competitiveness of the national framework program - Portugal2020. We wish to thank the anonymous reviewers for their helpful comments.

Conflict of Interest Statement

The authors declare no conflict of interest.

References

[1] Najmul Hassan, Kok-Lim Yau, and Celimuge Wu. Edge Computing in 5G: A Review. *IEEE ACCESS*, 2019. doi: 10.1109/ACCESS.2019.2938534.

[2] Yuan Ai, Mugen Peng, and Kecheng Zhang. Edge computing technologies for internet of things: a primer. *Digital Communications and Networks*, 4(2):77–86, 2018. doi: 10.1016/j.dcan.2017.07.001.

[3] ETSI. Multi-access edge computing (mec) framework and reference architecture. *ETSI GS MEC*, 3, 2019.

[4] Black Swan Telecom Journal. International Revenue Share Fraud: Are we winning the battle against telecom pirates? http://bswan.org/revenue_share_fraud.asp, 2012.

[5] Mark Yelland. Fraud in mobile networks. *Computer Fraud & Security*, 2013. doi: 10.1016/S1361-3723(13)70027-7.

[6] European Union. General Data Protection Regulation (GDPR). https://gdpr.eu/tag/gdpr/, 2016.

[7] Luís Ferreira, Leopoldo Silva, Diana Pinho, Francisco Morais, Carlos M. Martins, Pedro M. Pires, Pedro Fidalgo, Helena Rodrigues, Paulo Cortez, and André Pilastri. A federated machine learning approach to detect international revenue share fraud on the 5g edge. In *37th Symposium on Applied Computing*. ACM, 2022. doi: 10.1145/3477314.3507322.

[8] IEEE. Ieee 5g and beyond technology roadmap [white paper]. 2017. URL https://futurenetworks.ieee.org/images/files/pdf/ieee-5g-roadmap-white-paper.pdf.

[9] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal*, 5(1):450–465,

2018. doi: 10.1109/JIOT.2017.2750180.

[10] Merve Sahin, Aurélien Francillon, Payas Gupta, and Mustaque Ahamad. Sok: Fraud in telephony networks. In *2017 IEEE EuroS&P*, 2017. doi: 10.1109/EuroSP.2017.40.

[11] Communications Fraud Control Association. Fraud Loss Survey Report 2021. https://cfca.org/wp-content/uploads/2021/12/CFCA-Fraud-Loss-Survey-2021-2.pdf, 2021.

[12] Mobileum. Revenue Share Fraud. www.mobileum.com/products/risk-management/fraud-management/revenue-share-fraud/, 2022.

[13] Yoram Meijaard, Bram Cappers, Josh Mengerink, and Nicola Zannone. *Predictive Analytics to Prevent Voice over IP International Revenue Sharing Fraud*, pages 241–260. 2020. doi: 10.1007/978-3-030-49669-2_14.

[14] Merve Sahin and Aurelien Francillon. Understanding and Detecting International Revenue Share Fraud. In *28th NDSS, virtually, February 21-25, 2021*, 2021. doi: 10.14722/ndss.2021.24051.

[15] Shailendra Rathore, Pradip K. Sharma, Arun K. Sangaiah, and James J. Park. A hesitant fuzzy based security approach for fog and mobile-edge computing. *IEEE Access*, 2018. doi: 10.1109/ACCESS.2017.2774837.

[16] Ahmed Ghoneim, Ghulam Muhammad, Syed Umar Amin, and Brij B. Gupta. Medical image forgery detection for smart healthcare. *IEEE Commun. Mag.*, 56(4):33–37, 2018. doi: 10.1109/MCOM.2018.1700817.

[17] Huizi Xiao, Jun Zhao, Qingqi Pei, Jie Feng, Lei Liu, and Weisong Shi. Vehicle selection and resource optimization for federated learning in vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, 2021. doi: 10.1109/TITS.2021.3099597.

[18] Jianji Ren, Haichao Wang, Tingting Hou, Shuai Zheng, and Chaosheng Tang. Federated learning-based computation offloading optimization in edge computing-supported internet of things. *IEEE Access*, 7: 69194–69201, 2019. doi: 10.1109/ACCESS.2019.2919736.

[19] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. In-edge AI: intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Netw.*, 33(5):156–165, 2019. doi: 10.1109/MNET.2019.1800286.

[20] Shuai Yu, Xu Chen, Zhi Zhou, Xiaowen Gong, and Di Wu. When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5g

ultradense network. *IEEE Internet Things J.*, 8(4):2238–2251, 2021. doi: 10.1109/JIOT.2020.3026589.

[21] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan, and Yan Zhang. Differentially private asynchronous federated learning for mobile edge computing in urban informatics. *IEEE Trans. Ind. Informatics*, 16 (3):2134–2143, 2020. doi: 10.1109/TII.2019.2942179.

[22] Yongfeng Qian, Long Hu, Jing Chen, Xin Guan, Mohammad Mehedi Hassan, and Abdulhameed Alelaiwi. Privacy-aware service placement for mobile edge computing via federated learning. *Inf. Sci.*, 505:562–570, 2019. doi: 10.1016/j.ins.2019.07.069.

[23] Yunfan Ye, Shen Li, Fang Liu, Yonghao Tang, and Wanting Hu. Edgefed: Optimized federated learning based on edge computing. *IEEE Access*, 8: 209191–209198, 2020. doi: 10.1109/ACCESS.2020.3038287.

[24] Wenyu Zhang, Xiumin Wang, Pan Zhou, Weiwei Wu, and Xinglin Zhang. Client selection for federated learning with non-iid data in mobile edge computing. *IEEE Access*, 2021. doi: 10.1109/ACCESS.2021.3056919.

[25] Jiale Zhang, Bing Chen, Xiang Cheng, Huynh Thi Thanh Binh, and Shui Yu. Poisongan: Generative poisoning attacks against federated learning in edge computing systems. *IEEE Internet Things J.*, 8(5):3310–3322, 2021. doi: 10.1109/JIOT.2020.3023126.

[26] Zhengxin Yu, Jia Hu, Geyong Min, Haochuan Lu, Zhiwei Zhao, Haozhe Wang, and Nektarios Georgalas. Federated learning based proactive content caching in edge computing. In *IEEE GLOBECOM 2018, Abu Dhabi, 2018*, pages 1–6. IEEE, 2018. doi: 10.1109/GLOCOM.2018.8647616.

[27] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE J. Sel. Areas Commun.*, 37(6):1205–1221, 2019. doi: 10.1109/JSAC.2019.2904348.

[28] Dongdong Ye, Rong Yu, Miao Pan, and Zhu Han. Federated learning in vehicular edge computing: A selective model aggregation approach. *IEEE Access*, 8:23920–23935, 2020. doi: 10.1109/ACCESS.2020.2968399.

[29] Romano Fantacci and Benedetta Picano. Federated learning framework for mobile edge computing networks. *CAAI Trans. Intell. Technol.*, 5(1): 15–21, 2020. doi: 10.1049/trit.2019.0049.

[30] Afaf Taïk and Soumaya Cherkaoui. Electrical load forecasting using edge computing and federated learning. In *IEEE ICC, 2020*, pages 1–6. IEEE, 2020. doi: 10.1109/ICC40277.2020.9148937.

[31] Dawei Chen, Jiang (Linda) Xie, BaekGyu Kim, Li Wang, Choong Seon Hong, Li-Chun Wang, and Zhu Han. Federated learning based mobile edge computing for augmented reality applications. In *ICNC 2020, USA, 2020*, pages 767–773. IEEE, 2020. doi: 10.1109/ICNC47757.2020.9049708.

[32] Haifeng Sun, Shiqi Li, F. Richard Yu, Qi Qi, Jingyu Wang, and Jianxin Liao. Toward communication-efficient federated learning in the internet of things with edge computing. *IEEE Internet Things J.*, 7(11):11053–11067, 2020. doi: 10.1109/JIOT.2020.2994596.

[33] Jiale Zhang, Yanchao Zhao, Junyu Wang, and Bing Chen. Fedmec: Improving efficiency of differentially private federated learning via mobile edge computing. *Mob. Networks Appl.*, 25(6):2421–2433, 2020. doi: 10.1007/s11036-020-01586-4.

[34] Wentai Wu, Ligang He, Weiwei Lin, and Rui Mao. Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems. *IEEE Trans. Parallel Distributed Syst.*, 32(7):1539–1551, 2021. doi: 10.1109/TPDS.2020.3040867.

[35] Rong Yu and Peichun Li. Toward resource-efficient federated learning in mobile edge computing. *Netw.*, 2021. doi: 10.1109/MNET.011.2000295.

[36] Chenyuan Feng, Zhongyuan Zhao, Yidong Wang, Tony Quek, and Mugen Peng. On the design of federated learning in the mobile edge computing systems. *Trans. Commun.*, 2021. doi: 10.1109/TCOMM.2021.3087125.

[37] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of the 20th International Conf. on AI and Statistics*, volume 54, pages 1273–1282. PMLR, 2017.

[38] TensorFlow. TensorFlow, 2022. URL https://www.tensorflow.org/.

[39] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognit. Lett.*, 27(8):861–874, 2006. doi: 10.1016/j.patrec.2005.10.010.

[40] Pedro Pereira, Paulo Cortez, and Rui Mendes. Multi-objective grammatical evolution of decision trees for mobile marketing user conversion prediction. *Expert Syst. Appl.*, 2021. doi: 10.1016/j.eswa.2020.114287.

[41] Shin-Yuan Hung, David C. Yen, and Hsiu-Yu Wang. Applying data mining to telecom churn management. *Expert Syst. Appl.*, 31(3):515–524, 2006. doi: 10.1016/j.eswa.2005.09.080.

[42] Chris Richardson. *Microservices Patterns.* Manning, 2019.