Cureus Journal of Computer Science

Open Access Original Article

A Federated Learning (FL) Platform to Train the Machine Learning Model: A Step Towards Making FL More Efficient

Received 01/23/2025
Review began 01/24/2025
Review ended 08/15/2025
Published 08/17/2025

DOI:
https://doi.org/10.7759/s44389-024-01061-1

Gayatri M. Bhandari [1] , Nitin M. Shivale [1] , Shrishail S. Patil [2] , Pranav Prajapati [1] , Atharv Burujpatte [1] , Diya Gandhi [1] , Mangal Kasare [1]

1. Computer Engineering, JSPM's Bhivarabai Sawant Institute of Technology and Research, Pune, IND 2. Computer Enigineerng, Nirwan University, Jaipur, IND

Corresponding author: Gayatri M. Bhandari, gayatri.bhandari1980@gmail.com

Abstract

Federated learning is an emerging technology that can revolutionize the training of machine learning models. Federated learning refers to an approach to training a machine learning model in a decentralized and collaborative fashion. A central server distributes the model to client devices, where it is trained locally using the clients' own data. The client then sends the updated model weights to the server, which aggregates them to update the global model. This paper introduces a federated learning platform designed to enable collaborative training of machine learning models across multiple client devices while preserving data privacy. The platform supports a range of supervised learning algorithms, including convolutional neural networks and decision trees, and is compatible with widely used frameworks such as TensorFlow, PyTorch, and Flower. It offers a user-friendly interface where model developers can upload or deploy their machine learning models to a central server. Clients can then access these models and train them locally using their own data. The platform's modular design ensures flexibility in deployment and efficiency in handling real-world applications. The key features of this application include a model repository, secure API access for client integration, local model training capabilities on user-end devices, and a user-friendly UI. The platform aims to democratize machine learning by enabling distributed model training and deployment, promoting collaboration and efficiency across diverse use cases. The scalable infrastructure supports real-time inference, on-device training, and secure data handling, making it ideal for industries ranging from healthcare to finance and beyond.

Categories: AI applications, Data Security, Machine Learning (ML)
Keywords: federated learning, machine learning, data privacy, artificial intelligence, data security

Introduction

With the rise of artificial intelligence, machine learning has become a principal pillar of the modern technological era. Its presence spans across different fields - from agriculture and healthcare to finance and space research. The demand for machine learning and its applications is growing rapidly in areas such as the Internet of Things, natural language processing, pattern recognition, and image classification. One of the key aspects of machine learning is the training process, which typically requires large volumes of data - often collected and stored in a centralized manner. In today's world, data has become a major asset, and therefore, the security and privacy of data have become crucial. In this new era, where technology is rapidly evolving, one thing that will always remain at the forefront of consideration is data security.

Federated Learning (FL) has been establishing its foot in this new era of technology. FL uses distributed clients to train a machine learning model collaboratively via robust communication with the server. FL was first introduced by Google researchers [1]. Their paper laid the foundation of FL. The paper proposed a method for training machine learning models in a decentralized way, where the data stays on local devices and only model updates/weights are shared with a central server [1-5].

The development of FL aggregates model updates from the clients and performs aggregation. Initially, FL was used in mobile devices, but now it can be seen in different fields like healthcare and finance, where data privacy and security are very important. FL helps the model generalize well, which is crucial in areas with very strict data protection laws because even though it does not share raw data between different organizations or devices, it can train the model collaboratively. FL emerges as one of the vital technologies in the context of data privacy and security [6-9].

FL is introduced as the most fundamental approach to learning distributed machine learning. In FL, the model is trained over multiple decentralized devices to not share any user's private data. Whereas in traditional machine learning methods, the data must be sent to a central server where the training takes place, FL aims to keep the data on the client side and share only model updates, like gradients or weights, with a central server. It fundamentally addresses the issues that arise out of concerns about data security,

privacy, and compliance with regulations such as GDPR, wherein legal restrictions limit the sharing of sensitive information across borders or entities. The use of mobile devices further increases the demand for such decentralized learning techniques, Internet of Things (IoT) networks, and edge computing environments [9-10].

Heterogeneity

First and foremost, FL poses several technical challenges. One of the primary challenges in FL is data heterogeneity. Unlike centralized machine learning, where data can be curated and standardized, FL must work with data distributed across clients - each with unique usage patterns and environments. For example, users from different regions may interact with an application differently, leading to non-independent and identically distributed (non-IID) data, which complicates the training process. [11]. This could slow the convergence of the global model or even make it sub-optimal, as models that some clients update are different from those of others. Aggregation of such non-IID data has to be modelled carefully within aggregation techniques so that biases do not steer the global model toward any one particular client or subset of data [12-14].

Speaking of heterogeneity, another major bottleneck in FL systems is the communication overhead between the client and the server. Frequent exchange of model parameters between clients and the server - especially in large-scale deployments - increases communication overhead, often leading to bandwidth constraints and latency issues. These challenges are particularly evident in edge computing environments, where limited network resources can significantly impact training efficiency. Especially when working in resource-constrained or poorly connected environments, such as mobile devices or IoT sensors, model communication becomes a luxury [14]. Over the years, several methodologies have been adapted, from model compression to quantization and reducing the number of communication rounds to using trade-offs concerning model accuracy and efficiency in [9-14], making FL systems scalable and efficient in differently networked environments.

Security

Equally pressing is the security around model updates. Although FL, by design, inherently respects privacy, as the data is always local, sending the model updates is not immune to attacks. Such vulnerabilities can be exploited to attempt to reverse the shared model updates and, from there, breach privacy with model inversion and membership inference attacks [10,13]. That is why, along with simpler measures, very advanced privacy-preserving technologies such as differential privacy and homomorphic encryption are used in FL systems. The former adds random noise to model updates in such a way that isolation of individual elements is not possible [13], while the latter enables computation over encrypted data [13]. While these provide very strong assurances for privacy, they come at the price of additional computational overhead and may even result in a hit to the model's performance.

Likewise, the diversity of client devices in the FL approach introduces challenges. In practical deployments, clients typically include everything from high-end servers to mobile phones or even IoT devices, which can vary in computing capacity, network reliability, and battery usage. Devices with limited resources can be unable to handle the computations required for the training of the model or for maintaining a stable connection needed to obtain model updates. In such scenarios, models can still be trained using strategies like client selection (i.e., not including all clients in every single training round) and asynchronous training, where all devices are trained simultaneously, irrespective of time. This can improve model convergence and training stability across heterogeneous client data. Furthermore, lightweight model architectures or model distillation approaches can be implemented to ensure that low-computing devices can still participate in building the overall model [9].

FL represents a transformative shift in the way machine learning models are trained, offering a privacy-preserving alternative to centralized systems. However, the implementation of FL systems is not without challenges, particularly in terms of communication overhead, data heterogeneity, and security.

FL: A new approach to train machine learning models

FL is a new approach that provides a secure way of training machine learning models on multiple nodes under the context that the data should not be shared and should remain local to the nodes or the clients. In the whole process of FL for training machine learning models, different clients or nodes take part in training the model locally, and a server sends the model to the client and performs aggregation of the updates received from the clients [2-12]. Figure *1* shows the abstract view of FL.

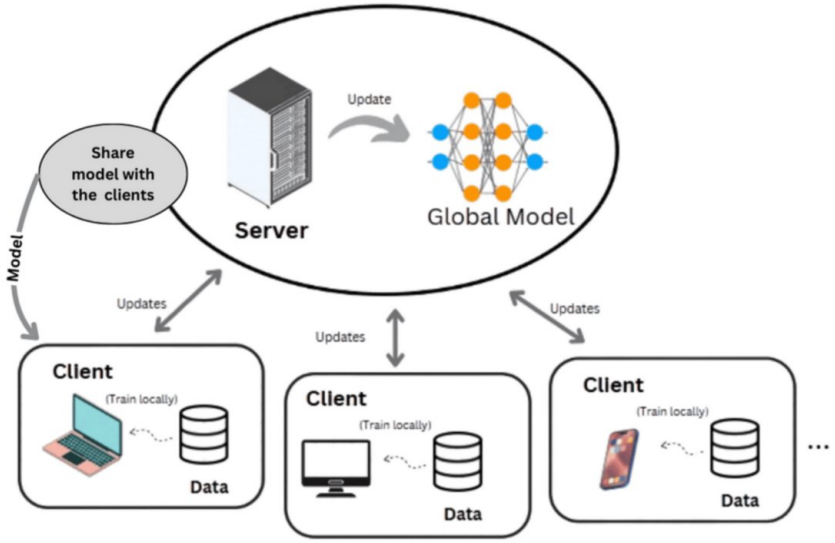

FIGURE 1: Federated learning

FL employs various techniques to create a secure environment, maintain the privacy of the client's data, and train the machine learning model securely and collaboratively. It encompasses different approaches throughout the learning process, such as sharing models to aggregate the updates/weights for system efficiency and model accuracy [10-13]. FL is constantly improving and reforming day by day since its use case can be seen in different fields like intrusion detection, smart city, finance, and healthcare [9-14].

FL can play a crucial role in today's world, where data security and privacy are the need of the hour. The ubiquitous declaration that "Artificial Intelligence is the future of technology" is coupled with the major challenge of ensuring data security and privacy. In this context, FL has the potential to revolutionize fields such as machine learning, data science, and artificial intelligence [8-12]. FL can play a major role in ensuring security and privacy, making these fields spread their magic everywhere. This can result in bringing the best out of this whole scenario of technology and make it reach its apogee, making the task not just easier but also secure.

Traditional approach of machine learning vs. FL approach

Table *1* represent the comparison between the traditional machine learning approach and FL approach.

Aspect	Traditional Machine Learning	Federated Learning
Data Handling	Centralized: All data is collected and stored in a central server for training	Decentralized: Data remains on individual devices, and only model updates are shared
Privacy	Low: Sensitive data is transferred to a central location, raising privacy concerns	High: Data never leaves the device, ensuring user privacy
Scalability	Limited: Requires transferring large volumes of data to the central server, which may be inefficient for large-scale systems	High: Scales well with distributed clients as only updates are communicated
Infrastructure	Relies on a powerful central server for both data storage and processing	Relies on distributed client devices for training, reducing server-side computational load
Communication Overhead	Low: Limited communication since the data is directly accessible in a centralized location	High: Frequent communication is required to exchange model updates between clients and the server.
Energy and Resource Usage	Centralized servers consume significant resources for storage and computation	Distributes the computation across clients, leveraging local device resources
Data Ownership	Relinquished: Users must share their data with the central authority	Retained: Users retain control and ownership of their data
Model Performance	Can achieve high performance if large and diverse datasets are collected centrally	Performance depends on data heterogeneity and aggregation techniques used
Risk of Data Breaches	High: A centralized database is a single point of failure, making it vulnerable to attacks	Reduced: Data breaches are minimized as raw data is never centralized
Training Time	Centralized and typically faster due to access to all data at once	Potentially slower due to distributed computation and network latency
Personalization	Limited: Models are generally global and may not adapt to individual user needs	High: Models can be personalized for individual client devices while maintaining a global model

TABLE 1: Traditional machine learning vs federated learning

Process of FL

FL entails different steps, each and every one having their own specific task. All these steps need to be executed properly, thus ensuring proper execution of FL. These steps can be seen in Figure *2*.

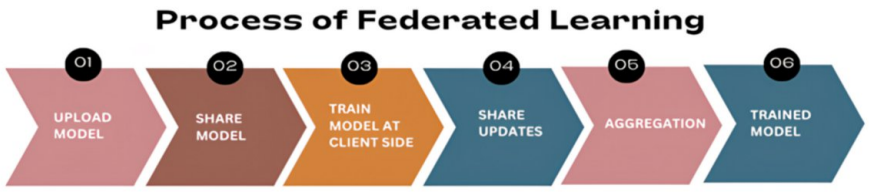

FIGURE 2: Process of federated learning

1. Upload Model: First, the model is uploaded to the server. The central server initializes a global model with predefined parameters. This model acts as the baseline for training across all participating clients.

2. Share Model With Client: The global model is distributed to selected clients. Each client receives a copy of the model to perform local training on its private data.

3. Training Model at Client Side: Clients independently train the global model using their local datasets. They apply standard optimization methods, like stochastic gradient descent, to compute updates based on their unique data.

4. Share Updates: After completing local training, clients send only the computed model updates (e.g.,

weights or gradients) back to the central server. No raw data is shared, preserving privacy.

5. Aggregation: The server collects updates from all clients and combines them to improve the global model. This aggregation is commonly done using weighted averaging, where contributions are proportional to the size of each client's dataset. Then, the global model's weights are updated with these aggregate weights.

6. Trained Model: Once the global model achieves the desired accuracy, it is finalized and made available for deployment in real-world applications.

Significance of FL

A Paradigm Shift in AI Training: FL transforms machine learning by addressing privacy challenges and enabling decentralized collaboration without compromising data security.

Cross-Industry Versatility: The platform has proven applications in multiple domains, including finance (risk management, anti-money laundering), healthcare (secure diagnostics), and IoT (real-time edge computing) [8,9].

Ethical and Trustworthy AI: By emphasizing data privacy, FL fosters trust and ensures compliance with ethical standards in AI development [6,10].

Accelerating Innovation: With privacy-preserving collaboration, FL drives the advancement of machine learning technologies, enabling secure and scalable AI solutions [3,7].

Proposed platform and objectives

FL is evolving steadily. Although, with different applications in different domains, a common platform that makes the process of training machine learning models with FL techniques easy has become a necessity. The proposed platform will encapsulate most of the major concerns of communication overheads, data transmission, security, privacy, and aggregation into one consistent "umbrella" in order to better extend the applicability of FL to multivaried domains. However, FL is not without its challenges, mainly coming in the form of:

Communication Overhead: Efficient client-server communication is handled through asynchronous updates and model compression techniques to reduce bandwidth usage and latency.

Client-Side Training: The platform provides automated local training scripts, minimizing user intervention and ensuring consistency in model updates across devices.

Model Aggregation: A secure aggregation strategy, such as Federated Averaging (FedAvg), is implemented to combine client updates. The platform also supports differential privacy and secure multiparty computation to protect individual client contributions during aggregation.

Scalability: To support a growing number of clients, the platform uses a modular architecture with dynamic client selection and resource-aware scheduling to optimize performance without overwhelming system resources.

This paper proposes an extensive platform to tackle these challenges and makes FL accessible to everyone from beginners in development to experienced industry professionals.

Objectives of the Proposed Platform

Data Privacy: Client data is kept private on local devices. Strict privacy rules are maintained; the disclosure of sensitive data is prevented. Model updates are communicated to the server only, protecting the information from being shared.

Security Connection: This prevents an unreliable and incorrect update from entering the system. This means that updating exchanges between servers and clients use encrypted channels, so they are both confidential and integrity-compliant.

Efficient FL Coordination: This platform manages all the phases involved in FL - beginning with the distribution of models, aggregation of updates, and so on. Well-structured aggregation algorithm ensures the global model update will neither be biased nor erroneous.

Performance Measurement: The platform includes real-time monitoring tools that track key performance indicators tailored to FL. These include global model accuracy, round-wise training loss,

number of active participating clients, and communication time per round. These metrics are visualized through a user-friendly interface, allowing users to observe training progress, identify convergence trends, and assess client participation levels.

Reporting: It reports the progress during the training and produces final comprehensive reports at the end, summarizing model accuracy and other evaluation metrics. It ensures transparency and informed decision-making.

Addressing these critical challenges makes the proposed platform simplify how FL is adopted and implemented, which has now sent it into applications across various industries, making it one of the inevitable tools for today's data-driven world.

Materials And Methods

The proposed system for implementing FL comprises a central server and multiple clients that collaboratively train a machine learning model. The methodology is designed to ensure data privacy, secure communication, and efficient training through a decentralized approach. Figure *3* represent the block diagram of the whole process.

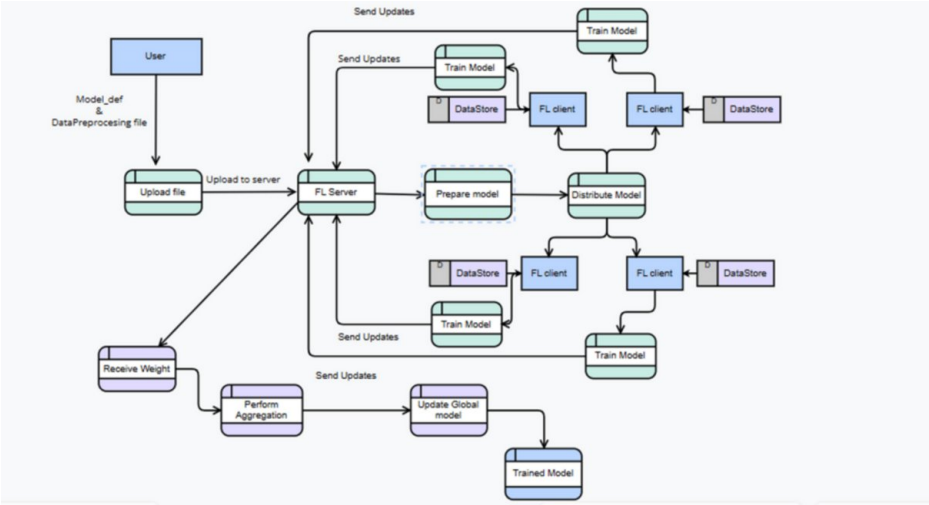

FIGURE 3: Block diagram

Below, we describe the step-by-step process of the FL workflow:

Server Initialization: The central server initializes a global model with predefined parameters (e.g., weights, structure). Additionally, the server generates a data preprocessing and training script that is to be shared with the clients. For confidential communication, the server generates a symmetric encryption key using the Fernet encryption key, which encrypts the model and scripts.

Distribution of Global Model: Clients securely fetch the encrypted global model and the corresponding preprocessing/training script from the server using a WebSocket-based communication protocol. This approach enables persistent, bidirectional connections that reduce the overhead of repeated HTTP handshakes, making it more efficient for the iterative communication required in FL. Compared to traditional RESTful APIs or HTTP polling, WebSockets offer lower latency and improved bandwidth utilization, particularly in environments with frequent model updates and asynchronous client participation. This makes WebSockets well-suited for FL workloads, where timely delivery and minimal communication delay are critical for maintaining training efficiency across distributed clients.

Local Model Training at Client Side: Each client decrypts the received global model and preprocessing script using the shared symmetric encryption key. Once decrypted, clients process their local datasets and perform training using data that may follow non-identical distributions (non-IID). To address the challenges posed by non-IID data, the platform supports strategies such as data reweighting, personalized model layers, and local fine-tuning. These approaches help maintain model generalizability while respecting the diversity in client data.

Local training is performed using configurable hyperparameters defined in the shared script, including batch size, number of local epochs, learning rate, and optimizer type. These parameters are selected to balance training stability with communication efficiency, and can be adapted dynamically based on client

device capabilities or prior training performance. After local training is completed, each client produces an updated set of model weights, which are prepared for secure transmission back to the server.

Sending Updates to the Server: The clients serialize and send encrypted updated model weights back to the server over WebSocket. It guarantees that all updates are only sent encrypted back to the server and hence protects client data's privacy.

Aggregating Updates on the Server: The server decrypts the received model weights from all participating clients and performs aggregation using the Federated Averaging (FedAvg) algorithm. In FedAvg, the global model is updated by computing a weighted average of the locally trained models, with each client's contribution proportional to the size of its local dataset. The aggregation is calculated as:

$$w_t = \frac{n_1}{n} \times w_1 + \frac{n_2}{n} \times w_2 + \cdots + \frac{n_k}{n} \times w_k$$

where:

w_t is the updated global model after round *t*.

w_k is the local model from client *k*.

n_k is the number of data samples at client *k*.

■ is the total number of samples across all participating clients.

Iterative Training: After each round of training, the updated global model is shared with clients to continue the next round of local training. This cycle repeats until a stopping condition is reached - either a set number of communication rounds or when the model's performance stabilizes. Convergence is typically determined by tracking improvements in accuracy or loss across rounds. For example, if there's less than a 0.5% accuracy gain over three consecutive rounds, the system concludes that further training would bring minimal benefit.

Once training is complete, the final version of the global model is sent to the clients for evaluation on their local data. This step helps measure how well the model performs across different real-world scenarios, especially considering variations in client data. The clients report back key performance metrics like accuracy or loss, which the server then aggregates to provide a clear picture of the model's overall effectiveness.

Finalizing the Global Model: After convergence, the model is ready for deployment. Clients evaluate the final global model on their local data and return performance metrics like accuracy and loss to ensure it performs well across diverse conditions.

The trained model is then saved in an appropriate format - such as .pkl for PyTorch or .h5 for TensorFlow - based on the framework used. This saved model can then be used for deployment in real-world applications or further offline analysis. Once saved, temporary files and training scripts are removed, marking the completion of the training process.

Functionality and modules

The concept behind the proposed platform is to formalize the FL process in a unified manner in line with the principles of FL. The FL technique, in the context of training the machine learning model, includes several steps like communication set-up, data sharing, etc. About the proposed platform, all these steps can be broken down into systematic modules. Every module has a crucial role, and these have to be handled correctly to achieve the desired goal. These modules are described below:

Server Communication Establishment

The central idea of FL is to learn the model in a distributed and collaborative manner. Thus, one of the major tasks here is networking. A reliable, private communication must be established between client and server in order to exchange not only the model from the server side to the client side but also model updates from the client side to the server side. For implementing this module, a properly suited communication protocol should be used, for example, WebSockets or gRPC. Since the communication is handled within our platform, we use Socket.IO instead of raw WebSockets. Socket.IO provides built-in support for features like automatic reconnection, message acknowledgments, and real-time event handling, which are essential for maintaining a reliable connection between clients and the server during the federated training process.

According to the needs of the proposed platform, the websockets protocol can be applied to achieve communication. Based on the platform's requirements, symmetric key encryption is used to ensure secure communication between the server and clients during the training process. However, this key is not used for authentication. Instead, authentication relies on a unique model ID and a secret access key, both of which are shared securely by the model owner with the participating clients - typically via email or other controlled channels. Proper key distribution ensures that only authorized clients can access and train the model, and key rotation policies can be implemented to maintain long-term security.

Sending the Model to the Client for Training

As per the proposed platform, the model definition and data preprocessing and training function (separate files for the model definition and data preprocessing and training function) will be given as input to the server. The server will take the above-mentioned input. The server will build the input model instance and then will distribute it to all the participants clients with an objective of model confidentiality. And this, in turn, will also deliver the file with data preprocessing and training functions. For secure transmission, all the data will be encrypted. The whole process can be seen in Figure *4*.

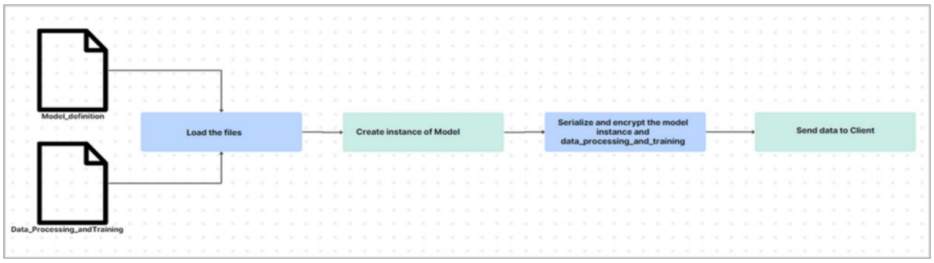

FIGURE 4: Sending model from server to the clients

Model Training at Client Side

Once the client receives the encrypted global model and training script from the server, it separates and stores the files in temporary memory. Before training begins, the client automatically installs any required dependencies specified in the training script to ensure compatibility. The client preprocesses its local data - normalizing, resizing, or encoding as needed - to match the global model's expected input format, ensuring consistent training across clients. The model is then trained on the client's private dataset. Before sending updates back to the server, the client also validates the trained model locally using a subset of its data. This ensures that the updates being contributed are meaningful and not corrupted.

To conserve device storage and maintain cleanliness, the client can delete all temporary data and installed dependencies immediately after training concludes. The client then sends the updated model weights back to the server.

Figure *5* illustrates the overall client-side training workflow.

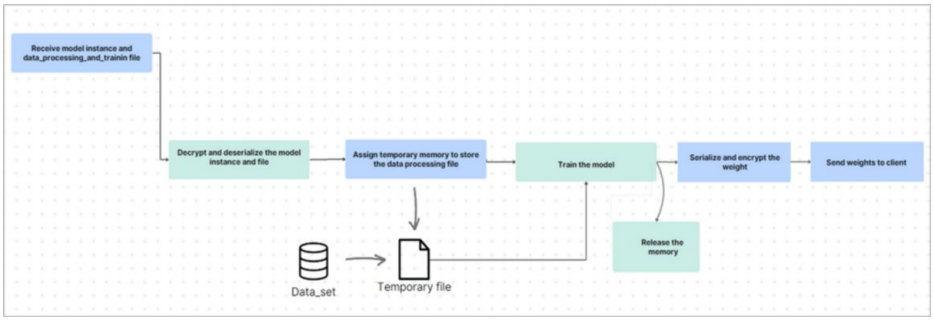

FIGURE 5: Model training at client side

Receiving Weigths and Aggregation

As per the proposed idea, after the clients have trained the model and sent model updates back to the server. The server will now receive all the weights and will store them in a separate file, considering the risk of losing received weights. The server will carry out aggregation after receiving weights from each

client.

Figure 6 shows the server receiving weights from the clients and storing them.

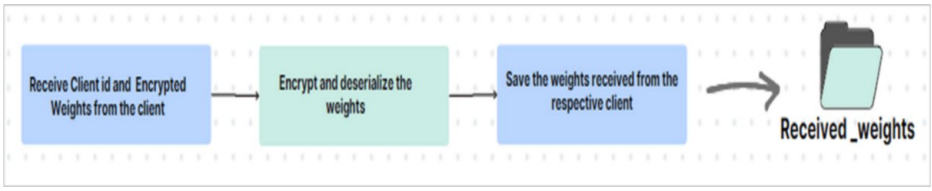

FIGURE 6: Receiving weights from the clients

The server will aggregate the updates/weights and update the global model, i.e., the actual model. Figure 7 represents the aggregation performed by the server.

FIGURE 7: Aggregation

Aggregation plays a very critical role in FL. It significantly affects the accuracy of the model. It is very important to select the best-suited averaging algorithm for the model.

Model Performance Tracking

One of the major aspects of any machine learning model is its accuracy. The accuracy of the model significantly depends upon the data that has been used to train the model and how the model has been created (for example, how the neural network model has been created). The proposed platform also includes the performance and accuracy tracking of the model. There are many methods using which the model's performance can be tracked. In the context of FL, one of the methods is evaluation metrics, which evaluate the effectiveness of the federated system.

• Performance Metrics for Model Inference and Training

• Client-Side Resource Utilization (Memory, CPU, GPU)

• User Experience and Platform Usability

Server-client interaction

Figure 8 represents the proposed server-client interaction throughout the process of FL for training the machine learning model. It gives an abstract view of how the server and client will interact for sending models, files, and updates/weights, respectively.

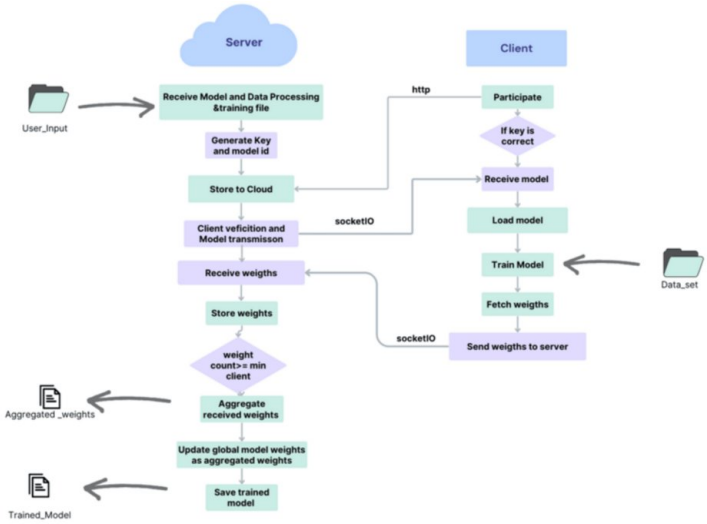

FIGURE 8: Server-client flow diagram

This paper suggests a way to use the key for verifying the participating clients, thus aiming for the authenticity of the clients. The key is generated by the server itself when it starts the FL process, and the client has to provide a correct key to participate in the FL process. The same key will be used for the encryption and decryption of the datamodel, i.e., the received model locally, without sharing the dataset. As the local model gets trained, the updates, which are nothing but different parameters like weighs, intercept, etc., based on the machine learning model, can be accessed. The client will encrypt these updates and will send them to the server. The server will receive data, and after decryption, it will store the updates from each client in a separate file. As some threshold has been achieved, which is mainly the number of clients, the server will perform aggregation and will update the parameter of the global model with the aggregated parameter. As the parameter of the global model gets updated, it gets trained. Finally, the given model will be trained and can be used for the specified task. This trained model is the output of the whole process.

Choosing the right aggregation technique: Importance of aggregation in FL

"*Aggregation is the foundation of FL, enabling the global model to generalize well across diverse client data distributions while adhering to privacy-preserving principles.*"

Aggregation in FL refers to the process of combining updates from various clients' locally trained models into a single global model [15]. This approach allows FL systems to operate in a decentralized manner, preserving privacy by avoiding the transfer of raw data to a central server. Aggregation ensures that the distributed learning process is efficient, collaborative, and effective, leveraging insights from different data sources without compromising privacy. Figure *9* show the abstract view of the aggregation in FL.

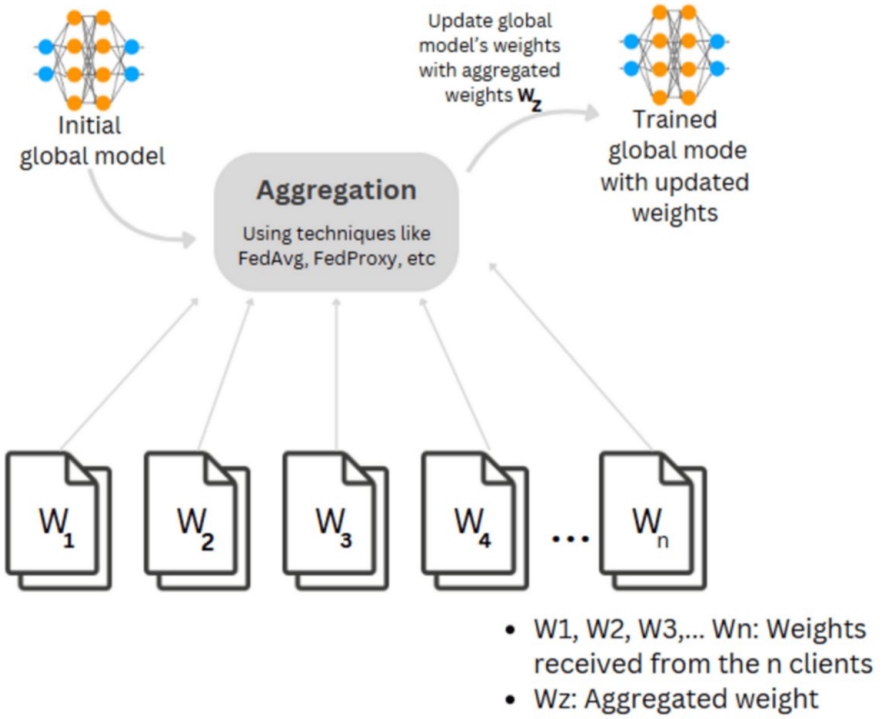

FIGURE 9: Aggregation

Many averaging techniques can be used as per the requirement. Some of them are as follows:

FedAvg: The most commonly used aggregation method and probably the simplest one is FedAvg. This algorithm aggregates the updates from clients based on their data size; more significant the data, higher weight for the update from that client [15]. In this way, updates represent better the data distribution among clients. FedAvg does very well on IID cases, but when dealing with non-IID, there might be some bias or convergence may not be fast.

Federated Proximal (FedProx): FedProx is an extension of FedAvg in that it adds a mechanism to prevent extreme deviations between local and global models [15]. This is especially beneficial when data seen by clients is very heterogeneous or when clients differ widely in computational capability. Smoothening updates ensures convergence and minimizes conflicts for model training across different clients.

Federated Normalized Averaging (FedNova): This method normalizes client updates based on the number of local training iterations. In this way, clients with various computational resources or training periods are contributing equitably to the global model [15]. This is particularly efficient in asynchronous FL systems, in which clients operate under different resource constraints. Thus, FedNova balances fairness and performance, and it is thus suitable for resource-diverse environments.

Zona (Robust Aggregation): For the noisier or malicious client update, Zona resorts to aggressive statistical operations like median and trimmed mean aggregations. This enables Zona to curb the overwhelming effect of outliers on the global model, rendering its solution secure and reliable. Zona finds application in environments with adversarial clients or less trustworthy updates where it can totally crash the overall system [15]. Zona trades little loss in accuracy for the robustness of its model.

Results

Experimental setup

The main experiment of our setup revolves around a central server and multiple clients. This system trains a machine learning model collaboratively. The key components and workflow of our system are described below.

System Architecture

In this architecture, a client-server platform is setup, where the server maintains a global model and initiates the learning process, while each client performs local training using its own dataset and shares

the model updates back to the server. Communication is carried out through WebSockets, allowing asynchronous real-time exchange of information, specifically sending and receiving messages. Figure *10* shows the system architecture.

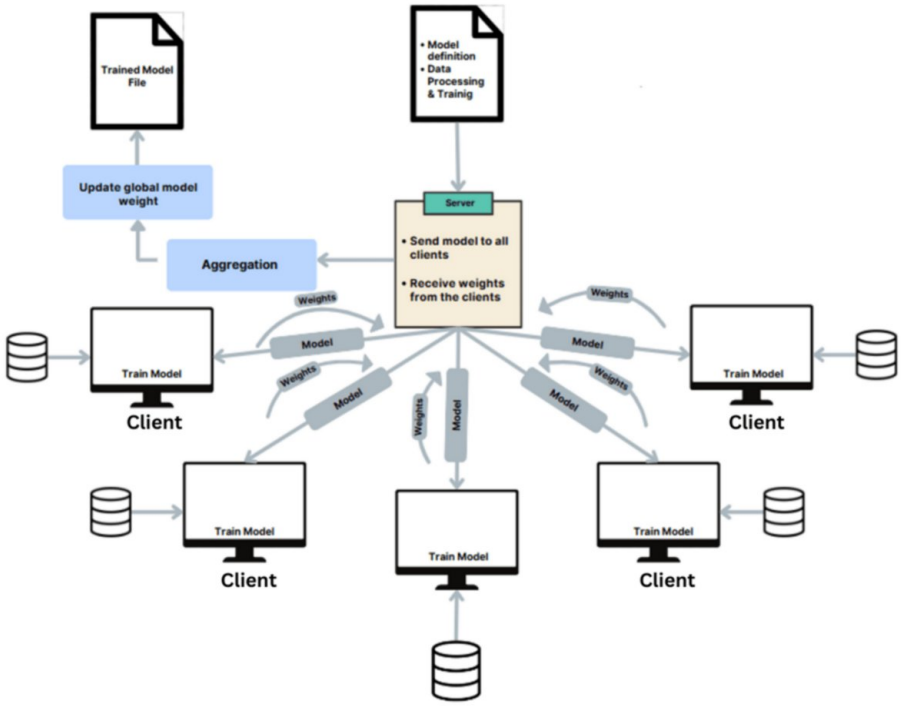

FIGURE 10: System architecture

Security and Encryption

To secure the data transfers, we use Fernet encryption for the model, and weights exchanges. Every time a client starts running their local model, he/she inputs a predefined key to start decryption; only clients which know this key can decrypt and make use of the data being passed. This encryption method increases confidentiality while protecting data from adversaries intercepting the data being sent.

Workflow

This is how the experimental setup works:

Server-Side Operations

1. The server starts a global deep learning model, Input_Model, and generates an encryption key by using the cryptographic library.

2. It transmits the encrypted model architecture along with a training script, data_processing_and_training.py, to connected clients.

3. The server waits and listens for incoming client updates while receiving the encrypted model weights as soon as the local training has been completed.

4. After the server gets the updates from each of the participating clients, it decrypts and aggregates the model weights with an averaging function.

5. The final aggregated model is kept for evaluation and further experimentation.

Client-Side Operations

1. Each client connects to the server through WebSocket and enters the pre-shared encryption key for secure communication.

2. The client receives and decrypts the global model and training script.

3. It is executed and dynamically loaded into the memory through exec() of the script decryption so that it can locally process some data of its choice, and further train on the model that had been sent over to the client.

4. The client ends training, picks updated weights, encrypts them and then transfers the updated encrypted model back to the server.

5. Continue sending until enough number of clients to attain the required end.

Data Distribution

The clients in this experiment are operating on heterogeneous datasets. This means that every client can have its unique dataset, and some might have a comparison of previously shared data by the clients. This is a non-IID setting, in which observations can show statistically different distributions across devices nonuniformly, reminiscent of the challenges of FL in real-world scenarios.

Experimental Environment

1. Programming Language: Python

2. Deep Learning Framework: TensorFlow/Keras

3. Networking: WebSockets via the websockets library

4. Encryption: Fernet symmetric encryption from the cryptography library

5. Hardware: The experiments were conducted on a local machine with an Intel i5 12th Gen processor and an RTX 3050 GPU

Experiment results

This section represents the experimental results we obtained by applying FL to the MNIST dataset [16] with our proposed platform. As mentioned previously, the evaluation mainly goes with the accuracy trends of the participants and the relative performance of aggregation algorithms: FedAvg, FedNova, FedProx, and Zona.

Client Performance Trends

Figure *11* shows the accuracy of each client at each epoch of training. The accuracy curve for each client was inspected after several training iterations. In all clients, accuracy increases monotonically, which shows that local model training and aggregation are effective.

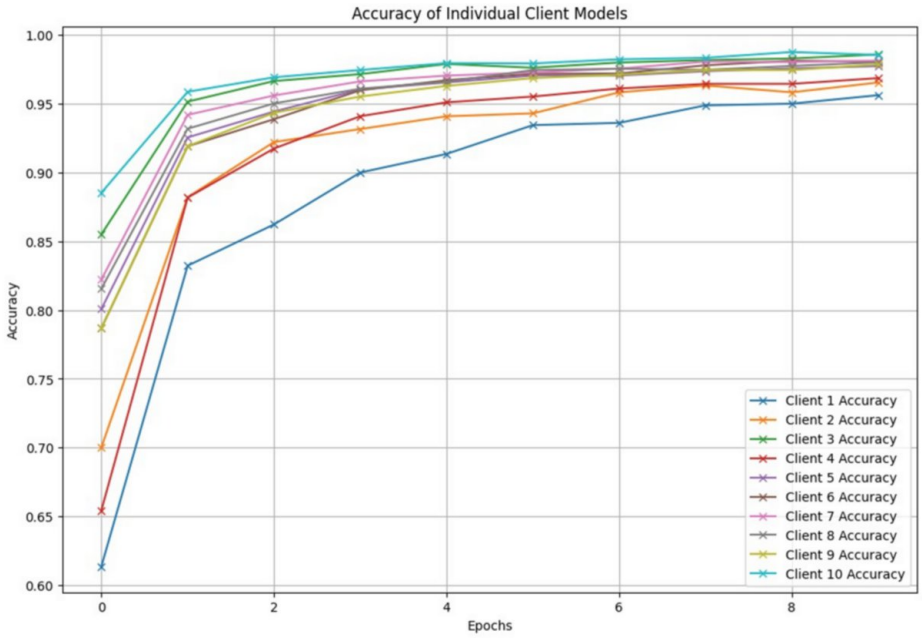

FIGURE 11: Individual client accuracy after each epoch

Though minor deviations were noticed between the clients, they were small and did not have a noticeable effect on the overall learning. The stability of the clients shows that the distribution of data between the clients is fairly uniform, and no individual biases were large enough to cause noticeable effects on the training process.

Figure *12* represents the accuracy of the locally trained model for each client after the whole training process.

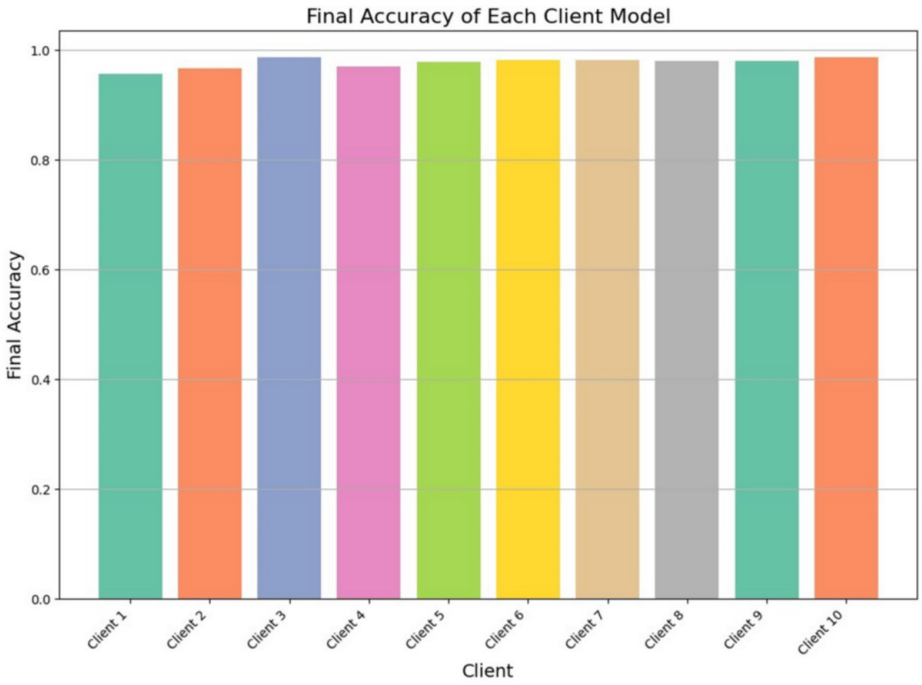

FIGURE 12: Accuracy of each client

Comparison of Aggregation Technique

The impact of different aggregation methods is assessed in terms of final accuracy and fairness. Figure *13*

shows the accuracy after implementing different aggregation techniques.

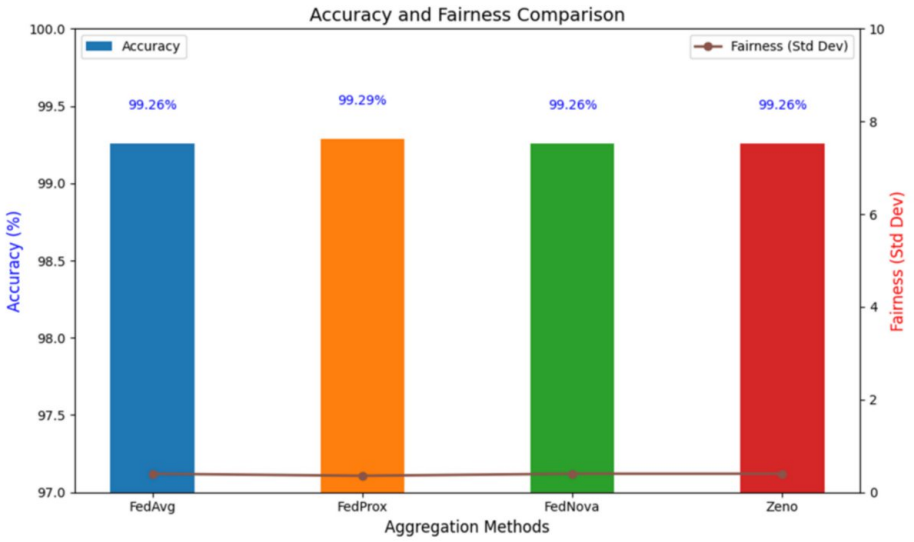

FIGURE 13: Accuracy and fairness comparison

The graph compares accuracy and fairness.

1. Accuracy: Accuracy reflects the effectiveness of the aggregated global model in predicting outcomes for unseen test data. Higher accuracy indicates that the chosen aggregation method successfully incorporates useful updates from client models.

2. Fairness: Fairness evaluates the equitable contribution of all clients to the global model, ensuring that no single client disproportionately influences the model. Fairness should be small in FL because there should be an equitable contribution from all clients to the global model.

Among the above algorithms, the FedProx algorithm showed the maximum accuracy and turned out to be the best amongst the methods adopted in our experiments. Performance-wise, it varies with dataset and model architecture. Results would vary in applications.

Comparison Between Traditional ML and FL Approach

This paper compares the traditional machine learning approach to the FL approach. Figure *14* shows the accuracy of the model trained using the traditional approach compared to the FL approach.

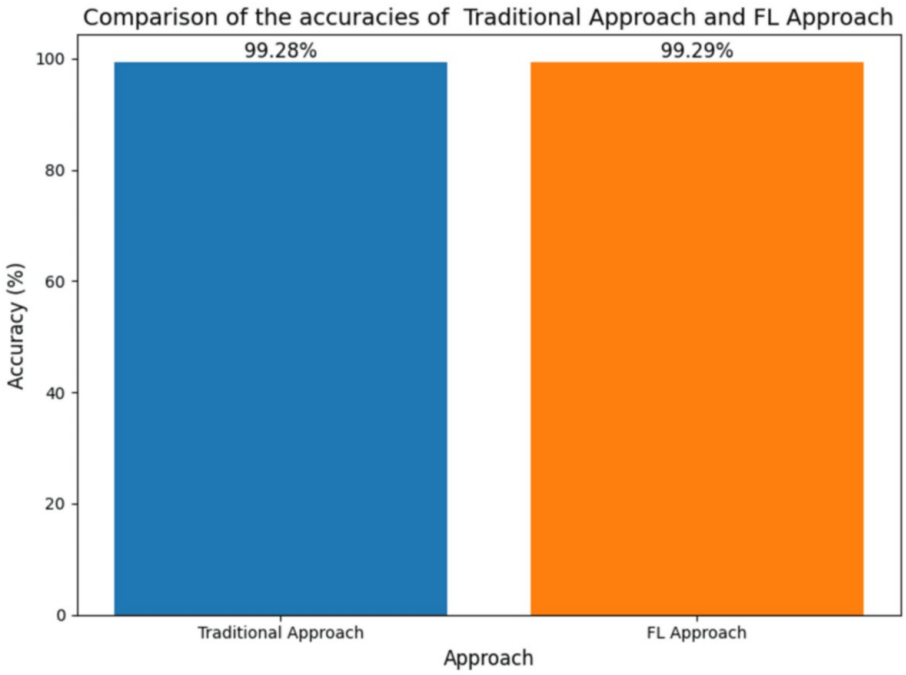

FIGURE 14: Comparison of the accuracies of traditional approach and FL approach

FL, Federated Learning

Discussion

Experimental analysis

While conducting the experiment, various insights were discovered that highlight the significance of this platform and demonstrate how it can make FL more efficient.

Collaboration in Client-Server Architecture: The client-server architecture facilitates the collaboration between the global model and clients seamlessly. WebSockets allow streams of updates to flow bidirectionally over the network in real time, even if and when clients are lost from time to time.

Real-World Simulation of FL: This configuration is quite similar to real-world FL scenarios, where devices may have intermittent network access and may not always be able to keep network access. In the experiment, there is a server and 10 clients, which are real-world conditions, i.e., training of a machine learning model on the MNIST dataset with an FL method.

Better Data Protection with Encryption: For safeguarding private information, the system applied Fernet encryption during data exchange. This allowed the model parameters and updates to remain private across the process. This security mechanism reduces the system's vulnerability to potential attacks or unauthorized access. It serves also as a critical enforcer in contexts where privacy matters.

Alignment with Data Protection Standards: Pursuing encryption as a key objective, the system addresses the hard data security requirements of FL. Such protection is particularly crucial in critical areas like health care and finance, where securing an individual's and a company's personal and confidential data is most important.

These aspects illustrate how the experimental test platform achieves an optimal tension between collaboration and security. It provides a reliable and secure environment for FL.

The impact of aggregation techniques

Since aggregation plays a crucial role in FL, the experiment conducted provides us with some insights into the impact of the aggregation techniques used to aggregate the weights received from the clients.

The main observations are as follows:

FedAvg: The baseline federated averaging method converged stably but needed more rounds to reach optimal accuracy than other methods.

Overall Accuracy: 99.26%
Fairness: 0.40%

FedNova: It was faster than FedAvg because it normalized the local updates; therefore, under different local epochs, it was better than others. However, it indicated the lowest level of accuracy compared to the rest of the four.

Overall Accuracy: 99.26%
Fairness: 0.40%

FedProx: It was eventually better than FedAvg and FedNova in terms of handling client heterogeneity and also resulted in the highest accuracy among those tested. It at the same time balances fairness and performance.

Overall Accuracy: 99.29
Fairness: 0.36%

Zona: With the use of robust statistical aggregation, it shows to be highly resistant to noisy or adversarial updates. In comparison to FedProx, this has a minor trade-off with lower accuracy but offers superior security and stability.

Overall Accuracy: 99.26%
Fairness: 0.40%

Here, FedProx provides the best accuracy and low fairness.

Below are some points describing how aggregation techniques affect the FL approach.

Impact on Model Accuracy: The aggregation technique directly affects the accuracy of the global model. Selecting an appropriate technique ensures the model generalizes well across diverse client datasets.

Fairness Across Clients: Some techniques, such as FedNova, normalize contributions from clients with varying computational capabilities or data volumes. This ensures that all clients, regardless of their resources, contribute fairly to the global model, improving inclusivity in the training process.

Stability and Convergence: Aggregation methods influence the stability and convergence rate of the global model. Techniques like FedProx reduce drastic deviations between local and global updates, leading to smoother and faster convergence, particularly in large-scale FL systems.

Balancing Trade-Offs: Every aggregation technique has its strengths and limitations. The choice must carefully balance trade-offs between accuracy, fairness, robustness, and computational complexity to align with the goals of the FL system.

Analysis of the FL platform with respect to the traditional approach.

Data Privacy

Traditional ML: In traditional ML, data from different sources is gathered into a centralized server for training. This allows for comprehensive model development but it raises significant privacy concerns. Transferring sensitive or proprietary data to a central location increases the risk of exposure and breaches.

FL: Here, data remains on the local devices where it is generated. Only the updates to the model, such as weights or gradients, are shared with a central server for aggregation. This setup drastically reduces the likelihood of sensitive data being exposed, making FL particularly advantageous in the filed where data privacy is very important. In experiment setup, we used Fernet library for encryption of the data, shared between the server and the client, thus making it more secure.

Accuracy

Traditional ML: Models trained using traditional ML methods benefit from direct access to a comprehensive and unified dataset. This centralized approach allows the model to capture global patterns effectively, often leading to higher accuracy. However, this method can also overfit to specific data

characteristics if the dataset is not diverse or properly regularized.

FL: Here, models are trained on data that remains distributed across multiple devices, which can lead to challenges when the data is non-IID. This lack of centralized access to the complete dataset can slightly reduce the accuracy of the FL model. The experimental result shows the accuracy of the model trained by the traditional approach and the FL approach is approximately the same, i.e., 99.2%.

Training Time

Traditional ML: The training duration in traditional ML depends on the dataset size and the hardware used. Large datasets can significantly increase time due to the need for data collection, transfer, and centralized processing. High-performance computing systems like GPUs or TPUs can help accelerate training, but they cannot eliminate delays caused by the initial data handling process.

FL: Here, FL distributes the training workload across multiple devices, allowing computations to occur locally. This can reduce reliance on centralized resources and speed up the process. However, communication between devices and the server introduces overhead, especially if devices have varying computational capabilities.

Conclusions

This work focuses on the impact of the proposed federated platform in solving the problems of distributed machine learning in practical applications. Through the use of a robust client-server framework and WebSocket-based communication in combination with Fernet encryption, the system ensured continuous communication and data protection, despite inconsistent connectivity and vendor-heterogeneous datasets. Experimental results illustrated the superiorities of several aggregation schemes (i.e., FedAvg, FedProx, FedNova, and Zeno) in both accuracy and fairness. Of these, FedProx turned out to be the most successful one with the highest accuracy of 99.29% and a balanced fairness measure. The alternative approaches, i.e., FedNova and Zeno, exhibited certain advantages, i.e., faster speed and robustness to adversarial updates, respectively, demonstrating the tradeoff between aggregation methods.

In addition, the contrast study between traditional machine learning and the FL strategy demonstrated the significant advantage of FL which can be used to defend the user data information and obtain similar accuracy. Non-IID distribution of datasets as a result of clients brought about problems; however, the aggregation methods solved them efficiently, and they guaranteed client-consistent performance. The applicability of the proposed platform to FL and, more broadly, to applications in the wild where privacy is an issue (e.g., healthcare, finance) is verified. Future work could focus on the platform's scalability, the incorporation of novel forms of encryption, as well as the deployment of the platform to more demanding environments and tasks.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Gayatri M. Bhandari, Nitin M. Shivale, Shrishail S. Patil, Pranav Prajapati, Atharv Burujpatte, Diya Gandhi, Mangal Kasare

Acquisition, analysis, or interpretation of data: Gayatri M. Bhandari, Nitin M. Shivale, Shrishail S. Patil, Pranav Prajapati, Atharv Burujpatte, Diya Gandhi, Mangal Kasare

Drafting of the manuscript: Gayatri M. Bhandari, Nitin M. Shivale, Shrishail S. Patil, Pranav Prajapati, Atharv Burujpatte, Diya Gandhi, Mangal Kasare

Critical review of the manuscript for important intellectual content: Gayatri M. Bhandari, Nitin M. Shivale, Shrishail S. Patil, Pranav Prajapati, Atharv Burujpatte, Diya Gandhi, Mangal Kasare

Supervision: Gayatri M. Bhandari, Nitin M. Shivale, Shrishail S. Patil, Pranav Prajapati, Atharv Burujpatte, Diya Gandhi, Mangal Kasare

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue.
Animal subjects: All authors have confirmed that this study did not involve animal subjects or tissue.
Conflicts of interest: In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from

any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

References

1. McMahan HB, Moore E, Ramage D, Hampson S, Agüera y Arcas B: Communication-efficient learning of deep networks from decentralized data. arXiv. 2017, 10.48550/arXiv.1602.05629
2. Wen J, Zhang Z, Lan Y, Cui Z, Cai J, Zhang W: A survey on federated learning: challenges and applications. International Journal of Machine Learning and Cybernetics. 2022, 14:513-535. 10.1007/s13042-022-01647-y
3. Liu Y, Zhang L, Ge N, Li G: A systematic literature review on federated learning: From a model quality perspective. arXiv. 2020, 10.48550/arXiv.2012.01973
4. Hard A, Rao K, Mathews R, et al.: Federated learning for mobile keyboard prediction. arXiv. 2019, 10.48550/arXiv.1811.03604
5. Rahman KMJ, Ahmed F, Akhter N, et al.: Challenges, applications and design aspects of federated learning: A survey. IEEE Access. 2021, 9:124682-124700. 10.1109/access.2021.3111118
6. Yuan H, Morningstar W, Ning L, Singhal K: What do we mean by generalization in federated learning?. arXiv. 2022, 10.48550/arXiv.2110.14216
7. Chen M, Jiang M, Dou Q, Wang Z, Li X: FedSoup: Improving generalization and personalization in federated learning via selective model interpolation. arXiv. 2023, 10.48550/arXiv.2307.10507
8. Zhu H, Zhang H, Jin Y: From federated learning to federated neural architecture search: a survey. Complex & Intelligent Systems. 2021, 7:639-657. 10.1007/s40747-020-00247-z
9. Bharati S, Mondal MRH, Podder P, Prasath VBS: Federated learning: Applications, challenges and future directions. International Journal of Hybrid Intelligent Systems. 2022, 18:19-35. 10.3233/his-220006
10. Fu L, Zhang Z, Tan L, Yao Z, Tan H, Xie J, She K: Blockchain-enabled device command operation security for Industrial Internet of Things. Future Generation Computer Systems. 2023, 148:280-297. 10.1016/j.future.2023.06.004
11. Bonawitz K, Eichner H, Grieskamp W, et al.: Towards federated learning at scale: System design. Proceedings of Machine Learning and Systems. 2019, 374-388.
12. Kairouz P, McMahan HB, Avent B, et al.: Advances and Open Problems in Federated Learning. Now Publishers, Boston; 2021. 10.1561/2200000083
13. Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V: Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems. 2020, 429-450.
14. Pillutla K, Kakade SM, Harchaoui Z: Robust aggregation for federated learning. IEEE Transactions on Signal Processing. 2022, 70:1142-1154. 10.1109/tsp.2022.3153135
15. Qi P, Chiaro D, Guzzo A, Ianni M, Fortino G, Piccialli F: Model aggregation techniques in federated learning: A comprehensive survey. Future Generation Computer Systems. 2024, 150:272-293. 10.1016/j.future.2023.09.008
16. MNIST Dataset. Accessed: January 19, 2025: https://www.kaggle.com/datasets/hojjatk/mnist-dataset.