Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

AI-empowered mobile edge computing: inducing balanced federated learning strategy over edge for balanced data and optimized computation cost

Momina Shaheen[1*], Muhammad S. Farooq[1] and Tariq Umer[2]

Abstract

In Mobile Edge Computing, the framework of federated learning can enable collaborative learning models across edge nodes, without necessitating the direct exchange of data from edge nodes. It addresses significant challenges encompassing access rights, privacy, security, and the utilization of heterogeneous data sources over mobile edge computing. Edge devices generate and gather data, across the network, in non-IID (independent and identically distributed) manner leading to potential variations in the number of data samples among these edge networks. A method is proposed to work in federated learning under edge computing setting, which involves AI techniques such as data augmentation and class estimation and balancing during training process with minimized computational overhead. This is accomplished through the implementation of data augmentation techniques to refine data distribution. Additionally, we leveraged class estimation and employed linear regression for client-side model training. This strategic approach yields a reduction in computational costs. To validate the effectiveness of the proposed approach, it is applied to two distinct datasets. One dataset pertains to image data (FashionMNIST), while the other comprises numerical and textual data concerning stocks for predictive analysis of stock values. This approach demonstrates commendable performance across both dataset types and approaching more than 92% of accuracy in the paradigm of federated learning.

Keywords Artificial intelligence, Class estimation, Data imbalance, Distributed machine learning, Federated learning, Edge devices, Mobile edge computing, Privacy preserving algorithms

Introduction

Owing to the inherent privacy concerns associated with edge data, individuals exhibit reluctance towards the prospect of relinquishing their data to centralized data repositories and cloud servers [1, 2]. Analogously, industries encounter the dual challenges of increased computational and communicative cost, along with the looming spectre of privacy breaches, when considering the storage of data in central server infrastructures.

Federated Learning (FL), a widely adopted Artificial Intelligence (AI) technique, offers an effective avenue to protect and secure the confidentiality of data residing within edge nodes [3]. By facilitating the collaborative construction of a cohesive learning model across diverse edge nodes, FL eliminates the need for direct exchange of data samples under the scenario of Mobile Edge Computing (MEC). This paradigm effectively delivers to a range of critical concerns, including access authorization,

*Correspondence:
Momina Shaheen
Momina.shaheen@roehampton.ac.uk; s2018288003@umt.edu.pk
[1] School of Systems and Technology, University of Management and Technology, Lahore 54000, Pakistan
[2] Department of Computer Science, COMSATS University Islamabad Lahore Campus, Lahore 54000, Pakistan

privacy preservation, security assurance, and the management of disparate datasets. The wide-ranging utility of FL encompasses various domains, including prognostication and monitoring of mobile traffic [4], healthcare [5–7], the emerging field of the Internet of Things (IoT) [8], agriculture [9, 10], transportation and autonomous vehicles [11], finance and stock market [12], disaster management [13], pharmaceutical sciences, and advanced medical artificial intelligence [14].

Preliminaries

FL is one of distributed machine learning (DML) technique that allows multiple nodes to train a machine learning model without exchanging data samples [15, 16], as shown in Fig. 1. FL differs from DML in several ways. In DML, the data is first centralized on the server, and then the server splits it into subsets for learning tasks. In contrast, in FL, the data is not concentrated on the server, but rather the algorithm is distributed over the edge devices for processing [4, 13]. This means that FL has more training subsets than DML, and the data may not be identically distributed [17]. It presents new encounters to existing privacy-preserving techniques and algorithms [18]. It is crucial to create computationally and communication-efficient techniques that can withstand dropped devices without sacrificing accuracy, in addition to offering stringent privacy guarantees.

Process of federated learning

FL is an iterative approach that incorporates several client–server interactions, known as a FL round, to achieve higher performance than centralised machine learning [19]. Diffusing the current or updated global model state to the contributing nodes (participants) initiates each interaction/round of this process. After that, the nodes' local models are taught to produce prospective model updates. Subsequently, an aggregated global update is created by processing and combining the changes from local nodes [20]. This makes it possible to update the central model appropriately (see Fig. 1). With this system, local updates are processed and combined into global updates by a central server, called the FL server. Local nodes carry out the local training in accordance with the FL server's directives. The model is trained iteratively. The following describes the specifics of these steps:

Step 1. *Setup and Initialization:* A central server or orchestrator manages the FL process. It holds the initial model architecture and distributes updates. Edge devices/clients are individual devices that store local

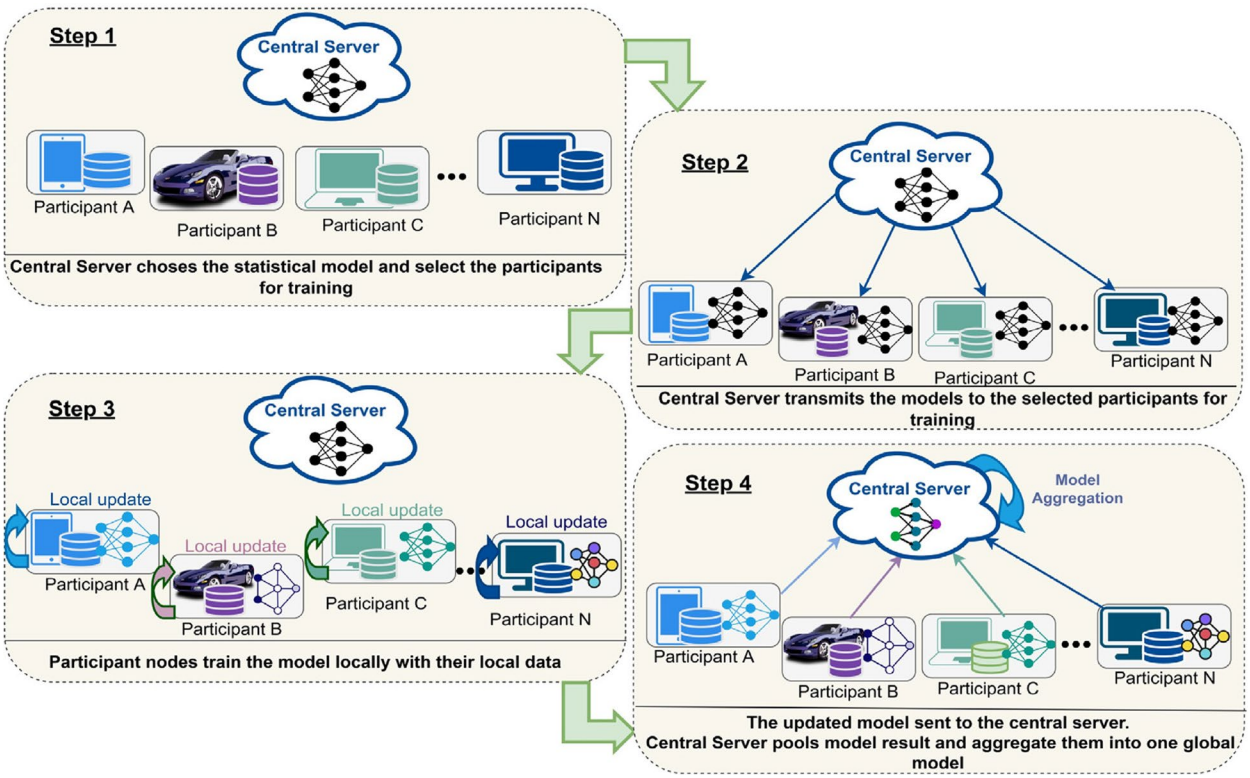

Fig. 1 Step by step process of federated learning

data and participate in the training process. The central server creates an initial model and sends it to all participating edge devices [21].

Step 2. *Local Training:* Each edge device trains the model received from the FL server, using its own local data. The training process might involve multiple iterations or epochs to improve the model's performance [22–24].

Step 3. *Model Update:* After local training, each edge device generates a model update, which essentially consists of weight changes that reflect what the device has learned from its local data. However, the actual data remains on the edge device and is not shared.

Step 4. *Models Aggregation and Global Update:* The edge devices dispatch their model updates back to the central server. The central server aggregates these updates using techniques such as averaging or weighted averaging. This creates a global model update that incorporates the knowledge from all edge devices without revealing their individual data. The central server applies the aggregated model update to the global model, enhancing its performance by leveraging the collective knowledge of the edge devices.

This is an iterative learning process. Steps 2 to 4 are repeated for a predefined number of iterations [25]. In each iteration, the edge devices refine their local models, and the central server integrates their updates into the global model. Over multiple iterations, the global model converges to a state where it becomes more accurate due to the aggregated insights from the diverse data sources on the edge devices.

Motivation and rationale

Most classification tasks involve imbalanced classes, which can result in biased training of machine learning (ML) algorithms. Learning with an imbalanced distribution is a challenging problem in ML. One common solution to this problem is ensemble learning, which combines multiple models to improve overall performance. Another solution is sampling, which involves subsampling the data to obtain a balanced proportion of classes. However, sampling can be computationally expensive, and it is not always feasible to obtain a large enough dataset to utilize this technique. FL can be a promising solution for learning with imbalanced data. FL can address the privacy concerns associated with data sharing, and it can also be more efficient than DML in terms of computation and communication. However, there are still challenges to be addressed, such as the need to develop robust FL algorithms that can handle imbalanced data. Based on the nearby data they have access to; the edge nodes train a shared model. The distribution of edge data depends on how they are used. For example, cameras installed in parks tend to capture more photos of people compared to cameras located in the wild. These imbalances can be categorised into three types for better comprehension; *Size Imbalance* (irregular size of the data sample on each edge node); *Local Imbalance* (non-IID (non-identical distribution) and independent distribution of data [26]); *Global Imbalance* (data residing at all nodes class imbalanced classes [27]).

Problem statement

Federated Learning is an effective machine learning strategy with advantage of data privacy protection. However, it struggles to deal with unbalanced or skewed datasets present over edge devices. This local and global imbalanced data distribution leads to bias in the iterative model training phase and results in a decrease in the accuracy of FL execution [26, 27]. The objective of this research is to improve the accuracy by addressing the challenge of local data imbalance in a federated fashion. Moreover, solving the issue of imbalanced data without compromising privacy or increasing computation overhead.

Contributions of this article

This study primarily contributes to the solution of unbalanced data problems by applying a method that corrects imbalanced data via client-side class estimation and data augmentation.

1. The Balanced FL (Bal-Fed) [12] approach, has been utilized for implementation in the FL setting. This technique is tailored for attaining maximum accuracy with less training rounds to reduce the computation cost. Its goal is to achieve a balance between training accuracy and reduced computation cost.
2. We applied this approach to train a Linear Regression machine learning algorithm in an FL setting, using an unevenly distributed dataset.
3. To evaluate Bal-Fed applicability in diverse problems, this approach is implemented on both textual and visual datasets separately. Two distinct datasets are utilized in this study. The first dataset is FashionMNIST, which consists of image data. The other dataset is stock market data, which includes both text and numerical data.
4. The dataset of last 10 years stock market prices is fetched for the stocks of Amazon and Booking.
5. By showing positive results in both kinds of datasets, this method has been demonstrated to improve the model's accuracy in the FL setting. In the FL setting, it demonstrates an accuracy rate higher than 92% and

95% for images and stocks data, after dealing with the problem of data imbalance.
6. This approach yields optimal performance with in 80 iterative rounds (while the pre-set iterations are 100) and terminates the iterative process, thus putting lesser computation load over the mobile edge devices.

Organization of the article

This article is divided into five main sections. "Related Work" section explains the studies and presents the results of the experiments conducted to address issues and challenges like the one highlighted. "Materials and Methods" section elaborates on the methodology and materials used to carry out the experiments. "Setup and Implementation" section comprises the setup and implementation of the experiment. Results of the implemented methodology are described in "Results and Discussion" section, where "Discussion on the results of Fashion MNIST (images) data" presents the results of the image dataset, while "Discussion on the results of stock data" presents the results for text data. "Implications" explains the implication of this research and highlights applicability of the field. Finally, "Implications" section concludes the entire experiment and presents the corresponding results. It also highlights future directions in this area.

Related work

In the realm of distributed data management, FL emerges as an evolving paradigm designed to address the complexities of privacy preservation. The development of healthcare frameworks has captured the attention of numerous research endeavours [28–31]. While the landscape of Machine Learning (ML) comprises a myriad of approaches and frameworks, there is a scarcity of comprehensive investigations that delve into the assessment of data balance within FL paradigms [32]. This section undertakes a thorough review of these empirical investigations, with a specific emphasis on studies relevant to our own research, which are concisely summarized in Table 1.

Across the network, nodes frequently accumulate and aggregate data in a manner that deviates from the Independent and Identically Distributed (IID) assumption [33, 40, 41]. In the context of next-word prediction, cellular users might extensively engage with linguistic expressions. Moreover, the volume of data across several nodes can vary significantly. The improvement of the FL algorithm's convergence trajectory requires an evaluation of

Table 1 Summary of the literature

Author	Research Type	Problem Area	Contribution	Related Studies
Zhao et. al	Experiment	Statistical heterogeneity	A method is developed to enhance training on non-IID data by generating a restricted subset of data that is distributed globally across all edge devices [33]	[3]
Mcmahan, et.al	Experiment	Communication cost	A realistic method for the FL is based on iterative model averaging is proposed and evaluated an exhaustive empirical evaluation [3]	[27, 33, 34]
C. T Dinh et	Experiment	Convergence analysis of FL algorithms and resource allocation	An optimization issue of resource allocation in wireless networks is addressed by proposing a FL algorithm. The goal is to capture the trade-off between the convergence time of FL and the energy consumption of UEs with heterogeneous computation and power resources [34]	[27, 33]
W. Luping et. al	Experiment	Communication cost	They suggested a system called Communication-Mitigated Federated Learning (CMFL), which provides clients with feedback on the overall trend of model updates [35]	[21, 24, 36]
M. Duan et.al	Experiment	Statistical challenges in FL	They provided evidence that inaccurate FL will result from unevenly distributed training data [27]	[34, 37]
S. U. Stich et.al	Experiment	Communication cost	They suggest structured updates, which would allow them to directly learn an update from a constrained space parametrized by utilizing fewer variables, thereby reducing the communication cost by two orders of magnitude [38]	[35, 36, 39]
D. C. Verma et. al	Numerical Experiment	Communication cost	When equipped with error compensation, stochastic gradient descent (SGD) with k-sparsification or compression (such as top-k or random-k) converges at the same rate as vanilla SGD, according to an evaluation of this technique that considers accumulated errors in memory	[35, 36, 38, 39]

the statistical heterogeneity intrinsic to the data. Recent research has introduced methodological tools to quantify statistical heterogeneity by applying relevant metrics [42]. Notably, these metrics, although valuable, cannot be quantified until training begins. Addressing this, Verma et al. proposed strategies specifically designed to improve machine learning models, even when dealing with highly skewed data distributions [37]. This investigation covered a wide range of environmental contexts. Noteworthy among these investigations is an AI model that emerges from a combination of heterogeneous data sources, representing the FL methodology. In a similar vein, a significant contribution emerges in the work of authors who elaborated on an expanded version of DropConnect, known as DropConnect Generalization [43]. This innovation plays a role in regulating densely interconnected neural network layers. DropConnect selectively nullifies a fraction of the network's weights, in contrast to the Dropout technique, which extends this nullification to randomly selected activations within each layer. A comparative analysis was conducted between DropConnect and Dropout across various datasets. Notably, the integration of multiple DropConnect-trained models outperformed in ground-breaking results across various benchmarks for image classification.

The procedural algorithms that enable individual clients to independently update their respective local data within the existing model were originally formulated by Konecny et al. in 2016 [36]. These algorithms enable clients to transmit their updated data to a central server. The server then aggregates the changes from multiple clients and computes a fresh global model. Primarily targeted at mobile phones, the efficacy of communication among the main constituents of this system is paramount. In this research, structured updates and sketch updates, were introduced to mitigate the costs associated with uplink transmission. Notably, Chen et al. [44] elucidated an end-to-end tree boosting mechanism referred to as XGBoost, which is frequently adopted by data scientists to achieve state-of-the-art results across diverse machine learning projects. Their work introduces a novel approach for handling sparse data, known as sparsity-aware methodology, as well as a weighted quantile sketch designed specifically for tree-based learning. The study further explores methods to improve the scalability of XGBoost by examining data compression techniques, cache access patterns, and sharding. Ultimately, XGBoost has been demonstrated to scale adeptly to billions of samples, while consuming considerably fewer resources than previous systems [4].

The significance of imbalanced datasets and their multifaceted applications within data mining were initially introduced by Han et al. [45]. Following this, they synthesized performance evaluation matrices and existing strategies aimed at mitigating the challenges posed by imbalanced data. The popular oversampling strategy, SMOTE is used to address this issue. This study introduces two additional variations, namely borderline-SMOTE 1 and borderline-SMOTE 2, which enrich the oversampling methodology. Nilsson et al. [46] conducted a benchmarking analysis on three FL algorithms. By centralizing data storage, the efficacy of these algorithms is appraised and compared. Notable among these algorithms are Federated Averaging (FedAvg), Federated-Stochastic Variance Reduced Gradient, and CO-OP. Their evaluation encompasses both non-IID. and IID. data partitioning schemes were used with the MNIST dataset to evaluate their performance, and it was found that the FedAvg algorithm achieved the highest accuracy.

Integrating FL with deep reinforcement learning (DRL) to enhance caching in edge is introduced in [47]. Their application, called the "In-Edge Al" framework, enhances caching, networking, and mobile edge computing (MEC). The framework effectively utilizes edge nodes and device collaboration, demonstrating robust performance with minimal learning overhead. The authors also highlight challenges and opportunities, emphasizing the promising potential of the "In-Edge Al" framework [14]. Similarly, Xu et al. [48] conducted a comprehensive survey on the expansion of FL in healthcare informatics. Their work addresses vulnerabilities, common statistical issues, remedies, and privacy concerns associated with FL. The outcomes of their research are envisioned as essential tools for the computational exploration of ML algorithms are tasked with managing extensive distributed data while also considering privacy and health informatics.

Clustered Federated Learning (CFL) [49] was developed as a solution to the decrease in accuracy in FL settings caused by divergent local client data distributions. CFL supports Federated Multi-Task Learning (FMTL), leveraging the geometric properties of the FL loss surface to effectively cluster client populations based on their data distribution characteristics. This clustering process maintains the communication mechanism of FL intact, providing robust mathematical guarantees for the quality of clustering. It integrates deep neural networks (DNNs) and ensures scalability while preserving privacy.

Frameworks for secure FL are introduced in [50], offering a comprehensive and robust platform that includes Federated Transfer Learning (FTL), vertical and horizontal FL. These frameworks are accompanied by concepts, infrastructure details, implications, and a comprehensive examination of advancements in this domain. The authors also advocate for the establishment of data networks between enterprises, based on federated processes,

to facilitate data sharing while respecting end-user privacy [51].

In a recent contribution, Mohri et al. [52] presented an agnostic FL framework that optimizes a centralized model to be adaptable to various target distributions. Their framework is lauded for instilling a sense of fairness, as they propose a rapid stochastic optimization approach to address the related optimization challenges. Convergence bounds are provided, assuming a hypothesis set and a convex loss function. The effectiveness of their approach is demonstrated across diverse datasets, indicating its potential applicability to contexts beyond FL learning, such as domain adaptation, cloud computing, and drifting [41]. In the realm of mobile devices, Bonawitz et al. [53] devised a scalable production method using TensorFlow (TF). Their work outlines foundational concepts, addresses challenges, and offers potential solutions [54].

It is evident, considering recent advancements in the field of FL, that numerous frameworks and techniques have emerged to address challenges such as communication costs, statistical heterogeneity, convergence, and resource allocation. In FL and imbalanced data, scholars have investigated diverse strategies to alleviate the influence of class imbalance on model performance. These methods encompass oversampling [45], target distribution [52], and class weights [44]. The efficacy of these techniques can be contingent on the distinct characteristics of the dataset and the FL configuration. However, the issue of class imbalance and data imbalance remains inadequately addressed in certain works. This research persists in practicing existing methods and introducing novel approaches to enhance the management of imbalanced data within federated learning. This article seeks to bridge the gap by focusing on and addressing the challenge posed by class imbalance through a novel approach.

Materials and methods

The issue of data balancing is effectively solved in centralized ML after decades of research. FL is relatively a new emerging area where it is necessary to maintain privacy in composition. Balanced federated data can be achieved by generating augmented and synthetic data [55, 56] on mobile edges, without compromising privacy. This follows the post-processing guarantees of differential privacy (DP) [57]. Augenstein et al. [58] explored and demonstrated the federated fashion of generating synthetic data. In the federated setting, data synthesis can be used. Additionally, the client's estimation must be employed in the approach of self-balancing. As a solution, we utilized our FL approach called Bal-Fed (as shown in Fig. 2) that will be implemented to rebalance training. This approach is proven to be successful for the implementation of balanced federated learning for the stock market data for some of the stock [12]. We now using this approach to prove the applicability of this approach to images data using benchmark dataset FashionMNIST. The whole strategy developed to achieve the objective of data imbalance reduction with optimized computation cost entails the following steps.

Step 1. Selection of nodes at the mobile edge layer.
Step 2. Executing the Class Estimation of the edge clients.

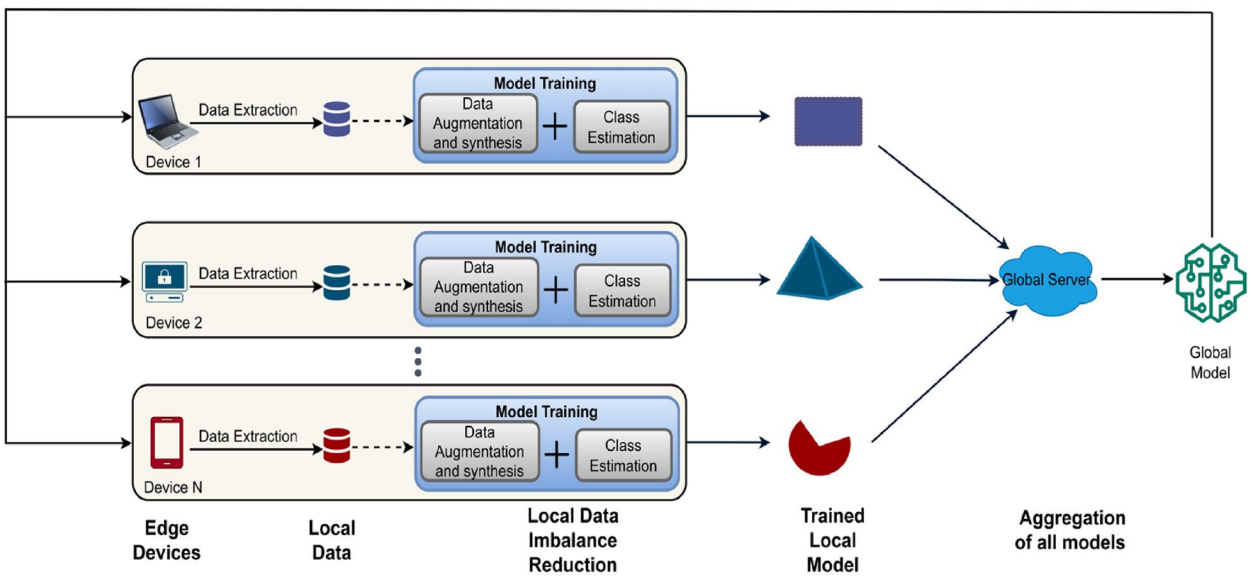

Fig. 2 Proposed methodology working in the scenario of edge networks

Step 3. Performing data augmentation [59] and class balancing algorithm on the global distribution to address data bias.
Step 4. The linear regression algorithm is used for model training using data from each mobile edge node (as depicted in Fig. 2).
Step 5. Sending the updated models to the server for aggregation, which is performed using FedAvg.

Merging these two approaches for implementing and measuring the proposed solution on the Flower Framework and TF using distributed datasets. For this research, we utilized the Fashion MNIST dataset and collected the stock market dataset to assess the model's fitting to predict stock prices. Specifically, we used Amazon (AMZN) and Booking incorporation (BKNG) for our research.

These datasets were converted into distributed datasets to make them suitable for the FL framework. The utilization of two distinct datasets serves the purpose of showcasing the versatility of BalFed across datasets of different natures. The stock market dataset includes both numerical and textual data, while the Fashion MNIST dataset consists of image data. The reason for selecting this dataset is to assess the performance of the model on established datasets and compare its results with existing outcomes. The decision to include the stock price dataset is prompted by the limited research conducted on the application of FL in stock price prediction. The majority of research in FL settings has primarily focused on stock news and other areas not on predictions. This choice contributes to addressing a potential gap in this domain [60–62]. Both datasets serve distinct objectives: the former is used for prediction tasks, while the latter is employed for classification problems. This deliberate differentiation helps evaluate the model's adaptability within the FL framework when faced with diverse problem domains.

Global aggregation over cloud server

For the aggregation of the global model, we adopt the well-established FedAvg algorithm. In synchronization with Algorithm 1 randomly selected subset of the federation's members (clients/devices) was designated to acquire the initial global model [63]. In the subsequent step, each client selected for the ongoing round of training computes updates to its local model using its own dataset. These updates are then communicated back to the server [64, 65], as described in "Process of federated learning" section. In the pursuit of refining the collective model, the server performs an averaging process on all the updates contributed by the clients. This iterative procedure continues until the model parameters converge, as determined by appropriate criteria. At that point, the process is repeated with a new round of training.

Algorithm 1 Federated Averaging (FedAvg). There are n edge devices, B is the local minibatch size, E represents total local epochs per communication round, η is learning rate, and f_i is the loss function

Server Executes
initialize ω_0;
for each round $t = 0, 1, \ldots$ **do**
for each client $i = 0, \ldots, n\text{-}1$ in parallel
$\omega^t \leftarrow \text{ClientUpdate}(i, \omega_t)$
$\omega_{t+1} \leftarrow \sum_{i=1}^{n} n\ \omega_{t+1}$
ClientUpdate (i, ω): *//Run on client i*
for each local epoch e from $0, \ldots, E\text{-}1$ **do**
for each minibatch b of size B **do**
$\omega_{e+1} \leftarrow \omega_e - \eta \nabla f_i(\omega_e; b)$
return ω_E to server

At the client side, "Gradient descent" takes place, and on the server side, aggregation takes place over "averaged clients updates". The amount of client computation is controlled by three key parameters. The fraction of clients that perform computation on each round is denoted by I.

Local training over mobile edge devices

Linear regression

Interpreting and understanding linear regression is simple. A linear equation, which is simply understood and visualised, represents the relationship between each independent variable and the dependent variable. Using linear regression analysis, one can forecast a variable's value depending on the value of another variable [66]. Predictability is required for the dependent variable. It is possible to anticipate the value of the other variable by using the independent variable [63]. This analysis determines the coefficients of the linear equation by using one or more independent variables that can precisely predict the value of the dependent variable. The linear regression technique lessens the discrepancies between expected and actual output values by fitting a line or surface. Simple linear regression algorithms that find the best-fit line using the "least squares" method can be created from a collection of paired data.

$$y = \beta 0 + \beta 1X + \varepsilon \tag{1}$$

Based on a certain value of the independent variable (x), the variable y indicates the expected value of the dependent variable (y). The intercept, or the expected value of y when x = 0, is represented by the symbol β0. Conversely, the regression coefficient β1 represents the anticipated shift in y with an increase in the independent variable (x).

Since it is anticipated to have an impact on y, variable x is regarded as the independent variable. The degree of variation in the regression coefficient estimation is quantified by the variable ε in the equation, which stands for the error of the estimate. Finding the regression coefficient (β1) that minimises the model's total error (e) will yield the best fit. Linear regression commonly employs the mean-square error (MSE) as a metric to evaluate the accuracy of the model. MSE is computed through:

Calculating the difference between the observed and predicted y-values for each associated x-value is the first step in the procedure. Afterwards, each of these distances must have its square calculated as part of the procedure. Each squared distance's mean is computed. A straight line is fitted to a set of data points using the statistical technique of linear regression, which entails figuring out the regression coefficient that minimises the mean squared error (MSE).

Although convolutional neural networks (CNN) are frequently employed to process visual input, their computational cost is relatively high [67]. Because FL computation is done at the client side, it is preferable for edge nodes to incur lower calculation costs. In comparison to more intricate models like SVM, Random Forest, and DL [68–71], linear regression techniques have lower computational requirements [60], which makes them appropriate for situations with limited processing resources or big datasets. In the FL configuration, Random Forest (RF) takes 515 s, whereas SVM takes 4989 s for the training cycle, and LR only takes 7.6 s [60]. As a result, we trained the clients' data locally using this technique. Additionally, this method yields results that are similar to CNNs. When the dependent and independent variables are both continuous, as is the case in the analysis of stock market datasets, LR is a good fit. Likewise, other studies have demonstrated the effectiveness of this approach on the FashionMNIST dataset.

Class balancing

In FL circumstances, it was not possible to obtain mobile edge node raw data in order to protect client privacy. For this reason, in accordance with their updated gradients, the class distribution along the edge side is assisted by the class estimation and balancing method [72]. After that, this class estimation method was applied to even the classes and accompanying data by using data augmentation [59]. The expected of gradient square for various classes during model training in FL has the approximate relationship shown below [73].

$$\frac{E||\nabla L(w_i)||^2}{E|\left|\nabla L\left(w_j\right)\right||^2} \approx \frac{n_i^2}{n_j^2} \tag{2}$$

where L denotes the cost function of the training algorithm. And for class i and class j the number of data samples n_i and n_j are; where $i \neq j$ and $i, j \in C$. Due to the correlation between gradient and class distribution, a class Ci, with class ratio $\frac{n_i^2}{\sum_j n_j^2}$, get a class estimation [72] as defined:

$$R_i = \frac{e^{\frac{\beta}{||\nabla L^{aux}(w_i)||^2}}}{\sum_j e^{\frac{\beta}{||\nabla L^{aux}(w_j)||^2}}} \tag{3}$$

In order to achieve class normalisation, β is adjusted as a hyperparameter. It is therefore possible to establish the composition vector $R=[R1,..., RC]$ that represents the raw data distribution. Consequently, each mobile edge node's class imbalance is evaluated by the Kullback–Leibler (KL) divergence using U, the vector of classes of magnitude C.

$$D_{KL}(R||U) = \sum\nolimits_{i \in C} R_i log \frac{R_i}{R_j} \tag{4}$$

After updating the model during FL training, the server can get the local model from every client device. With the class estimation method, the composition vector R^k for the selected client k can be revealed. Then, we define the reward for client k as follows:

$$r^k = \frac{1}{D_{KL}(R^K||U)} \tag{5}$$

The composition vector can be used to determine the class distribution. For instance, $Rk(t)$ denote the composition vector of client k at time slot t. Consequently, the class ratio can be approximated using mean of composition vector, which can be defined as.

$$\overline{R}^k = \frac{\sum_{t=1}^{T^k} R^k(t)}{T^k} \tag{6}$$

With the estimated composition vector $\overline{R}$ and reward r of each client, we can design the client selection scheme with minimal class imbalance according to Algorithm 2.

Algorithm 2 Class balancing algorithm

Initialize
Set $S_t = \emptyset$ and $R_{total} = \emptyset$
$k_0 = \arg\max_k r^{\hat{}}{}^k$
$St \leftarrow St \cup \{k_0\}$
while $|St| < K$ **do**
Select $k_{min} = \arg\min_k D_{KL}\ (\ (R_{total}+\overline{R}^k\ )||U\ )$ for $k \in K \backslash St$
Set $St \leftarrow St \cup \{k_{min}\}$, $R_{total} \leftarrow R_{total} + \overline{R}^k$ min.
end while
Outputs: St

Class estimation and data augmentation

The class estimation technique employed in [74] was adopted in this study, and the same technique was subsequently utilized for class estimation. Data augmentation, as outlined in [75], was then applied. Data augmentation, as elaborated upon in references [73, 76, 77], refers to methodologies within data analysis aimed at expanding the volume of the dataset. This is accomplished by creating new synthetic data that is generated from the present dataset or by attaching significantly altered copies of the existing data. Data augmentation is incorporated to help prevent overfitting and to offer regularisation for machine learning models during training.

For the context of a FL system focused on multi-class classification tasks, the system architecture includes a central server responsible for managing the global model. Accompanying the server are a collection of clients, denoted as $K=\{1, 2, \ldots, K\}$. Each client possesses an independent local dataset, designated as D_k. During the *rth* iteration of the FL process, a designated client, known as client k, is selected to participate in the learning process. This entails starting local learning with its unique local dataset D_k and the initial global model vector w_r that the server provides. Following that, client k uses its local dataset D_k to create a mini-batch collection, designated as B_k. Stochastic gradient descent (SGD) optimizer is used for the subsequent local learning [68, 78]. The following is the definition of the updating mechanism for this localised learning project:

$$W_{k,r+1} \leftarrow W_{k.r} - \eta \frac{1}{|D_k|} \sum_{x \in Bk}^{K} \nabla \; fk\left(W_{k,r}; X\right), \;\; \forall k \; \epsilon \; K \tag{7}$$

In the above equation, $|Dk|$ represents the size of the dataset Dk, while $fk(wk,r; x)$ stands for the loss function associated with the local model vector wk,r and the data instance x. The learning rate is denoted by η. The training of the local model is carried out for a predefined number of local epochs for each chosen client. The locally obtained vector is subsequently sent to the central server. The incoming local model vectors are then aggregated to update the global model vector that is kept up to date by the server. Every local model is given a unique weight throughout the aggregation process. These weights are determined by dividing the total amount of data utilised by all participating customers by the percentage of data used in local training for each client. The process of aggregating these weights is mathematically expressed as:

$$\partial_{r+1} \leftarrow \sum\nolimits_{k \epsilon S} \frac{|D'_k|}{|D|} \partial_k, r+1 \tag{8}$$

Here, S refers to the set of clients selected by central server to participate in the learning process, D represents the union of the local datasets from all the selected clients. The term D'_k denotes the data specifically used by client k for local learning, and it holds the relationship $D'_k \subset D_k$. This iterative process is reiterated until a predetermined round threshold is reached as given in Fig. 3.

Setup and implementation

The proposed framework should be implemented to achieve a balanced training process, as illustrated in Fig. 2. Data augmentation is employed for augmenting the amount of data by either adding modified copies of existing data or generating synthetic data from the current dataset. This technique helps to reduce overfitting. We are also considering future improvements for this study, such as utilizing data synthesis to create a new dataset based on the existing one. It takes.CSV data as input and produces a synthetic dataset using DP.

The model training and aggregation is set up in a sequential workflow of Fashion MNIST and stock data that is illustrated in Fig. 4. We have automated the process of converting the collected data into federated data. Every client receives a random distribution of the data. Next, the BalFed method is utilised. Open-source frameworks for decentralised data machine learning and other computations are the Flower and TF frameworks. TF was developed to enable collaborative research and testing with FL, a machine learning technique that involves building a shared global model across several clients using locally stored training data. With Flower and TF, developers may test new algorithms and simulate the integrated FL algorithms on their models and data. With TF's building blocks, non-learning computations can be implemented, including federated analytics, using python which is a concise and legible programming language that is used to create and simulate algorithms.

Results and discussion

This section encapsulates all the findings obtained throughout the course of this research. Within this section, we present numerical results that clearly demonstrate the effectiveness of the proposed algorithms. The effectiveness of our developed methodology was subjected to rigorous evaluation using Fashion MNIST, a well-known benchmark dataset of importance.

Discussion on the results of fashion MNIST (images) data

Fashion MNIST is widely used for testing and benchmarking machine learning algorithms, especially in the context of image classification and deep learning. Its similarity in size and structure to MNIST makes it an ideal substitute when researchers want to experiment with more complex datasets without significantly increasing the computational requirements.

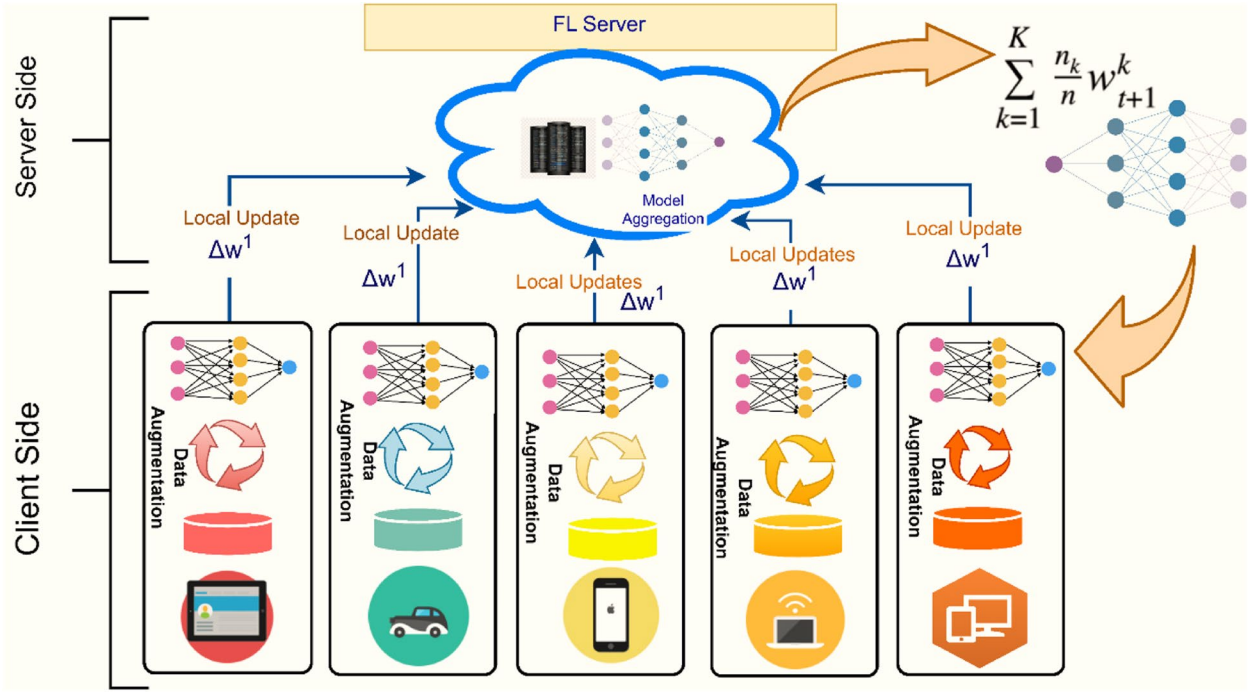

Fig. 3 Workflow of the Bal-Fed technique for client-side data augmentation with the implementation of FedAvg Algorithm

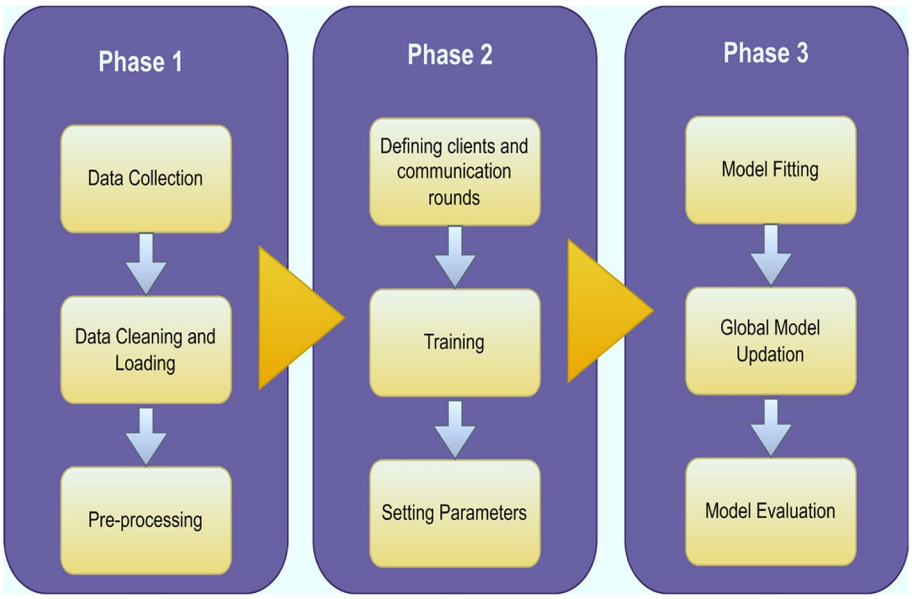

Fig. 4 Schematic workflow sequence used for stock data applied for comparing the prediction performance of BalFed

The Fashion MNIST dataset consists of 70,000 images of Zalando's clothing articles. It includes various types of clothing such as T-shirts, shoes, pants, tracksuits, etc. Each type is assigned a numerical value, for example, 0 for T-shirt/top, 1 for Trouser, 2 for Pullover, 3 for Dress, 4 for Coat, 5 for Sandal, 6 for Shirt, 7 for Sneaker, 8 for Bag, and 9 for Ankle boot. Images are in grayscale with a size of 28×28 pixels, as shown in Fig. 5. It's worth noting that while Fashion MNIST has been widely used, more challenging datasets with higher complexity and diversity have emerged in recent years to push the boundaries of machine learning performance in image recognition tasks.

Fig. 5 28×28 images from fashion MNIST

An algorithm's performance is typically assessed using a confusion matrix that shows the mistakes that were made. The number of projected test results that belong to the right class and the number of results that are assigned to the wrong class are displayed in this matrix. The information inserted into the matrix is useful for assessing and figuring out the algorithms' evaluation metrics. The widely used criterion of accuracy, which is defined as:

$$Accuracy(Acc) = \frac{TP + TN}{N} \tag{9}$$

where $N = TP + TN + FP + FN$ and following quantities of confusion metrics are represented as: *TN (true negatives), FN (false negatives), TP (true positives), FP (false positives).*

Accuracy refers to the ratio of the total number of correct predictions and classifications to the total number of accurate and incorrect forecasts. In the field of statistics, it is also referred to as precision. Accuracy can be misleading in several instances. For classification accuracy, it can provide a better insight into the performance of the model. However, it may be necessary to choose a model with lower accuracy due to the increased predictive power it provides in a specific situation. As a result, it is a good idea to utilize alternative performance metrics, such as the F1 measure, which is represented as Eq. 10.

$$F1 - Score = F1 = \frac{Precision \times Sensitivity}{Precision + Sensitivity} \quad (10)$$

To simulate the class distribution of the entire dataset, we performed an automated conversion of the dataset into a federated dataset. So that each client has a random distribution of classes and data samples. In other words, the distribution of data varies depending on the client. Our model is developed using linear regression [79]. In general, there are 1,22,570 parameters. A commonly used classifier, such as linear regression, can fulfill our requirement to validate the effectiveness of our BalFed approach.

For this experiment, we used TF and Google Colab. We used a standard SGD optimizer with the classifier. The selected 20 clients trained their local models for 20 epochs in each training round out of 100 rounds. At each training epoch, the client selects 10 batches with a batch size of 10. From this data, 60,000 images will be treated as a training set and 10,000 as a test set, as recommended. After implementing the Bal-Fed setting, the data is loaded onto 20 clients to obtain the results based on the evaluation measures. As compared to the FL technique, which gives 85% accuracy [79], using Bal-Fed can increase the accuracy to 92% for the same dataset (as shown in Table 2).

Discussion on the results of stock data

Real-time data is gathered for this experimental investigation from the Y-finance API. Amazon (AMZN) and Booking Inc. (BKNG), two significant stock market businesses, are the sources of the data. The retrieved data was organised continuously between January 1, 2013, and February 1, 2023. For every stock, there were 2,517 records, organised into several CSV files. The date, closing price, and projection make up each record. An automated method is used to transform this data into a federated dataset.

Metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), *R*-Squared, and MSE are commonly used in regression analysis to assess the predictive error rates and model performance. The MAE, or difference between the expected and original values, is obtained by averaging the absolute differences over the data set. The mean square error (MSE) represents the variation between the original and predicted values by squaring the average difference across the data set. How well the data fit together in regard to the original values is indicated by the coefficient of determination, or R-squared. A number between 0 and 1 is interpreted as a percentage.

$$MSE = \frac{P - \widehat{\mathrm{P}}}{n} \quad (11)$$

$$MAE = \frac{\left|P - \widehat{\mathrm{P}}\right|}{n} \quad (12)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(P_i - \widehat{P}_i\right)^2}{\sum_{i=1}^{n}\left(P_i - \widehat{P}\right)^2} \quad (13)$$

where P and $\hat{\mathrm{P}}$ represent the true price and predicted price, respectively, and n denotes the number of samples in the test dataset. Training time is measured to study the latency of ML model training. It increases depending on the complexity of the model, the size of the dataset, and the performance of the processing framework. Training delay reduction offers real-time prediction/classification benefits. Table 2 shows the MSE results of the LR model executed on various ML frameworks.

This model is developed using linear regression [72] in the Flower Framework [80]. Using this framework and Scikit-Learn, the Bal-Fed is implemented. The framework's evaluation function is the Mean Squared Error (iteration-by iteration results are provided in Appendix A). The minimum number of edge nodes is set to 20. By doing this, each node trains a linear regression algorithm using data from a single province before sending the gradient of loss from the model to the server. The server updates model parameters using FedAvg. To update each local model, the new parameters are delivered to the worker nodes. Until the application requirements are satisfied, this process is iteratively performed in a convergent manner without sharing any data. We chose a 90% and 10% data split for training and testing.

After performing a linear regression analysis, line, residuals, and scatter plots are important tools for assessing the quality of the regression model and identifying potential issues or patterns in the data. A line graph, in the context of linear regression, represents the relationship between the independent variable (X-axis) and the dependent variable (Y-axis) based on the predictions of the linear regression model. However, it is important to note that a line graph is not typically used directly to represent the outcome of a linear regression analysis. Instead, it is more commonly used to demonstrate the trend or pattern in the data before and after applying the regression model. To address this issue, a scatter plot is

Table 2 Results of fashion MNIST applied to evaluate the prediction performance of BalFed

Evaluation Measure	Results	Evaluation Measure	Results
Overpredicted data	4.3%	Accuracy	92.1%
Underpredicted data	3.1%	F1	92.3%

utilized to validate our model and visually represent the actual data points. Each point on the plot corresponds to a pair of values from the actual and predicted stock prices. The scatter plot, presented in Fig. 6, showcases the actual data points and overlays the regression line. The regression line represents the line of best fit generated by the linear regression model. It is worth noting that the model's predictions align with the actual data.

Residuals are the differences between the actual/observed values and the predicted values generated by the LR model. In other words, they represent the errors or discrepancies between the model's predictions and the actual data points. Residuals provide insights into how well the model captures the underlying relationships in the data. A well-fitted model should have residuals that are randomly distributed around zero, with no discernible patterns. Patterns in the residuals can indicate issues such as underfitting or overfitting, heteroscedasticity (varying variance), or omitted variable bias.

The obtained data frame had columns named Date, Open, and Close. The prediction results of the proposed technique for the stock data are plotted in a line graph, as shown in Fig. 6. The predicted values are consistent with the actual values. The resultant graph and the fitted model in the scatter graph and residual graphs are shown in Figs. 7 and 8 respectively. For this dataset, 20 clients are selected for 100 communication rounds, with 5 epochs in each communication round. The reason for using fewer communication rounds with this dataset is that it achieved 95% accuracy with only 5 epochs. This accuracy is likely to increase by increasing the epochs, but it can lead to more communication time on client

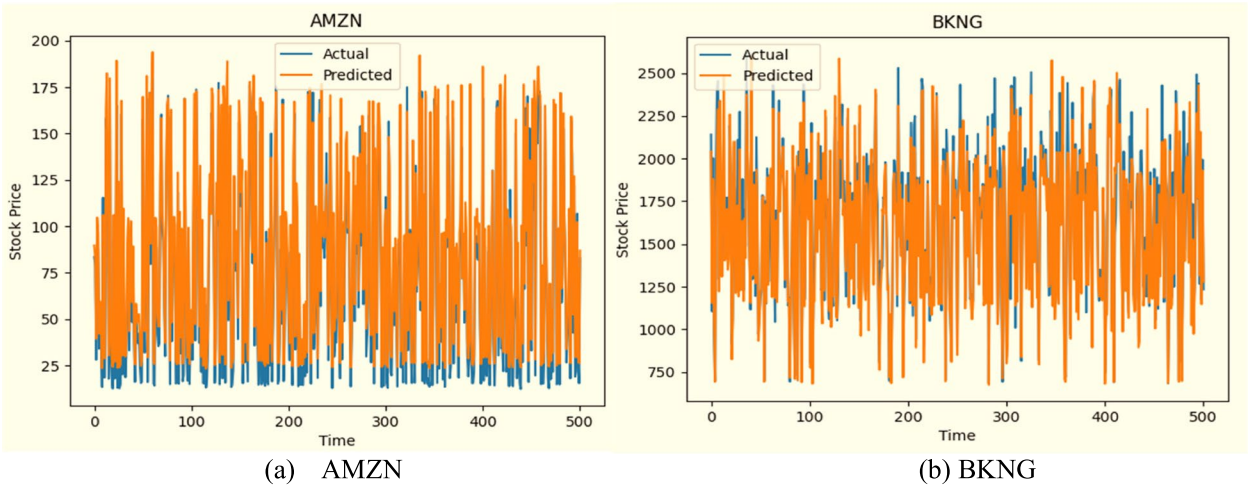

Fig. 6 Prediction of stock prices using Bal-Fed

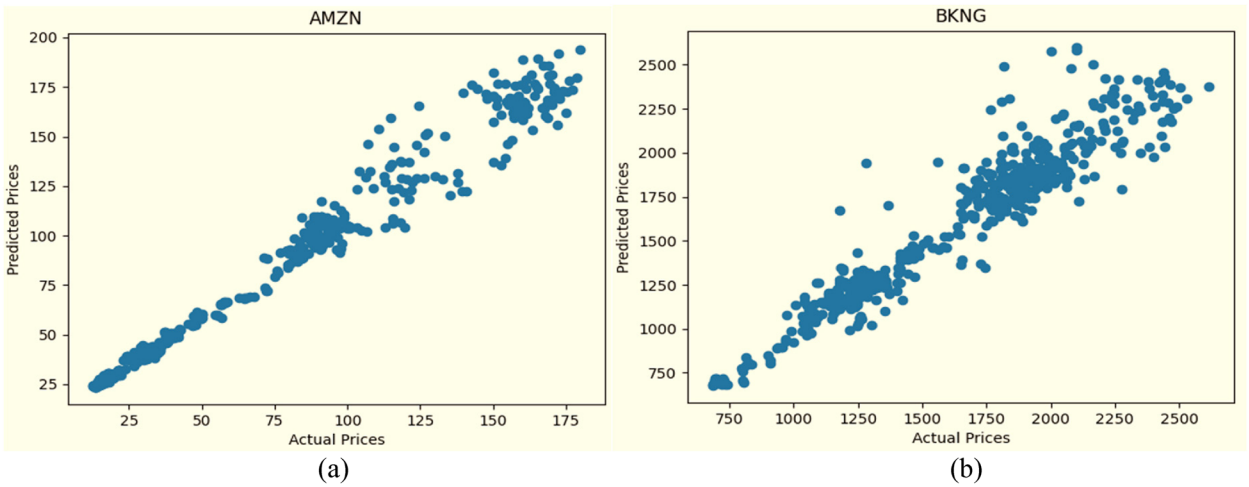

Fig. 7 Results of the BalFed model for stock price prediction in scatter graphs. **a** AMZN. **b** BKNG

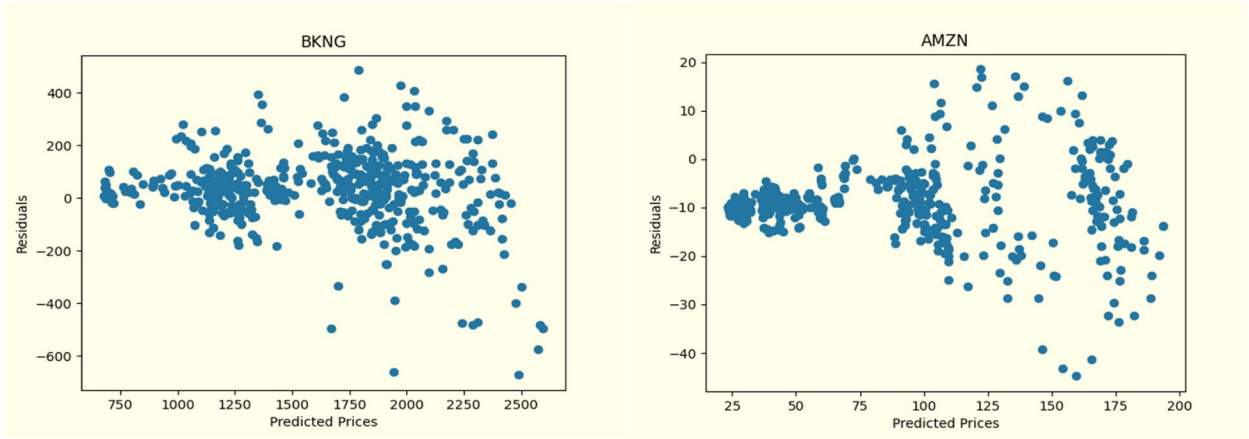

Fig. 8 Results of the BalFed model for stock price prediction in residual graphs

side. Thus, reducing learning time and computation costs the epochs are restricted.

The prediction values resulted in an accuracy of 95% with minimal data loss, as shown in Table 3. The values of Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared are obtained and are shown in Appendix A in Tables 4 and 5 for AMZN and BKNG respectively.

The accuracy of the predicted prices of the stocks and Fashion MNIST data is sufficient with the BalFed model, and it is better than the results reported in the literature, which provide 85% accuracy for the federated setting [79].

Analysis on the performance of bal-fed

This study presents a model inside the Flower framework that was created using linear regression. Metrics including *R*-squared, MSE, MAE, and RMSE are used in the evaluation; Table 1 provides a thorough breakdown of the results. Each edge node individually trains a linear regression algorithm with class estimation and balancing (Algorithm 2), with a minimum of 20 edge nodes. The gradient of loss is then sent to the server by each node. For every local model update, the FL server distributes the modified parameters to edge nodes using the FedAvg algorithm (Algorithm 2). Until the training termination requirements—100 rounds of local data training—are satisfied, this iterative process proceeds in a convergent manner without sharing any data. The model goes through several iterations in order to attain an optimal state, after which additional training has little effect on its performance. Using a decentralised or distributed training strategy, each client processes independently its own local data, with 90% of the data being partitioned for training and 10% for testing.

Twenty clients and one hundred communication rounds—each with five epochs—were included in the dataset. The dataset's remarkable 92% accuracy within 75 rounds was a factor in the decision to reduce the number of communication rounds. And the termination conditions were met in 80 rounds (see Tables 4 and 5 in Appendices A and B, respectively). As a result, the reduction in communication expenses and learning time was effectively achieved.

The mean absolute error (MAE) is the mean difference between the expected and actual values. It provides information about the average size of errors and is preferably less. Greater weight is assigned to larger errors in the Mean Squared Error (MSE) calculation, where lower MSE values denote superior performance. The average size of errors in the same units as the target variable is measured by RMSE, which is the square root of MSE and offers an extra evaluation of prediction ability. *R*-squared, which goes from 0 to 1 and indicates a perfect fit, evaluates how well the model's predictions account for variance in the actual data.

In Tables 4 and 5 in Appendices A and B, the *R*-squared values stand out with notably high values (e.g., 0.94, 0.90) across the majority of cases. This suggests that the model adeptly captures variations in stock prices, demonstrating strong performance. Using the AMZN stock as an example, during the 80th training iteration, the R2 is 0.93, the MAE is 10.75, the MSE is 153.5, and the RMSE is 12.9. The *R*-squared value indicates a moderate fit, while the low MAE, MSE, and RMSE values show that prediction errors have relatively small magnitudes. These metrics point to a moderately fitting model for AMZN stock AAL during

Table 3 Results of fashion MNIST applied to evaluate the prediction performance of BalFed

Evaluation Measure	Result
Accuracy	95.01%
Data Loss	19

the 80th training cycle. Comparable trends are noted with BKNG stock. In the case of Images data, both accuracy and data loss are minimal, indicating a strong fit of the model to the images data. However, it is important to note that further exploration is needed, particularly when dealing with data featuring more complex features.

Implications

FL presents a promising avenue for training decentralized data, residing on local client devices, thereby enhancing efficiency and safeguarding privacy. Nonetheless, the distribution and volume of training data at the clients' end can engender substantial challenges, including class imbalance and the presence of non-IID data. These challenges can exert a pronounced influence on the performance of the shared model. Despite concerted efforts to facilitate the convergence of FL models in the face of non-IID data, the issue of data imbalance remains inadequately addressed. As FL training entails the exchange of encrypted gradients, rendering the training data partially concealed from both clients and servers, conventional methods for addressing class imbalance exhibit suboptimal performance within the FL paradigm.

Hence, the development of novel techniques to detect and alleviate class imbalance in the FL context assumes paramount significance. This study introduces Bal-Fed, a method capable of inferring the composition of training data for each FL iteration, thereby mitigating the adverse effects of imbalance. Through experimental validation, we underscore the significance of class estimation and the implementation of client-side strategies in FL training. The efficacy of our proposed approach in reducing the imbalance's impact is vividly demonstrated. Notably, our method markedly surpasses prior approaches, while concurrently upholding client privacy. It achieves accuracy rates of 92% for image data and 95% for stock price data, underscoring its proficiency. Addressing imbalanced data in federated learning can have notable positive impacts across various applications.

In essence, addressing imbalanced data with bal-fed in federated learning can lead to more robust and accurate models with optimized communication cost. Specifically, in the scenarios where certain outcomes or events are infrequent but crucial for decision-making in various domains, such as in finance, telecommunication, environmental monitoring and retail. In finance to identify rare fraudulent transactions in financial datasets and enhancing risk prediction models by addressing imbalances in data related to high-risk scenarios. Similarly, in in telecommunication, the detection of rare security threats or attacks on telecommunication networks and in quality of service (QoS) prediction by enhancing models for predicting rare instances of service degradation. Additionally, it can help in improving models for identifying infrequent environmental events, such as natural disasters or unusual phenomena. Lastly, bal-fed can help in enhancing models to identify patterns in customer behaviour for targeted marketing strategies with ensuring the user data privacy.

Conclusions and future work

In traditional centralized machine learning, all local data is uploaded to a single server, which raises privacy concerns. FL is a machine learning methodology in which users' data is used to train a model, but the data is not shared with the cloud server. Only the results or trained models are uploaded. This approach is more efficient in terms of generalization, privacy, and system correctness. However, a major challenge in FL is data imbalance. This occurs when the distribution of classes in the local data of different clients is significantly different. To address this issue, we propose a data balancing technique called data augmentation.

This technique is implemented in the TF and Flower frameworks utilizes various deep learning (DL) algorithms address reduce data imbalance prior to the training of local models. We also address the problem of client selection caused by imbalanced FL data. We propose a method to manage the class distribution by automatically generating augmented data for each client during local model training. This is done without requiring any information about the clients' data. Additionally, we propose a combination of client selection and data balancing techniques to further mitigate the impact of data imbalance.

Our numerical results show that the proposed technique can select a well-balanced client set and improve the algorithm convergence performance of the global model. We applied the technique to the Fashion MNIST dataset and Stock Price Data, and it achieved good results in terms of accuracy and F1 measures. Extensive experiments demonstrate that our method can significantly outperform previous solutions for imbalanced data. The accuracy of the predicted prices of the stocks and Fashion MNIST data is sufficient with the BalFed model, and it is better than the results reported in the literature, which provide 85% accuracy for the federated setting [79].

In the future, we are planning to enhance our research even further and improve the prediction capabilities of Bal-fed. And to apply this technique to various applications such medical imaging and human activity recognition. We are intending to develop a mobile application that can benefit users in predicting the value of stocks. Moreover, we intend to use this mobile app for diagnosis through medical images.

Appendix

Table 4 Subset of the variations of *R* Squared, MAE and MSE w.r.t training iteration for Amazon stock data

Iteration	R^2	Mean Absolute Error	Mean Squared Error	RMSE
1.0	0.3615177826550059	40.96361933203462	1737.5484136732996	41.68391
2.0	0.9255199796283194	12.716530955158412	202.6879335579717	14.23685
3.0	0.8515782541509318	18.79209251323908	403.9109658011402	20.09754
4.0	0.9243534700591688	12.850174272434495	205.8624414711924	14.34791
5.0	0.9609688128652012	8.408773012706197	106.218427776839	10.30623
6.0	0.9250792087966996	12.7731807363897	203.8874354994026	14.27892
7.0	0.9617042805276084	8.28923883674258	104.21694576934058	10.20867
8.0	0.9286376443006984	12.397115932683178	194.20360438059365	13.9357
9.0	0.8620514402837214	18.039734846372887	375.4095174337836	19.37549
10.0	0.8743202633104741	17.15786093304368	342.0214708936338	18.49382
11.0	0.9384821120168914	11.359716950476905	167.41313348093698	12.93882
12.0	0.926201780429661	12.678215539696287	200.83249910947848	14.17154
13.0	0.886739542665897	16.180369846213594	308.2239765284608	17.55631
14.0	0.9700487663367746	7.000049442662013	81.50848547591667	9.028205
15.0	0.937015264424928	11.541408384374543	171.40497324918394	13.09217
16.0	0.9410540724981264	11.10399754816772	160.4138690804599	12.66546
17.0	0.9401899223977492	11.208699766396784	162.76554402973278	12.75796
18.0	0.9690657845859636	7.1852609748363205	84.1835456974748	9.175159
19.0	0.9301302255728854	12.255617368765243	190.14173366389383	13.78919
20.0	0.9687592319636918	7.224793924087916	85.01778979709947	9.220509
21.0	0.9329022814368728	11.981803812313634	182.597935045486	13.51288
22.0	0.8915445081247867	15.806575027457988	295.1478721608782	17.17987
23.0	0.9417113504650878	11.02097510494964	158.62517041695585	12.59465
24.0	0.8899391463823086	15.94524740114928	299.5166606302168	17.30655
25.0	0.8886091525513703	16.034310572021777	303.1360702368802	17.4108
26.0	0.9646866618021436	7.913732518296435	96.10077320922473	9.8031
27.0	0.8782390101493596	16.834853979885658	331.35709815384695	18.20322
28.0	0.9293871623254576	12.325391285130737	192.16388609312807	13.86232
29.0	0.9234656574829344	12.936987208509857	208.2785108488014	14.43186
30.0	0.936571013803313	11.605372140959007	172.613944997952	13.13826
31.0	0.9324229835842084	12.02345397518054	183.9022833905925	13.56106
32.0	0.9386227055164836	11.382648650451255	167.03052609493136	12.92403
33.0	0.8809643858529465	16.64830154339	323.94033367429296	17.99834
34.0	0.962823034984888	8.212727580896487	101.17239733912844	10.05845
35.0	0.8808074393616578	16.66970512144188	324.36744365411994	18.0102
36.0	0.9367202845672844	11.582531265434168	172.20772353696154	13.12279
37.0	0.9631794592999776	8.134917631988039	100.20243374976867	10.01012
38.0	0.9410027797806186	11.112040113751195	160.55345570874437	12.67097
39.0	0.9633998162196346	8.042574696610275	99.60276032772089	9.980118
40.0	0.9412937119452012	11.07017553588928	159.761720704504	12.63969
41.0	0.9443041165103528	10.745022147360432	151.56928631150757	12.31135
42.0	0.9442515753995112	10.722724533058134	151.71227028398997	12.31715
43.0	0.9439412662172152	10.753689211848709	152.55673738549885	12.35139
44.0	0.945258265238518	10.609202304849765	148.97269150589858	12.20544
45.0	0.9446622804303602	10.674541218935016	150.5945886078944	12.2717
46.0	0.946493353930386	10.447767506782654	145.61155420402483	12.06696
47.0	0.968690355006104	7.186795514646853	85.20522970559553	9.230668

Iteration	R^2	Mean Absolute Error	Mean Squared Error	RMSE
48.0	0.9677465038634836	7.378766903071973	87.77380093757685	9.368767
49.0	0.946317823711565	10.490168453791073	146.08923744245894	12.08674
50.0	0.945410926455777	10.565014394757728	148.5572433560955	12.18841
51.0	0.9707223047313192	6.906313573578722	79.67553612009034	8.926115
52.0	0.9437459057536264	10.786055982880171	153.08838612117538	12.37289
53.0	0.9473330837010708	10.340557274457414	143.32633608622862	11.9719
54.0	0.9422240857922032	10.953009493226672	157.22982622402222	12.53913
55.0	0.9780766721735912	4.970546977933044	59.66155744418119	7.724089
56.0	0.965823545175718	7.728229635687458	93.00688923153136	9.644008
57.0	0.9465771578711024	10.430880965837932	145.3834924032485	12.05751
58.0	0.9699918602787336	6.983781391599809	81.66334809885521	9.036778
59.0	0.9432256818716706	10.869668080030564	154.50410946677803	12.42997
60.0	0.9675333492919967	7.44109577081824	88.3538740821258	9.399674
61.0	0.9698010524426132	7.087261906356872	82.18260743601567	9.065462
62.0	0.9432546065766684	10.844841364264672	154.42539454893156	12.4268
63.0	0.944709929239288	10.689819574875214	150.4649184148697	12.26641
64.0	0.9449085794979192	10.637380077498303	149.92431691903553	12.24436
65.0	0.9699938961762776	7.084714695995746	81.65780766178169	9.036471
66.0	0.9470381533911216	10.43315927089861	144.12895153623575	12.00537
67.0	0.9446930165920684	10.62629156306595	150.51094404024994	12.26829
68.0	0.9439744782522378	10.80443010037735	152.4663550424978	12.34773
69.0	0.971156356925419	6.836967060228443	78.49431809894386	8.859702
70.0	0.9665326337963058	7.600628595806969	91.07719444225494	9.543437
71.0	0.947763712780499	10.287385426858236	142.15443363771007	11.92285
72.0	0.9468465832672822	10.43335397762259	144.6502853427884	12.02706
73.0	0.9718219903997166	6.709823238547304	76.68288098145766	8.756876
74.0	0.9671389256205566	7.479511358440764	89.42724810258935	9.456598
75.0	0.9698974336920227	7.066564034449682	81.92031808406514	9.050984
76.0	0.9662523083898398	7.675919504420153	91.84006449890738	9.583322
77.0	0.9668929760512146	7.561010331246476	90.09656867635692	9.491921
78.0	0.9443582413389444	10.746195145466636	151.42199245193723	12.30536
79.0	0.9449207608112742	10.691859863357596	149.8911670189732	12.243
80.0	0.9435800119287726	10.790699972764727	153.53984513504042	12.39112

Table 5 Subset of the variations of *R* Squared, MAE and MSE w.r.t training iteration for booking stock data

Iteration	R^2	Mean Absolute Error	Mean Squared Error	RMSE
1.0	0.904473168894601	94.77189384457652	19308.578000066365	138.9553
2.0	0.9019995395841656	97.3802934537165	19808.565950374246	140.7429
3.0	0.9029192069645476	96.16348872214292	19622.676089455148	140.081
4.0	0.9014572181931952	98.34954858168337	19918.183894960934	141.1318
5.0	0.9015842597151764	97.67447046785648	19892.505338391675	141.0408
6.0	0.9007347762253178	99.4329553872091	20064.209120814925	141.6482
7.0	0.901618210297254	98.07734583190258	19885.642999773245	141.0165
8.0	0.9016148088585128	97.6001349760327	19886.33052331491	141.0189
9.0	0.9019910248685704	97.36040420265304	19810.287006629496	140.749
10.0	0.902988435789992	95.66526895672348	19608.68305566056	140.031
11.0	0.9022947639760628	97.30240910832026	19748.893048713984	140.5308
12.0	0.9013144752573176	98.45982947905988	19947.03613552476	141.234
13.0	0.9008160764114556	99.31400379399425	20047.77613578555	141.5902
14.0	0.902636699789774	96.47800423701274	19679.778494680744	140.2846
15.0	0.8985528847337011	101.95552679528058	20505.22890097575	143.1965
16.0	0.9006766831662754	99.44642896922424	20075.951312498728	141.6896
17.0	0.9012989072959902	98.1266953139028	19950.18284511528	141.2451
18.0	0.8999877352013833	100.34409833497128	20215.20648682171	142.1802
19.0	0.8997077297576517	100.70451820161364	20271.80322397056	142.3791
20.0	0.8984004464235772	102.13938573788955	20536.04084110992	143.304
21.0	0.8988945981042313	101.55793034127475	20436.159308776365	142.9551
22.0	0.9021456574950945	97.47959522766708	19779.03153530225	140.6379
23.0	0.9010257305894952	98.97901937546504	20005.399308219137	141.4404
24.0	0.900897071233089	99.15753338287824	20031.40487325109	141.5323
25.0	0.9020272621411032	97.6432564689966	19802.962465502023	140.723
26.0	0.9015659907857352	98.41606235748912	19896.1979873052	141.0539
27.0	0.8995345638797079	100.78432081034104	20306.804770892373	142.5019
28.0	0.9028911674416386	95.74501616737116	19628.343641795247	140.1012
29.0	0.9015256483401906	98.24474011555496	19904.35229586438	141.0828
30.0	0.8999979936559085	100.35084038261316	20213.13297336933	142.1729
31.0	0.901273223358203	98.5835464063248	19955.37426945516	141.2635
32.0	0.9000788261373317	100.23912633929268	20196.794524219084	142.1154
33.0	0.89983207219135	100.55275579120982	20246.67022675921	142.2908
34.0	0.9022379142755804	97.35443706451568	19760.38392372096	140.5716
35.0	0.8996031447739338	100.71070853854928	20292.94270166897	142.4533
36.0	0.9016747581567552	98.29814363109834	19874.21313506108	140.9759
37.0	0.9003021671149882	99.9603050477386	20151.65122115134	141.9565
38.0	0.9001606181699455	99.97340347593776	20180.26212360208	142.0572
39.0	0.8998923681194483	100.46555602754654	20234.48277515737	142.248
40.0	0.9002507113070524	100.00501285538134	20162.051843359688	141.9931
41.0	0.8989205993990345	101.63780270204818	20430.90373792785	142.9367
42.0	0.9013254300785022	98.4171325766454	19944.82186737755	141.2261
43.0	0.9002471043503395	99.99395330516124	20162.780907688808	141.9957
44.0	0.9004829030574113	99.67581413424718	20115.11955772957	141.8278
45.0	0.8999160749822327	100.41510143321376	20229.69097159901	142.2311
46.0	0.9001971307844601	100.05726188484776	20172.881928350307	142.0313
47.0	0.9001514966715207	99.92675164744068	20182.105827216823	142.0637
48.0	0.8995207604744246	100.91174068193808	20309.594815579225	142.5117
49.0	0.9008618212924452	99.11743195603572	20038.529847674978	141.5575

Iteration	R^2	Mean Absolute Error	Mean Squared Error	RMSE
50.0	0.8974774234842602	103.15758943619034	20722.6089520105	143.9535
51.0	0.8991755125595203	101.33691486971712	20379.378835599065	142.7564
52.0	0.9001311157567137	100.1479071852964	20186.22536597584	142.0782
53.0	0.9004783215169627	99.6605779333571	20116.045612012826	141.831
54.0	0.90060461454844	99.51682775793176	20090.51834578904	141.741
55.0	0.9001520535790688	100.12600318536172	20181.99326101204	142.0633
56.0	0.8983828737096295	102.04653762129468	20539.59276588335	143.3164
57.0	0.8993487416184864	101.0124108429663	20344.364518069426	142.6337
58.0	0.900892029187926	99.07287497840214	20032.42400809713	141.5359
59.0	0.9005784575176187	99.52511952901628	20095.80539513444	141.7597
60.0	0.8992957684408985	101.0461921723004	20355.071842069723	142.6712
61.0	0.899029028977298	101.4027732673114	20408.987162812893	142.86
62.0	0.8971896790958473	103.47364729257694	20780.76994095422	144.1554
63.0	0.9001114594894006	100.17577764264075	20190.19843371201	142.0922
64.0	0.9005557834750162	99.58697308888854	20100.388437564503	141.7758
65.0	0.8992595435259189	101.2387063721842	20362.39388539866	142.6969
66.0	0.9000347529927889	100.27164298957032	20205.70290879822	142.1468
67.0	0.9004786988693877	99.67558585727772	20115.969338795705	141.8308
68.0	0.8981464699506715	102.42243563162364	20587.376413331207	143.483
69.0	0.8958333918804757	104.86433257814647	21054.912579053474	145.1031
70.0	0.8980664832374817	102.58180187254662	20603.54390965367	143.5393
71.0	0.899503933257743	100.80954220707812	20312.99605502225	142.5237
72.0	0.8997581529142341	100.47548996882408	20261.61132876568	142.3433
73.0	0.9004656054627979	99.67533450044894	20118.615873381863	141.8401
74.0	0.9002234226116497	100.02007908271472	20167.567632982405	142.0126
75.0	0.8998714754039858	100.32852319011526	20238.705762787897	142.2628
76.0	0.897298037120283	103.35709967627722	20758.867828819468	144.0794
77.0	0.8986946855622702	101.81152328808572	20476.567085994386	143.0964
78.0	0.8988227453383181	101.65661204053347	20450.68271250601	143.0059
79.0	0.8978172403536032	102.78495356189852	20653.92269442692	143.7147
80.0	0.898496710975923	102.11217521548537	20516.58314952629	143.2361
81.0	0.9006960225516197	99.37682595000273	20072.042295251154	141.6758

Authors' contributions
Conceptualization, is done by M.S.F. and M.S.T.U and M.S. did methodology, validation formal analysis.M.S.; did investigation and collected data and resourcesM.S.F and M.S wrote the original draft.M.S., and T.U. did review and editing.T.U. did visualization.M.S.F. did the supervision.T.U., and M.S.F did project administrationAll authors have read and agreed to the published version of the manuscript.

Funding
This experimental study did not receive specific funding.

Availability of data and materials
The data used for this research are uploaded to GitHub and can be shared on request. The Fashion MNIST dataset is a publicly available dataset on https://www.kaggle.com/datasets/zalando-research/fashionmnist.

Declarations

Competing interests
The authors declare no competing interests.

Received: 7 January 2024 Accepted: 14 February 2024
Published online: 04 March 2024

References

1. Nishio T, Yonetani R (2019) Client selection for federated learning with heterogeneous resources in mobile edge. ICC 2019 – 2019 IEEE International Conference on Communications (ICC). pp 1–7
2. Manzoor MI, Shaheen M, Khalid H, Anum A, Hussain N, Faheem MR (2018) Requirement Elicitation Methods for Cloud Providers in IT Industry. IJMECS 10(10):40–47
3. McMahan B, Moore E, Ramage D, Hampson S, Arcas BAy (2017) Communication-Efficient Learning of Deep Networks from Decentralized Data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, in Proceedings of Machine Learning Research 54:1273–1282
4. Shaheen M, Farooq MS, Umer T, Kim BS (2022) Applications of federated learning; taxonomy, challenges, and research trends. Electronics 11(4). Available: https://www.mdpi.com/2079-9292/11/4/670

5. Kang J, Xiong Z, Niyato D, Zou Y, Zhang Y, Guizani M (2019) Reliable Federated Learning for Mobile Networks. pp. 1–8. Available: http://arxiv.org/abs/1910.06837
6. Rana N, Marwaha H (2023) Role of federated learning in healthcare systems: A survey. Math Found Comput 1–25
7. Srinivasan K, Prasanna S, Midha R, Mohan S (2023) Federated Learning Framework for IID and Non-IID datasets of Medical Images. EMITTER Int'l J Eng Technol 1–20
8. Zhao Y, Zhao J, Jiang L, Tan R, Niyato D (2019) Mobile Edge Computing, Blockchain and Reputation-based Crowdsourcing IoT Federated Learning: A Secure, Decentralized and Privacy-preserving System. pp. 1–7. Available: http://arxiv.org/abs/1906.10893
9. Mehta S, Kukreja V, Gupta A (2023) Transforming Agriculture: Federated Learning CNNs for Wheat Disease Severity Assessment. In: 2023 8th International Conference on Communication and Electronics Systems (ICCES). IEEE. pp. 792–797. https://doi.org/10.1109/ICCES57224.2023.10192885
10. Mehta S, Kukreja V, Yadav R (2023) Advanced Mango Leaf Disease Detection and Severity Analysis with Federated Learning and CNN. In: 2023 3rd International Conference on Intelligent Technologies (CONIT). IEEE, Hubli, 23-25 June 2023, 1–6. https://doi.org/10.1109/CONIT59222.2023.10205922
11. Pokhrel S, Choi J (2020) Federated Learning With Blockchain for Autonomous Vehicles: Analysis and Design. IEEE Transactions on Communications. PP. https://doi.org/10.1109/TCOMM.2020.2990686
12. Shaheen M, Farooq MS, Umer T (2024) Reduction in data imbalance for client-side training in federated learning for the prediction of stock market prices. J Sens Actuator Netw 13:1. https://doi.org/10.3390/jsan13010001
13. Farooq MS, Tehseen R, Qureshi JN, Omer U, Yaqoob R, Tanweer HA, Atal Z (2023) FFM: Flood forecasting model using federated learning. IEEE Access 11:24472–24483
14. Silva S, Gutman BA, Romero E, Thompson PM, Altmann A, Lorenzi M (2019) Federated learning in distributed medical databases: Meta-analysis of large-scale subcortical brain data. Proc Int Symp Biomed Imaging 2019:270–274
15. Li S, Lv L, Li X, Ding Z (2021) Mobile app start-up prediction based on federated learning and attributed heterogeneous network embedding. Future Internet 13(10):256
16. Alsamhi SH, Shvetsov AV, Kumar S, Hassan J, Alhartomi MA, Shvetsova SV, Sahal R, Hawbani A (2022) Computing in the Sky: A Survey on Intelligent Ubiquitous Computing for UAV-Assisted 6G Networks and Industry 4.0/5.0. Drones 6:177. https://doi.org/10.3390/drones6070177
17. Shoham N, Avidor T, Keren A, Israel N, Benditkis D, Mor-Yosef L, Zeitak I (2019) Overcoming forgetting in federated learning on non-IID data. pp. 1–6. Available: http://arxiv.org/abs/1910.07796
18. Liu Y, Ma Z, Liu X, Ma S, Nepal S, Deng R (2019) Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing. arXiv preprint arXiv:1907.10218
19. K Bonawitz et al (2019) Towards Federated Learning at Scale: System Design
20. Guendouzi BS, Ouchani S, Assaad HE, Zaher ME (2023) A systematic review of federated learning: Challenges, aggregation methods, and development tools. J Netw Comput Appl 103714. https://doi.org/10.1016/j.jnca.2023.103714
21. Nishio T, Yonetani R (2019) Client selection for federated learning with heterogeneous resources in mobile edge. In: ICC 2019-2019 IEEE international conference on communications (ICC). IEEE. p. 1–7
22. K Bonawitz et al (2017) Practical secure aggregation for privacy-preserving machine learning. Proceedings of the ACM Conference on Computer and Communications Security. pp 1175–1191
23. Brecko A, Kajati E, Koziorek J, Zolotova I (2022) Federated Learning for Edge Computing: A Survey. Appl Sci 12:9124. https://doi.org/10.3390/app12189124
24. Shaheen M, Farooq MS, Umer T, Kim B-S (2022) Applications of federated learning; taxonomy, challenges, and research trends. Electronics 11:670. https://doi.org/10.3390/electronics11040670
25. Lim WYB, Luong NC, Hoang DT, Jiao Y, Liang YC, Yang Q, Miao C (2020) Federated learning in mobile edge networks: A comprehensive survey. IEEE Communications Surveys & Tutorials 22(3):2031–2063
26. Li X, Huang K, Yang W, Wang S, Zhang Z (2019) On the Convergence of FedAvg on Non-IID Data, 2019th edn. pp 1–26. Available: http://arxiv.org/abs/1907.02189
27. Duan M, Liu D, Chen X, Tan Y, Ren J, Qiao L, Liang L (2019) Astraea: Selfbalancing federated learning for improving classification accuracy of mobile deep learning applications," Proceedings – 2019 IEEE International Conference on Computer Design, ICCD 2019. pp. 246–254
28. Rahman A, Hossain MS, Muhammad G, Kundu D, Debnath T, Rahman M, ... Band SS (2023) Federated learning-based AI approaches in smart healthcare: concepts, taxonomies, challenges and open issues. Cluster Comput 26(4):2271–2311
29. Subramanian M, Rajasekar V, VE, S., Shanmugavadivel, K., & Nandhini, P. S. (2022) Effectiveness of decentralized federated learning algorithms in healthcare: a case study on cancer classification. Electronics 11(24):4117
30. Cremonesi F, Vesin M, Cansiz S, Bouillard Y, Balelli I, Innocenti L, ... Lorenzi M (2023) Fed-BioMed: Open, Transparent and Trusted Federated Learning for Real-world Healthcare Applications. arXiv preprint arXiv:2304.12012
31. Farooq, Muhammad Shoaib, Hafiz Ali Younas (2023) Beta Thalassemia Carriers detection empowered federated Learning. arXiv preprint arXiv:2306.01818
32. Berghout T, Benbouzid M, Bentrcia T, Lim WH, Amirat Y (2023) Federated learning for condition monitoring of industrial processes: a review on fault diagnosis methods, challenges, and prospects. Electronics 12:158. https://doi.org/10.3390/electronics12010158
33. Zhao Y, Li M, Lai L, Suda N (2018) D Civin. V Chandra, Federated learning with non-iid data
34. Dinh CT, Tran NH, Nguyen MN, Hong CS, Bao W, Zomaya AY, Gramoli V (2020) Federated learning over wireless networks: Convergence analysis and resource allocation. IEEE/ACM Trans Networking 29(1):398–409
35. Wang L, Wang W, Li B (2019) CMFL: Mitigating Communication Overhead for Federated Learning. 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS) 954–964. https://api.semanticscholar.org/CorpusID:204781679
36. Konečný J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D (2016) Fed-´ erated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492
37. Verma DC, White G, Julier S, Pasteris S, Chakraborty S, Cirincione G (2019) Approaches to address the data skew problem in federated learning. In Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, vol. 11006. International Society for Optics and Photonics. p. 110061I
38. Stich SU, Cordonnier JB, Jaggi M (2018) Sparsified SGD with memory. 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada, Advances in Neural Information Processing Systems 31
39. Wang H, Sievert S, Liu S, Charles Z, Papailiopoulos D, Wright S (2018) Atomo: Communication-efficient learning via atomic sparsification. Advances in neural information processing systems 31
40. Lim WYB, Luong NC, Hoang DT, Jiao Y, Liang YC, Yang Q, Niyato D, Miao C (2020) Federated learning in mobile edge networks: A comprehensive survey. IEEE Commun Surv Tutor 22(3):2031–2063
41. Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V (2020) Federated optimization in heterogeneous networks
42. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: Challenges, methods, and future directions". IEEE Signal Proc Mag 37(3):50–60. Available: https://doi.org/10.1109/MSP.2020.2975749
43. Wan L, Zeiler M, Zhang S, Le Cun Y, Fergus R (2013) Regularization of Neural Networks using DropConnect. Proceedings of the 30th International Conference on Machine Learning, in Proceedings of Machine Learning Research 28(3):1058–1066. Available from: https://proceedings.mlr.press/v28/wan13.html
44. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. pp 785–794
45. Han H, Wang WY, Mao BH (2005) Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: International conference on intelligent computing. Springer, pp. 878–887. https://api.semanticscholar.org/CorpusID:12126950
46. Nilsson A, Smith S, Ulm G, Gustavsson E, Jirstrand M (2018) A performance evaluation of federated learning algorithms. Proceedings of the second workshop on distributed infrastructures for deep learning. pp 1–8

47. Wang H, Wu Z, Xing E P (2018) Removing confounding factors associated weights in deep neural networks improves the prediction accuracy for healthcare applications. In: BIOCOMPUTING 2019: Proceedings of the Pacific Symposium. pp. 54–65
48. Xu J, Glicksberg BS, Su C, Walker P, Bian J, Wang F (2021) Federated learning for healthcare informatics. J Healthc Inform Res 5(1):1–19
49. Sattler F, Muller K-R, Samek W (2020) Clustered federated learning: Model-agnostic¨ distributed multitask optimization under privacy constraints. xIEEE Trans Neural Netw Learn Syst 32(8):3710–3722
50. Yang Q, Liu Y, Chen T, Tong Y (2019) Federated machine learning: Concept and applications. ACM Trans Intell Syst Technol 10(2):1–19
51. Lim WYB, Luong NC, Hoang DT, Jiao Y, Liang YC, Yang Q, Niyato D, Miao, (2020) Federated learning in mobile edge networks: A comprehensive survey. IEEE Commun Surv Tutor 22(3):2031–2063
52. Mohri M, Sivek G, Suresh AT (2019) Agnostic Federated Learning. Proceedings of the 36th International Conference on Machine Learning, in Proceedings of Machine Learning Research 97:4615–4625 Available from: https://proceedings.mlr.press/v97/mohri19a.html
53. Bonawitz K, Eichner H, Grieskamp W, Huba D, Ingerman A, Ivanov V, Kiddon C, Konečný J, Mazzocchi S, McMahan B et al (2019) Towards federated learning at scale: System design. Proc Mach Learn Syst 1:374–388
54. Abadi M, Chu A, IGoodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 308–318
55. Dwork C, Roth A et al (2014) The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 9(3–4):211–407
56. Abay NC, Zhou Y, Kantarcioglu M, Thuraisingham B, Sweeney L (2018) Privacy preserving synthetic data release using deep learning. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, pp. 510–526. https://doi.org/10.1007/978-3-030-10925-7_31
57. Dwork C, McSherry F, Nissim K, Smith A (2016) Calibrating noise to sensitivity in private data analysis. J Priv Confidentiality 7(3):17–51
58. Augenstein S, McMahan HB, Ramage D, Ramaswamy S, Kairouz P, Chen M, Mathews R et al (2019) Generative models for effective ml on private, decentralized datasets. arXiv preprint rXiv:1911.06679
59. Wong SC, Gatt A, Stamatescu V, McDonnell MD (2016) "Understanding data augmentation for classification: when to warp?", 2016 international conference on digital image computing: techniques and applications (DICTA). IEEE, pp. 1–6
60. Pourroostaei Ardakani S, Du N, Lin C, et al (2023) A federated learning-enabled predictive analysis to forecast stock market trends. J Ambient Intell Human Comput 14:4529–4535. https://doi.org/10.1007/s12652-023-04570-4
61. Menegatti D, Ciccarelli E, Viscione M, Giuseppi A (2023) "Vertically-Advised Federated Learning for Multi-Strategic Stock Predictions through Stochastic Attention-based LSTM.", 2023 31st Mediterranean Conference on Control and Automation (MED). IEEE, pp. 521–528. https://doi.org/10.1109/MED59994.2023.10185757
62. Sakhare NN, Shaik IS (2024) Spatial federated learning approach for the sentiment analysis of stock news stored on blockchain. Spat Inf Res 32(1):13–27
63. Montgomery DC, Peck EA, Vining GG (2021) Introduction to linear regression analysis. John Wiley & Sons
64. Li X, Huang K, Yang W, Wang S, Zhang Z (2019) On the convergence of fedavg on non-iid data. arXiv preprint arXiv:1907.02189
65. Zhou Y, Ye Q, Lv J (2021) Communication-efficient federated learning with compensated overlap-fedavg. IEEE Trans Parallel Distrib Syst 33(1):192–205
66. Su X, Yan X, Tsai CL (2012) Linear regression. Wiley Interdiscip Rev Comput Stat 4(3):275–294
67. Shaheen M, Saif U, Awan SM, Ahmad F, Anum A (2023) Classification of images of skin lesion using deep learning. IJISAE 13(2):23
68. Shaheen M, Awan SM, Hussain N, Gondal ZA (2019) Sentiment analysis on mobile phone reviews using supervised learning techniques. IJMECS 11(7):32
69. Ahmad F, Najam A (2012) Video-based face classification approach: A survey. 2012 International Conference on Robotics and Artificial Intelligence, ICRAI 2012. p. 179–186. https://doi.org/10.1109/ICRAI.2012.6413396
70. Ahmad F, Najam A, Ahmed Z (2013) Image-based face detection and recognition:" state of the art". arXiv preprint arXiv:1302.6379
71. Ahmad F, Ahmed Z, Najam A (2013) Soft biometric gender classification using face for real time surveillance in cross dataset environment. 2013 16th International Multi Topic Conference, INMIC 2013, p. 131–135. https://doi.org/10.1109/INMIC.2013.6731338
72. Bejjanki KK, Gyani J, Gugulothu N (2020) Class imbalance reduction (CIR): a novel approach to software defect prediction in the presence of class imbalance. Symmetry 12(3):407
73. Anand R, Mehrotra KG, Mohan CK, Ranka S (1993) An improved algorithm for neural network classification of imbalanced training sets. IEEE Transactions on Neural Networks 4(6):962–969
74. Yang M, Wang X, Zhu H, Wang H, Qian H (2021) "Federated learning with class imbalance reduction". 2021 29th European Signal Processing Conference (EUSIPCO). IEEE, pp. 2174–2178
75. Seol M, Kim T (2023) Performance Enhancement in Federated Learning by Reducing Class Imbalance of Non-IID Data. Sensors 23:1152. https://doi.org/10.3390/s23031152. Accessed 22 Apr 2023
76. Van Dyk DA, Meng XL (2001) The art of data augmentation. J Comput Graph Stat 10(1):1–50
77. Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. J Big Data 6(1):1–48
78. Keskar NS, Socher R (2017) Improving generalization performance by switching from adam to sgd. arXiv preprint arXiv:1712.07628. Accessed 6 Aug 2022
79. Chandiramani K, Garg D, Maheswari N (2019) Performance analysis of distributed and federated learning models on private data. Procedia Computer Science 165:349–355
80. Flower (2022) Flower a friendly federated learning framework, 2022. https://fower.dev/. Accessed 12 Aug 2022

Publisher's Note