Re-balancing the non-IID IoT Edge Devices Data in Federated Learning System

RANEEN YOUNIS1, MARCO FISICHELLA1
1L3S Research Center, Leibniz University of Hannover, Appelstr. 9a, 30167 Hannover, Germany (e-mail: {younis, mfisichella}@L3S.de)
Corresponding author: Raneen Younis (e-mail: younis@l3s.de).

ABSTRACT In recent years, the data available from IoT devices have increased rapidly. Using a machine learning solution to detect faults in these devices requires the release of device data to a central server. However, these data typically contain sensitive information, leading to the need for privacy-preserving distributed machine learning solutions, such as federated learning, where a model is trained locally on the edge device, and only the trained model weights are shared with a central server. Device failure data are typically imbalanced, i.e., the number of failures is minimal compared to the number of normal samples. Therefore, re-balancing techniques are needed to improve the performance of a machine learning model. In this paper, we present FL-M-SMOTE, a new approach to re-balance the data in different non-IID scenarios by generating synthetic data for the minority class in supervised learning tasks using a modified SMOTE method. Our approach takes \( k \) samples from the minority class and generates \( M \) new synthetic samples based on one of the nearest neighbors of each \( k \) sample. An experimental campaign on a real IoT dataset and three well-known public datasets show that the proposed solution improves the balance accuracy without compromising the model’s accuracy.

INDEX TERMS Federated learning, Imbalanced Data, IoT, Non-IID Data.

I. INTRODUCTION

Internet of Things (IoT) is now widely available and uses sensors to collect data for various applications such as healthcare, smart grids, and many more. These sensor data are used as input to artificial intelligence (AI) or Big Data models to predict or classify helpful information. One of the use cases for AI models in IoT is device failure detection, where the client devices need to transmit their sensor data to a central server to detect the failure. Using a traditional solution such as a central Machine Learning (ML) infrastructure to share these data is not optimal because it has several limitations, such as limited communication bandwidth. Moreover, sharing user data on servers could compromise their confidentiality and cause privacy issues, as they could contain sensitive information such as pictures, gender, salary, health status, and so on. To address this issue, various approaches to on-device inference have been introduced in recent years. Federated Learning (FL) [1] was introduced in 2017 by researchers at Google to train a ML model on the user’s device without compromising their privacy by sharing only the trained model weights. FL allows many users to create a shared ML model without revealing their private training data. In each round, the Federated Averaging Algorithm (FedAvg) averages the trained weights from each participating Edge device. In each round, participating clients train a local model based on their local data and then send the trained weights back to the central server. The central server aggregates all the weights from the participating clients and updates the global model. In general, the performance of a classification task in the FL setup depends on the training dataset, which should be independently and identically distributed (IID) and balanced over the edge devices. However, in many cases, such as device failure or anomaly detection tasks, the number of anomaly samples is much smaller than the rest of the training data, which means that the training data are imbalanced in one or more of the client. The central ML model can assume that the training data are balanced and IID since it is collected from all clients and trained with a central model. However, in FL, this assumption is probably not true because each client trains the model on its local data, which may be IID different from other data. FedAvg authors [2] claim that the algorithm is capable of being adapted to some extent with non-IID data.
However, other research [3] shows that the accuracy of the FL model usually drops when it is trained on non-IID data. The reason for the decrease in accuracy of the FL model is due to the convergence of the weights of the different local models from non-IID data. Therefore, the difference between the averaged weights obtained from the participating clients (convergence from non-IID data) and the joint model keeps increasing at the beginning of each round and degrades the performance of the FL model [4]. Our work introduces a new approach to deal with the imbalanced data of the FL clients, where the proportion of minority class samples is smaller than the total number of data samples. This small fraction of the minority class affects the classification accuracy. We generate syntactic samples for minority classes by using the state-of-the-art re-balancing method SMOTE and modifying the algorithm when selecting the number of samples. Figure 1 shows our proposed framework for generating synthetic samples for minority class labels. In each global round, each participating client evaluates whether its data is imbalanced. If so, clients generate new samples in the minority class, train their local model, and pass the weights to the central server, which aggregates the model weights of all participants. Our proposed approach improves the balance accuracy for all the imbalance data tested, as each client node checks if its data is imbalanced to apply the re-balancing technique to its minority class. The main contributions of our work are as follows:

- We modify the SMOTE re-balancing technique by generating synthetic points based on $k$ randomly selected samples instead of all minority class samples.
- From each randomly selected sample $k$, $M$ new samples are generated based on one of the $k$ nearest neighbors.
- We verify the performance of the proposed approach through extensive experiments on IoT device failure detection hotels dataset [5]. Also, we tested our approach on well-known public datasets: Bank [6], Compass [7], and Adult census income [6].

The rest of the paper is organized as follows: In Section II, we discuss the related work in the context of federated learning and the different re-balancing techniques. Section III introduces the notions and the definition of FL. Section IV discusses our proposed approach to re-balancing data in FL setup. In section V, we evaluate and discuss the results of applying our approach in different datasets. Finally, section VII concludes the paper and provides directions for future work.

II. RELATED WORK

In this section we present the related work in two parts. First, we discuss the earlier work that presents various solutions to the non-IID data in the FL setup. Then, we briefly review the state-of-the-art re-balancing techniques that are now widely available.

A. FEDERATED LEARNING AND NON-IID

Federated learning is now widely used as a decentralized solution to train data on edge devices and share only the trained weight [8, 9]. Instead of sharing a large amount of data from each edge device to the central server, federated learning distributes the learning process to the edge devices (clients) by propagating a global model to each participating client. The clients then apply stochastic gradient descent (SGD) to their local data and share the learned gradients with the server. The server averages all updated gradients from the clients and updates the initial model. Several research papers such as [10, 11, 12] address the FL challenge of how to learn a global model on non-IID training data. The non-IID data is challenging in the FL setup because of the different data distribution between clients. This causes each client’s local model to converge to its local optimum, which is different from the global optimum, decreasing the accuracy of the FedAvg model [13]. Karimireddy et al. introduce SCAFFOLD [10], which improves the drifts of the model’s local training updates by adding the difference between the update direction of the server model and the update direction of each client. Using the SCAFFOLD method doubled the communication size for each round compared to the FedAvg algorithm. FedNova [11] scales and normalizes each client’s local updates based on its local step count before updating the global model. FedNova provides lightweight modifications to FedAvg and modest computational costs when updating.
the global model. FedProx [12] adds regularization terms to the loss function to reduce the distance between the global and local models when the clients’ data are non-IID, and this makes the averaging of the local models not far from the global optima. Wand et al. [14] proposed a decentralized framework for re-balance the local data on each participating client using P2PK-SMOTE. The presented P2PK-KSMOTE method artificially generates synthetic points for the minority class based on random k points. Further research introduces the non-IID in FL settings such as [15, 16] where they adjust the local model of each client. Zhang et al. [5] present the CDW_FedAvg algorithm to overcome the challenges of heterogeneous data of the IoT devices in detecting device failures. Their algorithm considers the distance between the positive and negative classes of each client dataset when updating the global model weights to reduce the impact of data heterogeneity. Duan et al. [17] introduce a self-balancing FL framework, Astraea, for class imbalance problems using Z-score-based data augmentation, and a down sample of the client’s local data. A federated anomaly detection solution was proposed by [18] to detect the compromised IoT device by aggregating the behavior profiles. An autoencoder approach for anomaly detection using the updated weights of the clients in the server was presented by [19].

B. BALANCING DATA

The class imbalance problem is common in classification tasks where one output class is considered a minority class, meaning that this output class has fewer samples than the other output class in a binary classification dataset. This problem affects the performance of the model and causes it to favor the majority class and generally miss classify the minority class. Anomaly detection in IoT devices, such as device failure detection, is an example of a class imbalance problem. In such an example, most of the data consist of benign samples, and only a small portion of the collected data is a device failure. Training a machine learning model on imbalanced data usually does not give good results. Therefore, several methods have been introduced to re-balance the data, such as oversampling [20] and under-sampling [21]. The under-sampling method in [21] removes points from the majority class based on the distance of the removed sample and other points in the same class. The oversampling methods generate random samples from the minority classes. SMOTE [20] is a popular method used to generate synthetic samples from the minority class. SMOTE takes the k nearest neighbors of a data point and then multiplies the vector between a neighbor and the data sample by a random number between 0 and 1.

III. PRELIMINARIES

Let us define the central server as $S$ and the participating clients in the FL rounds as $\{C_1, C_2, \ldots, C_n\}$. The local dataset for each client is defined as $D_{ci}$. The global model is defined as $w^g_t$, and the local model for each client is defined as $w^l_{ci}$, where $w^t$ are the model weights that will be shared at round $t$.

A. FEDERATED LEARNING

In federated learning tasks, the server $S$ first initiates its global model $w_g$ and sends it to a random number of clients $\{c_n\}$. The selected clients train their local models using their local datasets and the global model, and then send the updated local model to the central server $S$ in each round. The server updates the weights of the global model using the FedAvg [2] algorithm as follows:

$$w^{t+1} = w^t - \eta \sum_{i \in \{c_n\}} \frac{|D^i|}{n} \Delta w^i_t,$$

where $w^{t+1}$ is the global updated weights, $w^t$ is the global weights at round $t$, $\eta$ is the learning rate, $n$ is the total number of training samples, $|D^i|$ is the number of local training samples on client $i$, and $w^i_t$ is the local weights of client $i$.

B. FEDERATED LEARNING AND NON-IID

Non-IID is one of the challenges in FL tasks. Different data distribution between clients will affect FedAvg accuracy, since the local model target of each client is different from the target of the global optimum. Therefore, there will be a deviation in the local updates. The local models will be updated towards the local optima that are away from the global optima. The averaging of these local updates will deviate from the global optima, so the accuracy of the global model will be worse than the IID setting [13].

C. NON-IID SCENARIOS

In the IoT network of devices, there are a number of IoT devices connected to a central server or node that collects data from these devices to perform various operations. The setup of FL protects the privacy of the shared data as the IoT devices share only the learned weights. However, sharing only the trained weights of each node poses some challenges when the data are imbalanced between the devices. There are several scenarios for training a binary classification network to detect anomalies (the number of benign data is greater than malicious data) in edge devices.

1) Balanced dataset scenario: In this scenario, the dataset in each client or edge device is balanced and IID. The $|D_{\text{min}}|$ is equal to $|D_{\text{maj}}|$ where $\text{min}$ stands for minority and $\text{maj}$ for majority. The global dataset (the aggregated data of all the clients) is also balanced.

2) Locally imbalanced dataset scenario: The dataset in each client or edge device is imbalanced. The $|D_{\text{min}}|$ is smaller than $|D_{\text{maj}}|$.

3) Mixed imbalanced dataset scenario: The ratio between the min class to maj class is different in each client. For example, some clients could have $|D_{\text{min}}|$ larger than $|D_{\text{maj}}|$, where other clients could have $|D_{\text{min}}| \approx |D_{\text{maj}}|$.
IV. APPROACH

In this section, we describe our approach for re-balancing the data in the FL setup. Figure 2 shows the process flow of the federated learning framework, including the re-balance step for the client data. The upper block contains the central server node that aggregates the weights of the clients connected to it. The bottom block contains an example of a client node participating in the learning process. The following subsection describes the system architecture and the re-balancing process in more detail.

A. SYSTEM ARCHITECTURE

Figure 2 shows our system design for re-balancing clients data in a FL setup. Our system consists of three steps: 1) sharing the global model, 2) re-balancing the clients’ local data, and 3) sharing the locally trained model. The upper block describes the central server node, which contains the global model that is shared with selected participating clients. The central server node shares the global model with the selected participating clients in each round. The client takes this global model and trains it based on its local data. As shown in the bottom block of Figure 2, the participating client tests its dataset to determine whether it is balanced or not. A client considers its data as imbalanced if the ratio between $D_{\text{min}}:D_{\text{maj}}$ is less than the threshold value $\tau$ or if the value of the balance accuracy from three previous communication rounds is still improving. If the data are imbalanced, the client applies the re-balancing step using an oversampling technique. This technique creates new synthetic data from the minority class; this step is described in Section IV-B. Then, the participating clients train their local model using the original data combined with the newly generated synthetic data from the minority class, which balances the client’s local dataset. The client then sends its updated trained model to the server node. The server aggregates the models of the participating clients into its global model using the FedAvg algorithm.

B. FL-M-SMOTE RE-BALANCING METHOD

SMOTE [20] is the state-of-the-art method to re-balance the data. It uses the minority class samples to generate new synthetic data for that class by linear interpolation. The SMOTE method is inspired by [22], where a handwritten character recognition technique was proposed that generates new training data from the original data by applying specific operations such as rotation and skewness to the real data. The SMOTE method generates new synthetic data by working in feature space instead of data space. SMOTE uses the $k$-nearest neighbors of all minority class data samples as follows:

$$\forall x_i \in D_{\text{min}}: \text{return } \{KNN\}$$

Then the SMOTE method randomly selects one of the neighbors to perform the linear interpolation. The newly generated samples from SMOTE are as follows:

$$x_{new} = x_i + (x_{ij} - x_i) \times R[0, 1],$$

where $x_{new}$ is the newly generated synthetic sample added to the $|D_{\text{min}}|$ training data, $x_i$ is a data sample in $|D_{\text{min}}|$, and $x_{ij}$ is one of the $\{KNN\}$ of $x_i$. $R[0, 1]$ is a random number between 0 and 1.

A modified version of SMOTE is the kSMOTE method proposed by [14]. kSMOTE takes only $k$ number of samples in the minority class instead of all data samples to reduce computational complexity (Algorithm 1 line 9-13). Also, $k$ nearest neighbors are selected instead of only one neighbor as in SMOTE. In our work, we randomly select one of the $k$ nearest neighbors, similar to the original SMOTE method.

Since we use the modified SMOTE method by taking $k$ samples from the minority class and generating $M$ new samples from each of them, we refer to this method as the M-SMOTE method. In this paper, we use the M-SMOTE method to re-balance the data of FL clients, so we call our proposed approach FL-M-SMOTE method.

Algorithm 1 explains the process of re-balancing the data and training a local model in a client node. The algorithm takes the server’s global model as input and returns the client’s trained model as output. At the beginning, the algorithm initializes the parameters (lines 1-7) required for the client training and the re-balancing process. Then, the client tests whether its data are balanced or not by checking the ratio of $D_{\text{min}}:D_{\text{maj}}$ and the previous three communication rounds’ balance accuracy (line 8). If the client data are imbalanced (i.e., its ratio value is less than the threshold $\tau$, or the balance accuracy from the previous communication rounds is still improving), the client goes through the re-balancing process (lines 9-17). The process randomly selects $k$ samples from $D_{\text{min}}$ (line 10), and for each selected sample $x_i$, its $k$ nearest neighbors are collected (line 12). From the $k$ nearest neighbors, one data point $x_{ij}$ is randomly selected to generate the new synthetic sample (line 13). A loop is
In this work, we use the hotel IoT dataset [5], and three well-known public datasets: Bank [6], Compass [7], and Adult census income [6]. We will discuss in detail each of the datasets used and the architecture of the deep neural network trained on the dataset, and how we partitioned the datasets for use in a FL setup.

1) **Hotels Dataset** is a real-world dataset provided by [5] to detect the failure of air conditions in hotels. This dataset was collected from four hotels (the four clients used in the FL approach). The data collected from the sensors have 70 features before being pre-processed. The data are then processed by removing the noisy data and redundant features and contains 17 features that are trained by a machine learning model. First, we run the experiments on the original data to test our approach when the data are balanced. Then, we re-process the data to change the ratio between the positive and negative labels. We split the positive labels so that the ratio between positive and negative labels is as follows: 1 : 4, 1 : 10, 1 : 20, 1 : 30. We use a fully connected layers neural network with four layers, the first input layer consists of 17 neurons, the following two hidden layers consist of 85 neurons and a ReLu activation function. The output layer has a sigmoid activation function to classify the output labels.

2) **Bank Dataset** is related to direct marketing campaigns of a Portuguese banking institution. The goal of the dataset is to decide whether a person subscribes to the product. The dataset contains 46 features with an imbalanced ratio of positive to negative of 1 : 8. To use this dataset in FL setup, we split the dataset among a number of clients in two ways. The first way is to randomly split the data across multiple clients, providing the same number of samples for each client and ensuring that each client has a similar ratio of positive to negative labels. The second way is to split the dataset based on one of its features. We divide the dataset into three categories based on the age feature. So, the first client has the values where the age is less than 30 years, the second has an age between 30 and 40 years, and the last client has the records where the age is more than 40 years.

3) **Adult census income Dataset** is used to predict whether a U.S. person’s annual income will exceed 50 thousand dollars. The dataset has 14 features with a positive to negative ratio of 1 : 3. The positive class is individuals who earn more than 50 thousand. To use this dataset in the FL setup, we use a similar approach to the bank dataset to distribute the dataset between clients.

4) **Compass Dataset** contains information on prisoners in Broward County. The purpose of this dataset is to determine whether an individual will be re-arrested within two years. This dataset contains 28 features with a balance of positive and negative ratio. To test this dataset with our approach, we first need to split the data so that the ratio of positive to negative class labels is imbalanced. To do this, we take only the female prisoner records, the ratio of positive to negative becomes 1 : 2. To test the dataset on a FL setup, we randomly split the data into multiple clients, similar to the bank and adult datasets.

**Algorithm 1** FL client node generating new samples for the minority class.

**Require:** The Global model from the central server

**Ensure:** The Client model trained on its local dataset

1: Model := Global model
2: \(X_{\text{train}}\) := Client training dataset
3: \(k\) := Integer number of samples from \(D_{\text{min}}\)
4: \(r\) := The ratio of new samples to be created based on \(D_{\text{min}}\)
5: \(R = \text{int}((D_{\text{maj}} - D_{\text{min}}) \times r)\)
6: \(M = \text{int}(R/k)\) \(\triangleright\) Number of new samples to be generated from each of the random selected \(k\) samples from \(D_{\text{min}}\)
7: LocalModel \(\leftarrow\) Model
8: if \(X_{\text{train}}\) is not balanced then
9: \(X_{\text{new}}\) = []
10: \(X_{\text{random}}\) = \{Random \(k\) samples from \(D_{\text{min}}\)\}
11: for \(x_i\) in \(X_{\text{random}}\) do
12: \(x_{\text{nearest}}\) = \{Nearest neighbors for \(x_i\)\}
13: \(x_{ij}\) = A randomly selected sample from \(X_{\text{nearest}}\)
14: for \(m\) in \{1,2,.....M\} do
15: \(x_{new} = x_i + (x_i - x_{ij}) \times R[0,1]\)
16: \(X_{\text{new}}\) .append(\(x_{new}\))
17: end for
18: end for
19: \(X_{\text{train}}\) .append(\(X_{\text{new}}\))
20: end if
21: LocalModel \(\leftarrow\) train(Model, \(X_{\text{train}}\))
22: return LocalModel

Then executed to generate \(M\) new samples (lines 14-16). The newly generated samples are then added to the training data (line 17), and the client trains its model (line 18).

**V. IMPLEMENTATION AND EVALUATION**

In this section, we present the results of our approach, which we tested on four datasets. We run the experiments on a core i5 8500 CPU PC with 16GB RAM. For all datasets, we train the models by randomly splitting the data into 80% training and 10% for each validation and test data. For the threshold \(\tau\), we set it to 0.33, which is the largest ratio we found in the datasets (i.e., the Compass dataset). We report the detailed results for all datasets.

**A. DATASETS AND NETWORK MODEL**

In this work, we use the hotel IoT dataset [5], and three well-known public datasets: Bank [6], Compass [7], and Adult census income [6]. We will discuss in detail each of the datasets used and the architecture of the deep neural network trained on the dataset, and how we partitioned the datasets for use in a FL setup.

1) **Hotels Dataset** is a real-world dataset provided by [5] to detect the failure of air conditions in hotels. This dataset was collected from four hotels (the four clients used in the FL approach). The data collected from the sensors have 70 features before being pre-processed. The data are then processed by removing the noisy data and redundant features and contains 17 features that are trained by a machine learning model. First, we run the experiments on the original data to test our approach when the data are balanced. Then, we re-process the data to change the ratio between the positive and negative labels. We split the positive labels so that the ratio between positive and negative labels is as follows: 1 : 4, 1 : 10, 1 : 20, 1 : 30. We use a fully connected layers neural network with four layers, the first input layer consists of 17 neurons, the following two hidden layers consist of 85 neurons and a ReLu activation function. The output layer has a sigmoid activation function to classify the output labels.

2) **Bank Dataset** is related to direct marketing campaigns of a Portuguese banking institution. The goal of the dataset is to decide whether a person subscribes to the product. The dataset contains 46 features with an imbalanced ratio of positive to negative of 1 : 8. To use this dataset in FL setup, we split the dataset among a number of clients in two ways. The first way is to randomly split the data across multiple clients, providing the same number of samples for each client and ensuring that each client has a similar ratio of positive to negative labels. The second way is to split the dataset based on one of its features. We divide the dataset into three categories based on the age feature. So, the first client has the values where the age is less than 30 years, the second has an age between 30 and 40 years, and the last client has the records where the age is more than 40 years.

3) **Adult census income Dataset** is used to predict whether a U.S. person’s annual income will exceed 50 thousand dollars. The dataset has 14 features with a positive to negative ratio of 1 : 3. The positive class is individuals who earn more than 50 thousand. To use this dataset in the FL setup, we use a similar approach to the bank dataset to distribute the dataset between clients.

4) **Compass Dataset** contains information on prisoners in Broward County. The purpose of this dataset is to determine whether an individual will be re-arrested within two years. This dataset contains 28 features with a balance of positive and negative ratio. To test this dataset with our approach, we first need to split the data so that the ratio of positive to negative class labels is imbalanced. To do this, we take only the female prisoner records, the ratio of positive to negative becomes 1 : 2. To test the dataset on a FL setup, we randomly split the data into multiple clients, similar to the bank and adult datasets.
**B. EVALUATION METRICS**

The most common evaluation metric for a classification task is accuracy, which is the number of correctly predicted data samples out of all data samples. This method can be misleading for problems with imbalanced data, where the minority data points are fewer than the majority data points, so the model may give good accuracy but be biased toward the majority class. To address this issue, we use several evaluation metrics such as sensitivity, specificity, balance accuracy, G-mean, Matthews correlation coefficient (MCC) [23], False Negative Rate (FNR), and False Positive Rate (FPR) to measure model performance. The formal definitions of these metrics are as follows:

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \tag{4}
\]

\[
\text{Specificity} = \frac{TN}{FP + TN} \tag{5}
\]

\[
\text{Balance Accuracy} = \frac{sensitivity + specificity}{2} \tag{6}
\]

\[
G - mean = \sqrt{\text{sensitivity} \cdot \text{specificity}} \tag{7}
\]

\[
MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{P \cdot (TP + FN) \cdot (TN + FP) \cdot N}} \tag{8}
\]

\[
FNR = \frac{FN}{FN + TP} \tag{9}
\]

\[
FPR = \frac{FP}{FP + TN} \tag{10}
\]

**C. BASELINE TECHNIQUES**

This section introduces the different baseline techniques we compare our FL-M-SMOTE method with.

- **FedAvg [1]**: The first method is the original method of federated learning, where the trained weights of the clients are averaged in the central server. In this method, we do not re-balance the clients’ data. We only report the results of the FL system without using any IID or re-balancing methods.

- **FL-SMOTE [20]**: Since we modified the most popular re-balancing method SMOTE, we compare our approach with it in the FL setup. We re-balance the clients’ data using the SMOTE method, so we call this approach FL-SMOTE.

- **FedNova [11]**: FedNova is introduced to handle the non-IID data in the FL setup. The method modifies FedAvg by normalizing the local updates of each participant based on its local step count before aggregating the updated weights of the local models to the global model.

**D. QUANTITATIVE ANALYSIS**

In this section, we evaluate our approach using the Hotels, Bank, Adult, and Compass datasets. We also consider the different scenarios defined in Section III-C.
1) Hotels Dataset

is balanced for each hotel in the dataset. We test this dataset in two case scenarios as follows:

- **Balanced dataset scenario:** where each client of the hotels dataset is balanced, i.e., there is neither a minority nor a majority class. In the first experiment, we test the performance of the FedAvg, FL-SMOTE and our FL-M-SMOTE methods. In each global round, the client is trained for three epochs (local round) with its local data using a fully connected neural network. Then, the client passes its trained model weights to the central node, which averages the shared weights and updates the global model. We perform the first experiment with 15 global rounds. In the method FL-M-SMOTE, we test different values for the two hyperparameters $k$ and $r$, where $k$ is the number of randomly selected $k$ samples from $D_{\text{min}}$ that are responsible for generating new samples, and $r$ is the ratio of new samples to be generated based on $D_{\text{min}}$. We reported the values that give the best balance accuracy. For FL-SMOTE, we set the nearest neighbor parameter to five. We evaluate our approach using the balanced data with the metric for balancing accuracy. We compare our method FL-M-SMOTE with the two baselines, FedAvg and FL-SMOTE. Figure 3 shows the balancing accuracy of the test datasets for each client and the aggregated test data run for 15 global rounds, as well as the distribution of the clients data (lower right sub-figure). The blue line shows the results of the method FL-M-SMOTE with $k = 3$ and $r = 0.4$. As can be seen in the figure, our proposed approach (blue line) slightly improves the balance accuracy compared to the FedAvg approach for all client datasets and the aggregated data. Moreover, the FL-M-SMOTE method shows better balance accuracy than the FL-SMOTE method for all the client datasets and the aggregated data. The figure also shows that FL-SMOTE performs better than FedAvg only for two dataset (Dataset1 and Dataset2). From this experiment, we can conclude that our proposed method FL-M-SMOTE works well in the first scenario and does not affect the performance of the balance accuracy in any of the clients or the aggregated data, and in some cases even improves the balance accuracy.

- **Locally imbalanced dataset scenario:** where the dataset for each client is imbalanced. Since the original hotel dataset is balanced, we modified the dataset to obtain unbalanced data for each customer to better mimic a scenario where component failures are lower than their normal activity. For this purpose, we split the hotels dataset with different re-sample ratios as explained in Section V-A. To evaluate the performance of our approach in this scenario, we compare the method FL-M-SMOTE with the two baseline approaches, FedAvg and FL-SMOTE, in the different split ratios of the hotels dataset. The split ratio indicates the ratio between the minority and majority class samples. Table 1 shows the evaluation metrics of the aggregated hotels dataset. The table shows the results of the original data, which is balanced, and the four different re-sampled data with the different ratios of minority to majority classes. With the original balanced data, the FL-M-SMOTE approach improves the balance accuracy, $G_{\text{mean}}$ and accuracy by 1% compared to the baseline FedAvg method. However, the FL-SMOTE baseline method degrades the FedAvg algorithm by 9% for the three metrics. The table also shows that the proposed method FL-M-SMOTE significantly improves the FedAvg accuracy, $G_{\text{mean}}$, and balance accuracy for all the imbalanced data with the different ratios; for example, the balance accuracy improves by about 20% in the case where the minority to major ratio is 1:30. Since we choose the positive label as the minority class and the FL-M-SMOTE generates new synthetic samples with positive label, the sensitivity also improves dramatically, while we observe a slight but expected deterioration in specificity, as shown in the table. In summary, our approach FL-M-SMOTE improves the balance accuracy, $G_{\text{mean}}$, and FedAvg model accuracy for both balanced and imbalanced data. In addition, our method outperforms the FL-SMOTE baseline for the original data and all ratios tested. We conclude that our approach FL-M-SMOTE achieved the intended goal of improving the performance of the model for imbalanced data. It is also shown that it does not degrade the performance for balanced data, but maintains or slightly improves the performance of the balancing accuracy.

2) Bank Dataset

is imbalanced where the ratio of positive to negative class being 1:8. To evaluate the dataset in the FL setup, the dataset was split into three clients both randomly and based on the age feature to have one case scenario as follows:

- **Locally imbalanced dataset scenario:** where each of the clients has imbalanced data. To evaluate our ap-
TABLE 1. The table shows the values of the evaluation metrics for the methods FedAvg, FL-SMOTE, and FL-M-SMOTE, which were tested with the original aggregated hotels dataset, and when it was re-sampled at a ratio of 1:4, 1:10, 1:20, and 1:30.

| Split       | Model      | Sensitivity | Specificity | Balance/ACC | MCC | G_mean | FN_rate | FP_rate | Accuracy |
|-------------|------------|-------------|-------------|-------------|-----|--------|---------|---------|----------|
| Original    | FedAvg     | 0.86        | 0.84        | 0.85        | 0.70| 0.85   | 0.15    | 0.14    | 0.85     |
|             | FL-SMOTE   | 0.83        | 0.69        | 0.76        | 0.69| 0.76   | 0.16    | 0.76    | 0.16     |
|             | FL-M-SMOTE | 0.86        | 0.86        | 0.86        | 0.73| 0.86   | 0.14    | 0.14    | 0.86     |
| 1:4         | FedAvg     | 0.58        | 0.96        | 0.77        | 0.70| 0.75   | 0.04    | 0.42    | 0.78     |
|             | FL-SMOTE   | 0.64        | 0.96        | 0.80        | 0.69| 0.78   | 0.03    | 0.35    | 0.80     |
|             | FL-M-SMOTE | 0.75        | 0.88        | **0.81**    | 0.70| 0.82   | 0.11    | 0.24    | **0.82** |
| 1:10        | FedAvg     | 0.41        | 0.98        | 0.67        | 0.72| 0.63   | 0.01    | 0.58    | 0.69     |
|             | FL-SMOTE   | 0.43        | 0.98        | 0.71        | 0.71| 0.65   | 0.01    | 0.56    | 0.71     |
|             | FL-M-SMOTE | 0.70        | 0.90        | **0.80**    | 0.73| 0.79   | 0.09    | 0.23    | **0.80** |
| 1:20        | FedAvg     | 0.26        | 0.99        | 0.63        | 0.71| 0.51   | 0.02    | 0.73    | 0.63     |
|             | FL-SMOTE   | 0.45        | 0.99        | 0.72        | 0.72| 0.67   | 0.00    | 0.54    | 0.73     |
|             | FL-M-SMOTE | 0.61        | 0.93        | **0.77**    | 0.73| 0.75   | 0.07    | 0.39    | **0.80** |
| 1:30        | FedAvg     | 0.17        | 0.99        | 0.59        | 0.71| 0.42   | 0.01    | 0.82    | 0.58     |
|             | FL-SMOTE   | 0.25        | 0.99        | 0.62        | 0.71| 0.50   | 0.00    | 0.74    | 0.63     |
|             | FL-M-SMOTE | 0.67        | 0.91        | **0.79**    | 0.72| 0.78   | 0.08    | 0.32    | **0.79** |

FIGURE 5. The evaluation metrics values for the Bank dataset tested with FedAvg and FL-M-SMOTE on 3, 5, 10, and 15 randomly distributed clients, and on 3 clients distributed based on age value. Note that Rn/C means randomly splitting the data for n clients.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
Similar to the Bank dataset, we evaluate the Adult dataset using the two split methods and compare the performance of our method FL-M-SMOTE with the FedAvg baseline method. Figure 6 shows the balance accuracy results when the experiment is run both for 30 global rounds on three randomly distributed clients and based on the age feature of the dataset. The figure shows that FL-M-SMOTE increases the balance accuracy of FedAvg by 2% when the dataset is divided among the three clients both randomly and based on the age feature. Since the balance accuracy improved in the age split case, we can conclude that our approach improves the performance of the model in the scenario where the clients have both balanced and imbalanced data.

The bar chart in Figure 7 shows the results of the evaluation metrics for the Adult dataset when the dataset is randomly split into 3, 5, 10, and 15 clients. The results show the average value of 5 repetitions for each partitioning of the dataset. The plot shows that FL-M-SMOTE improves the balance accuracy, G-mean, MCC, and sensitivity in all cases. It also shows that the accuracy is slightly improved in the case of random splitting 5, 10, and 15 and age splitting and remains unchanged in the case of random splitting of the data for three clients. Moreover, the specificity decreases slightly in the case of random splitting 3 and 15 and remains unchanged for the other splits.

4) Compass Dataset

is balanced and has an equal ratio of positive and negative values. To test this dataset with our approach, we split the data so that the ratio of positive and negative class labels is imbalanced, leading to the second scenario Locally imbalanced dataset scenario. We evaluate the dataset based on the random distribution of the data among the clients. Figure 8 shows the result of the experiment with the Compass dataset when randomly split among three clients. The result shown is the average result of 5 runs of the experiment. We can see that the FL-M-SMOTE approach improves the balance accuracy of the dataset by about 2% after 70 rounds. We also note that the FL-M-SMOTE curve fluctuates less and is more stable. We also test the dataset with 3, 5, and 10 randomly split clients and report the evaluation metrics for the FL-M-SMOTE and FedAvg approaches in Figure 9. The bar chart shows that the balance accuracy, G-mean, MCC, and sensitivity are improved in all cases of splitting. In addition, the accuracy of the model improves with the FL-M-SMOTE approach when splitting the data for 5 and 10 clients and remains unchanged when splitting the data for three clients. The specificity of the model decreases slightly in all cases, and this is due to adding more samples to the positive class label.

**E. COMPARISON OF FL-M-SMOTE WITH BASELINE TECHNIQUES**

In this section, we compare the FL-M-SMOTE with the baseline methods FedNova, and FedAvg. For this purpose, we perform experiments on the four datasets with the different splits. Table 2 shows the top-1 results of the balance accuracy for the methods FL-M-SMOTE, FedAvg and FedNova. The table shows that for the Hotels dataset, the FL-M-SMOTE balance accuracy, highlighted in bold, improved in all tested cases (the original balanced data and when the data are split to have different ratios between positive and negative classes). However, for the FedNova method, the balance accuracy decreases by 9% for a split ratio of 1:30 and improves by about 2% only for two splits 1:10 and 1:20. We tested the three methods for the Bank and Adult datasets both when the data were split randomly and based on age feature. The table shows that the FL-M-SMOTE improves the balance accuracy in all cases for both datasets, while FedNova does not improve the balance accuracy in any cases but lowers the value slightly in some cases. For the Compass dataset, the FL-M-SMOTE balance accuracy improved by 2%, while it decreased by 4% when using FedNova.

In summary, the experiments show that our proposed method FL-M-SMOTE improves the performance of balanced accuracy on imbalanced data in different tested scenarios for all tested datasets. Also, our approach does not degrade the accuracy performance but improves it in some splitting scenarios. Moreover, the proposed approach outperforms the baseline methods in all datasets.

| Dataset | Split | FedAvg | FedNova | FL-M-SMOTE |
|---------|-------|--------|---------|------------|
| Hotels  | No    | 0.85   | 0.85    | 0.86       |
|         | 1:04  | 0.77   | 0.77    | 0.81       |
|         | 1:10  | 0.67   | 0.69    | 0.80       |
|         | 1:20  | 0.63   | 0.66    | 0.77       |
|         | 1:30  | 0.59   | 0.50    | 0.79       |
| Bank    | Random | 0.74 | 0.73     | 0.83       |
|         | Age   | 0.72   | 0.72    | 0.78       |
| Adult   | Random | 0.76 | 0.75     | 0.78       |
|         | Age   | 0.70   | 0.71    | 0.72       |
| Compass | Random | 0.76 | 0.72     | 0.78       |

**TABLE 2.** The top-1 Balance accuracy of the different federated learning approaches runs on the four datasets with different splits between 3 clients. The best performance is highlighted in bold.
The FL-M-SMOTE method has two primary hyperparameters: $k$ and $r$. In this section, we present sensitivity analysis for the two parameters and show how they affect the balance accuracy result of a model. The results for the Adult dataset are shown in Fig. 10. The experiment shows that the balance accuracy decreases when the value for the parameter $r$ is increased. This is because the $r$ parameter determines the ratio of new samples to be created based on $D_{min}$. Thus, the more samples added to the minority class, the dataset will have more minority class samples than the majority. Thus, the smaller the parameter $r$, the better the balance accuracy, therefore for the adult dataset we set the $r$ value to 0.1. For the $k$ value, Fig. 10 shows that the best parameter value for the adult dataset is $k = 5$. A value smaller or larger than 5 results in lower balance accuracy; therefore, we choose the value of 5 for the Adult dataset.

**REFERENCES**

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” NIPS Workshop on Private Multi-Party Machine Learning, 2016.

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

[3] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” CoRR, 2018.

[4] H. Zhu, J. Xu, S. Liu, and Y. Jin, “Federated learning on non-iid data: A survey,” Neurocomputing, vol. 465, pp. 371–390, 2021.

[5] W. Zhang, Q. Lu, Q. Yu, Z. Li, Y. Liu, S. K. Lo, S. Chen, X. Xu, and L. Zhu, “Blockchain-based federated learning for device failure detection in industrial IoT,” IEEE...
Sensitivity analysis for the two hyperparameters $r$ and $k$ for the Adult dataset.

FIGURE 9. The evaluation metrics values for the Compass dataset tested with FedAvg and FL-M-SMOTE on 3, 5 and 10 randomly distributed clients. Note that $RnC$ means randomly splitting the data for $n$ clients.

FIGURE 10. Sensitivity analysis for the two hyperparameters $r$ and $k$ for the Adult dataset.

[10] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, “Scaffold: Stochastic controlled averaging for on-device federated learning,” in Proceedings of the 37th International Conference on Machine Learning, 2019.

[11] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, “Tackling the objective inconsistency problem in heterogeneous federated optimization,” Advances in Neural Information Processing Systems, 33, 2021.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” Proceedings of Machine Learning and Systems, vol. 2, pp. 429–450, 2020.

[13] Q. Li, Y. Diao, Q. Chen, and B. He, “Federated learning on non-iid data silos: An experimental study,” arXiv preprint arXiv:2102.02079, 2021.

[14] H. Wang, L. Muñoz-González, D. Eklund, and S. Raza, “Non-iid data re-balancing at iot edge with peer-to-peer federated learning for anomaly detection,” in Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2021, pp. 153–163.

[15] C. T. Dinh, N. H. Tran, and T. D. Nguyen, “Personalized federated learning with moreau envelopes,” Advances in Neural Information Processing Systems, 2020.

[16] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, “Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach,” in NeurIPS, 2020.

[17] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, “Self-balancing federated learning with global imbalanced data in mobile systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 1, pp. 59–71, 2020.

[18] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “Diot: A federated self-learning anomaly detection system for iot,” in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019, pp. 756–767.

[19] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, “Abnormal client behavior detection in federated learning,” NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality, pp. 740—750, 2019.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” Journal of artificial intelligence research, vol. 16, pp. 321–357, 2002.

[21] I. Mani and I. Zhang, “knn approach to unbalanced data distributions: a case study involving information extraction,” in Proceedings of workshop on learning from imbalanced datasets, vol. 126. ICML United States, 2003.

[22] T. M. Ha and H. Bunke, “Off-line, handwritten numeral recognition by perturbation method,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 5, pp. 535–539, 1997.

[23] D. Chicco, N. Tötsch, and G. Jurman, “The matthews

Internet of Things Journal, vol. 8, no. 7, pp. 5926–5937, 2020.

[6] M. Lichman et al., “Uci machine learning repository,” 2013.

[7] J. Larson, S. Mattu, L. Kirchner, and J. Angwin, “How we analyzed the compas recidivism algorithm,” ProPublica (5 2016), vol. 9, no. 1, 2016.

[8] S. Abdurahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, “A survey on federated learning: The journey from centralized to distributed on-site learning and beyond,” IEEE Internet of Things Journal, vol. 8, no. 7, pp. 5476–5497, 2021.

[9] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, “Re-alizing the heterogeneity: A self-organized federated learning framework for iot,” IEEE Internet of Things Journal, vol. 8, no. 5, pp. 3088–3098, 2021.

[15] C. T. Dinh, N. H. Tran, and T. D. Nguyen, “Personalized federated learning with moreau envelopes,” Advances in Neural Information Processing Systems, 2020.

[16] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, “Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach.” in NeurIPS, 2020.

[17] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, “Self-balancing federated learning with global imbalanced data in mobile systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 1, pp. 59–71, 2020.

[18] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “Diot: A federated self-learning anomaly detection system for iot,” in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019, pp. 756–767.

[19] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, “Abnormal client behavior detection in federated learning,” NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality, pp. 740—750, 2019.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” Journal of artificial intelligence research, vol. 16, pp. 321–357, 2002.

[21] I. Mani and I. Zhang, “knn approach to unbalanced data distributions: a case study involving information extraction,” in Proceedings of workshop on learning from imbalanced datasets, vol. 126. ICML United States, 2003.

[22] T. M. Ha and H. Bunke, “Off-line, handwritten numeral recognition by perturbation method,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 5, pp. 535–539, 1997.

[23] D. Chicco, N. Tötsch, and G. Jurman, “The matthews
correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation,” BioData mining, vol. 14, no. 1, pp. 1–22, 2021.

RANEEN YOUNIS is a PhD Student in Computer Science at the Leibniz University of Hannover. She received her MSc Degree in Computer Science from the Clausthal University of Technology in August 2019. Her research interests include time-series data analysis, Machine Learning, and data privacy.

MARCO FISICHELLA has 13+ years of experience in Artificial Intelligence both in the industry and in academia. From April 2021, he led the L3S Research Center of Leibniz University of Hannover the research group with a focus on artificial intelligence and intelligent systems. From 2015 till March 2021, he was Head of the Data at Otto Group where he was directing the department in charge of implementing algorithms against online frauds. Marco led various AI and data science projects for corporate clients (e.g. Otto, Vodafone, DriveNow, Bonprix, etc.).