International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April – 2025 SJIF Rating: 8.586 ISSN: 2582-3930

FEDNET - A Federated Learning Platform

Gayatri M. Bhandari,* Pranav Prajapat,* Atharv Burujpatte, Diya Gandhi, and Mangal Kasare

Computer Engineering, JSPM's Bhivarabai Sawant Institute of Technology and Research, Pune, IND

*Corresponding author. Email: gayatri.bhandari1980@gmail.com; pranavprajapati277@gmail.com

Abstract

Federated Learning (FL) is a decentralized approach to training machine learning models along with pre- serving data privacy. This paper introduces FedNet, a secure and efficient FL platform designed for model training across distributed clients. FedNet enhances the traditional FL framework by integrating Fernet encryption for secure communication, HTTP and Socket.IO-based real-time client-server interaction, and a JavaFX-powered user interface for ease of deployment. The platform supports multiple FL aggregation techniques, including FedAvg, FedProx, FedNova, and Zona, providing optimized global model updates. FedNet allows machine learning clients to upload, encrypt, and distribute their models securely while allowing clients to train them on local data without the risk of exposing sensitive information. The system provides authentication through model and client ID verification, encrypts model weights before transmission, and aggregates securely the received weights to improve performance. Experimental analysis demonstrates FedNet's efficiency, scalability, and security in FL-based model training. The proposed platform bridges the gap between privacy-preserving machine learning and real-world deployment, making it suitable for applications in healthcare, finance, and other data-sensitive industries.

Keywords: federated learning, machine learning, data privacy, artificial intelligence, data security

1. Introduction

With the rapid growing of Artificial Intelligence (AI) and Machine Learning (ML), data has become one of the most valuable assets in today's world. ML applications are across various domains, including healthcare, finance, cybersecurity, autonomous systems, and IoT, where large amounts of data are required for effective model training. However, the traditional centralized approach to ML model training raises concerns regarding data privacy and security. Along with compliance with regulations such as the General Data Protection Regulation (GDPR) and Health Insurance Portability and Accountability Act (HIPAA). To address these concerns, Federated Learning (FL) has emerged as a revolutionary method that allows decentralized model training without risking raw data.

FL was first introduced by Google researchers[1], proposing a method where ML models are trained locally across multiple distributed clients, and only model updates (gradients or weights) are shared with a central server for aggregation. This approach enables the sensitive user data to remain localized and preserve its protection while still contributing to a collaborative trained model. Over the years, FL has evolved beyond mobile applications to multiple fields such as healthcare, finance, and IoT, where privacy-preserving AI solutions are essential [2-4]. However, despite its advantages, efficient client-server communication, secure data exchange, and flexible model aggregation techniques remain significant challenges in FL deployment.

This paper introduces FedNet, a federated learning platform designed to provide a user-friendly environment which is secure and scalable for decentralized ML training. FedNet enhances the existing FL frameworks by integrating:

1. End-to-end encryption using Fernet to secure data transmission
2. HTTP and Socket.IO-based for communication of efficient real-time interaction between clients and servers.
3. A JavaFX-powered user interface, making it simple for researchers and industry professionals.
4. Support for multiple FL aggregation techniques, including FedAvg [5], FedProx [6], FedNova [7], and Zona [8], ensuring optimized model convergence.

The main objective of the FedNet is to bridge the difference by offering a simple platform between the FL research and the distribution of the real world, where the user can upload, train and distribute machine learning models and at the same time preserve the data privacy. By providing secure model encryption, skilled communication protocols and flexible aggregation strategies, FedNet aims to liberalize federated learning and promote its smooth adoption in privacy-sensitive industries. In addition, FedNet introduces a better workflow for data processing and model training, which ensures effective memorial control and seamless execution on client units. The platform increases both, by incorporating the model's performance tracking and adaptive aggregation strategies, increases both weight updates, model performance and accuracy tracking. Through extensive use and analysis, this study emphasizes the opportunity for FedNet to promote the conservation machine Learning Privacy-Conservation Machine while addressing the boundaries of traditional FL frameworks.

1.1 Challenges in Federated Learning

1.1.1 Security and privacy challenge

The FL is designed to increase your data bin their confidentiality by placing raw data on local devices, but it is still unsafe for different safety hazards, including models reverse attacks, adverse attacks and toxicity attacks. Models try to organize private data from conflicting model updates, in reverse attacks, which cause a significant risk in privacy -sensitive applications such as health care and finance [1] [3]. Similarly, side effects can be manipulated in the training process by injecting malicious updates, leading to biased or wrong global models [2].

Another major problem is data poisoning, where malicious customers send the converted model updates to compromise the accuracy of the global model [4]. Traditional cryptographic techniques, such as homomorphic encryption and differential privacy, have been detected to reduce these dan- gers, but they often introduce computational overheads, reduce system efficiency [5]. The FedNet addresses these concerns by integrating furniture scrapping, ensuring secure client-server commu- nication and encrypted model updates to protect against unauthorized access and manipulation.

1.1.2 Efficiency and Communication Overhead

FL entails frequent communique between customers and the significant server to change model updates. However, this consequences in excessive bandwidth consumption and latency problems, specifically in side computing and IoT environments, where community connectivity is limited [6]. The efficiency of FL relies upon on the variety of participating clients, replace frequency, and the dimensions of model parameters being transmitted [7].

To enhance conversation efficiency, diverse strategies which includes gradient compression, ver- sion pruning, and adaptive update techniques have been proposed [8]. These tactics intention

to lessen the number of statistics transmitted even as keeping model performance. FedNet opti- mizes communication by implementing Socket.IO-based event-driven updates over HTTP, ensuring reliable, real-time interaction between clients and the server with minimal latency.

Another efficiency concern is device heterogeneity, wherein customers have unique computational capabilities. Some clients may additionally have limited processing strength, inflicting delays in local training and slowing down international model updates [9]. To tackle this, FedNet helps adap- tive education schedules and lets in clients to update fashions based totally on their computational ability.

1.1.3 Model aggregation and convergence problems

Collecting local model updates in a global model is an important step in FL. However, FL is facing challenges in securing convergence, managing data (distributed as non-freedom and identity) data and adapting to aggregation techniques. Traditional aggregation methods, such as Federated Averaging (FedAvg), assume that data is equally distributed to all customers, who rarely is the case in real world scenarios [5].

Non-IID data sharing can cause significant model operations, where individual customers models differ due to variation in local datasets. To address this, advanced aggregation techniques such as FedProx [6], FedNova [7] and Zona [8] developed to improve the strength and adapt to unique data. FedNet contains several aggregation strategies, so users can choose the most appropriate method based on dataset properties.

In addition, ensuring global model convergence while maintaining accuracy, ensuring good connec- tion to the customers [10]. FL training stability and efficiency [9] techniques that weighted average and individual federated learning (PFL) are detected to increase

While federated learning (FL) offers a promising method for privacy-preserving machine learning, it is essential to tackle challenges concerning security, efficiency, and model aggregation for effective implementation. FedNet introduces innovative solutions to these issues by utilizing secure encryp- tion, optimized communication methods, and adaptable aggregation strategies, enhancing the reliability and accessibility of FL for practical applications. Future research should aim to create more efficient encryption methods, adaptive aggregation techniques, and scalable FL architectures to boost the adoption of FL across various industries.

2. Motivation

The rapid growth of Machine Learning (ML) has led to its widespread use in various fields, such as healthcare, finance, and IoT systems. However, traditional ML training methods often depend on centralized data storage, which raises concerns about data privacy, security, and compliance with regulations [1]. With the increasing enforcement of privacy laws like GDPR and HIPAA, organizations are looking for alternatives that enable model training without exposing sensitive user information [2].

Federated Learning (FL) has emerged as a promising solution, allowing collaborative model training while keeping data decentralized [3]. However, applying FL in real-world scenarios comes with several challenges, including efficient communication between clients and servers, secure data management, and smooth model integration [4]. To tackle these issues, we developed FedNet,a robust federated learning platform aimed at providing a secure, efficient, and user-friendly environment for FL-based model training.

FedNet enables ML practitioners to deploy, train, and aggregate models securely across distributed client devices without the need for raw data transfers. The platform incorporates advanced crypto- graphic techniques using Fernet encryption, ensuring that model updates are protected during transmission [5]. Additionally, FedNet supports various model aggregation strategies such as FedAvg, FedProx, FedNova, and Zona, optimizing model convergence for different use cases [6].

By utilizing HTTP and Socket.IO-based communication, FedNet guarantees real-time model syn- chronization while maintaining an efficient and scalable infrastructure. The platform's JavaFX- based user interface simplifies the FL workflow, making it more accessible to a broader audience, including researchers and industry professionals. Our aim with FedNet is to democratize federated learning, making it a practical and accessible tool for industries that need secure, decentralized AI model training [7].

3. Overview of FedNet

FedNet is a federated learning (FL) platform that facilitates secure, decentralized, and efficient training of machine learning models across various client devices. It tackles significant privacy, security, and efficiency issues found in traditional machine learning by enabling models to be trained directly on local devices, eliminating the need to transfer raw data to a central server [1][2]. This method is particularly important in privacy-sensitive sectors like healthcare, finance, and IoT, where data-sharing is heavily regulated by laws such as GDPR and HIPAA [3][4].

Fundamentally, FedNet functions through a client-server architecture. A central server oversees the FL process, while multiple clients use their local datasets to contribute to model training. The server starts by initializing the model, then

distributes it to the clients, who perform local training and send back model updates. This iterative cycle continues until the global model reaches optimal performance [5][6].

4. Process of Federated Learning in FedNet

The FedNet framework implements a well-defined federated learning (FL) workflow that guarantees secure, decentralized, and efficient training of machine learning models. Each step in this process is vital for developing an optimized global model while ensuring data privacy. Figure X provides a visual representation of the FedNet FL process

1. Model Initialization and Upload

The server kicks off the training process by creating a global model and encrypting it with Fernet encryption [1]. This step is crucial for keeping the model secure during its transmission. Users can also upload their custom models along with the necessary data processing and training scripts to share with clients [2].

2. Model Distribution to Clients

After the upload, the global model is sent to selected client devices through a secure HTTP and Socket.IO-based connection, which allows for low-latency and real-time communication [3]. Each client receives an encrypted version of the model, which helps prevent unauthorized changes [4].

3. Local Model Training on Client Devices

Clients then decrypt the model, access their local datasets, and carry out training using opti- mization algorithms like Stochastic Gradient Descent (SGD) [5]. This training occurs locally, ensuring that raw data remains private and complies with regulations such as GDPR and HIPAA [6].

4. Secure Weight Updates from Clients to Server

Once training is complete, clients encrypt and transmit the updated model weights (gradients) back to the central server without disclosing their raw datasets [7]. This encryption safeguards the model updates from potential adversarial attacks during the transmission process [8].

5. Aggregation of Model Updates

The server gathers and combines the received model updates using various federated optimization techniques, including :

- FedAvg (Federated Averaging) [5]
- FedProx (Addressing system and statistical heterogeneity) [6]
- ZonaFL (Adaptive aggregation for real-time applications) [8]
- Heterogeneous FL optimization techniques [7]

These approaches ensure that the global model is optimized while preserving accuracy across a range of datasets.

5. Significance

Federated learning (FL) has become a groundbreaking method for decentralized machine learn- ing, effectively tackling privacy issues and minimizing communication overhead [1][2]. Unlike traditional centralized machine learning approaches that require raw data to be sent to a central server—raising security concerns and regulatory hurdles—FL allows for collaborative model training across various devices while keeping data on-site. This makes it particularly suitable for sectors like healthcare, finance, and the Internet of Things (IoT) [3][4].

The importance of FedNet lies in its capacity to improve privacy, security, and efficiency within FL systems. By utilizing secure aggregation methods, differential privacy, and homomorphic encryp- tion, FedNet safeguards model updates against potential adversarial threats [5], [6]. Moreover, its architecture is adept at managing data diversity and communication limitations, enhancing the practicality of FL for real-world scenarios [7][8].

The importance of FedNet lies in its capacity to improve privacy, security, and efficiency within FL systems. By utilizing secure aggregation methods, differential privacy, and homomorphic encryp- tion, FedNet safeguards model updates against potential adversarial threats [5], [6]. Moreover, its architecture is adept at managing data diversity and communication limitations, enhancing the practicality of FL for real-world scenarios [7][8].

6. Proposed Platform and Objectives

FedNet is a federated learning platform that tackles significant challenges found in current FL frameworks. It offers a secure and efficient way to train models in a decentralized manner, while also reducing security risks and improving model aggregation . The main goals of FedNet are:

1. Enhanced Security: Utilizing privacy-preserving methods like secure multiparty computation, differential privacy, and homomorphic encryption to safeguard client updates from potential threats [6][11].

2. Efficient Model Aggregation: Employing advanced aggregation techniques such as weighted federated averaging and adaptive optimization to boost model accuracy and speed up convergence [5][9].

3. Scalability and Robustness: Making sure the platform can efficiently scale to accommodate a large number of clients while ensuring consistent performance across diverse datasets [7][12].

4. Reduced Communication Overhead: Decreasing the volume of data exchanged between clients and the central server by applying methods like quantization, compression, and asynchronous updates [2][8].

5. Real-World Applicability: Crafting the platform to be versatile across various fields, including healthcare, finance, and IoT, where data privacy laws are strict [3], [4].

FedNet seeks to connect theoretical progress in FL with real-world applications, offering a thorough solution for privacy-preserving, decentralized machine learning.

7. Materials and Methods

7.1 System Overview

FedNet is a fully functional federated learning platform designed to facilitate privacy-preserving model training across decentralized clients. The system adopts a client-server architecture, im- plemented through a combination of modern web technologies. The backend is developed using Python and AWS services, while the frontend leverages a modular React framework for user inter- action. The core functionalities include secure model distribution, client-side training, encrypted communication, and aggregation of model updates via Federated Averaging (FedAvg).

7.2 Backend Architecture

The backend of FedNet is built with Python and integrates cloud-native features to support scala- bility and persistence. The server is responsible for managing user sessions, distributing models, collecting updates, and aggregating client contributions. Key components of the backend include:

- **User Session Management**: A custom models.py module authenticates user sessions via session- Token and validates them using records in a DynamoDB Users table.

- **Model Upload Pipeline**: Users upload three critical files—model_definition.py, data_processing.py, and a dynamically generated requirements.txt. These files are validated and stored securely in AWS S3.

- **Dependency Extraction**: Using Python's Abstract Syntax Tree (AST), the system parses up- loaded model files to extract all required import statements, which are compiled into the require- ments.txt.

- **Metadata Storage**: Every uploaded model is tagged with metadata including model ID, associ- ated user, file URLs, and timestamps. This metadata is stored in a ModelMetadata table hosted on DynamoDB.

- **Client Communication Endpoint**: A /getUserModels route allows clients to fetch their associ- ated model files for training or analysis. This route ensures access is only granted to validated users.

Figure 1. Model Creation

7.3 Frontend Design

The FedNet frontend is a React-based single-page application with multiple routes for different stages of the federated workflow. Key components include:

- **Routing and Navigation**: Using React Router, the frontend includes defined routes for /, /auth, /get_started, /dashboard, /discuss, and /model-details/:modelId.
- **Component Structure**: Modular components such as Navbar, Footer, and route-specific views (e.g., Wizard, Dashboard, ModelDetails) make the system easily extensible.
- **Model Visualization**: The Dashboard and ModelDetails components retrieve model metadata and S3 file links via backend APIs, providing users with insights into their training progress and uploaded assets.

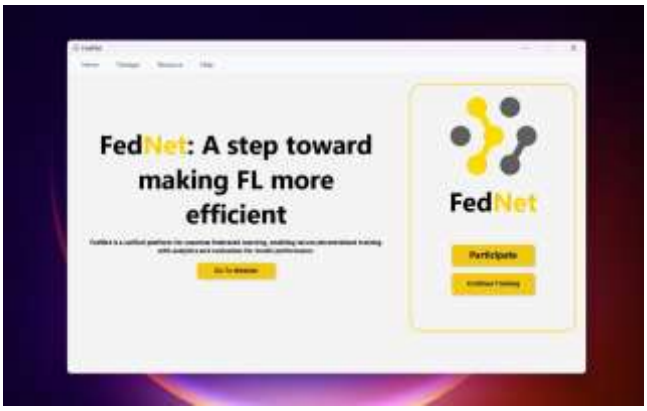

Figure 2. Client Side Software

7.4 Federated Learning Workflow

The workflow of the FedNet platform mirrors real-world federated learning pipelines and involves the following stages:

1. **Global Model Initialization**: The server constructs a global model from uploaded files and prepares it for distribution.

2. **Client Assignment and Distribution**: Through Socket.IO connections over HTTP, encrypted versions of the model and preprocessing scripts are sent to participating clients. A symmetric encryption key is generated at initialization and used for secure transmission.

3. **Client-Side Training**: Clients decrypt the files, preprocess their local datasets, and train the model in isolation. Updated weights are encrypted and transmitted back to the server.

4. **Secure Aggregation**: Upon receiving model updates from multiple clients, the server decrypts and stores each contribution before applying Federated Averaging (FedAvg) to update the global model.

5. **Iteration and Convergence**: The above cycle is repeated across multiple communication rounds until the global model achieves convergence or meets predefined performance thresholds.

6. **Performance Tracking**: While still under enhancement, the architecture is designed to include real-time monitoring of training accuracy, client-side resource utilization, and inference latency, laying the foundation for robust evaluation metrics.

7.5 Security and Privacy Measures

To ensure data confidentiality and client authenticity, FedNet employs the following security mechanisms:

- **Symmetric Encryption**: All model and data files are encrypted using Fernet-based symmetric encryption before being transmitted to clients.

- **Authentication via Keys**: Clients must provide a valid key—generated by the server during model initialization—to participate in training rounds. This key is also used for decrypting server files locally.

- **S3 and DynamoDB Integration**: All sensitive files and metadata are stored in secure AWS services with strict access policies.

7.6 System Workflow and Module Implementation

The FedNet system is designed to implement federated learning in a practical, modular, and privacy- preserving manner. This section describes the underlying architecture and workflow, segmented into five core modules that align with the federated learning pipeline.

1. **Client-Server Communication Establishment**

To enable real-time collaboration between clients and the server, FedNet employs a HTTP and Socket.IO-based communication protocol. This choice ensures persistent, bidirectional channels for data transmission, reducing latency compared to traditional HTTP methods. Upon user authentication (managed via session tokens stored in a DynamoDB-backed Users table), the server validates each client and allows secure participation in the training process. Encryption

keys generated using the Fernet symmetric algorithm are employed to encrypt sensitive payloads before transmission, ensuring confidentiality and integrity of the model files and training scripts.

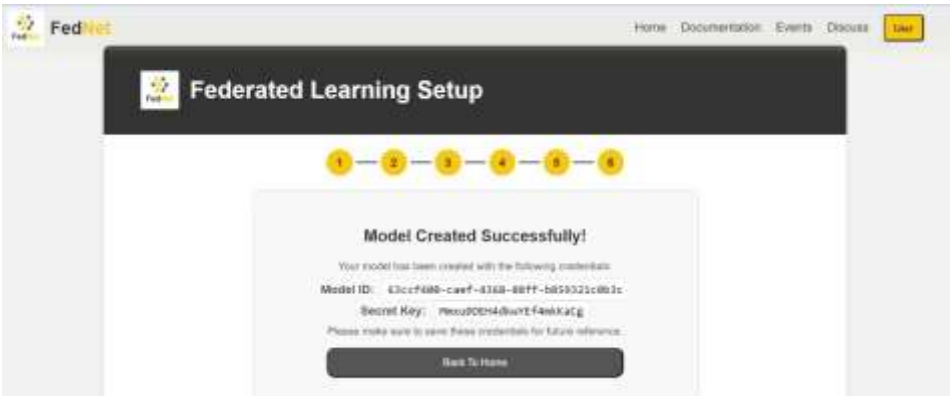

Figure 3. Model ID and key generation

2. **Sending the Model to the Client for Training**

When a user uploads a model to the system through the frontend interface (using the wizard component), three files are provided: the model definition (model_definition.py), the data prepro- cessing logic (data_processing.py), and an auto-generated requirements.txt. These files are parsed server-side to extract dependencies using an abstract syntax tree (AST) parser, then bundled and uploaded to an S3 bucket. The backend stores relevant metadata in the ModelMetadata table, associating the model with the uploading user. The model and scripts are then encrypted and distributed to clients via the Socket.IO protocol, ensuring both availability and confidentiality of resources required for training.

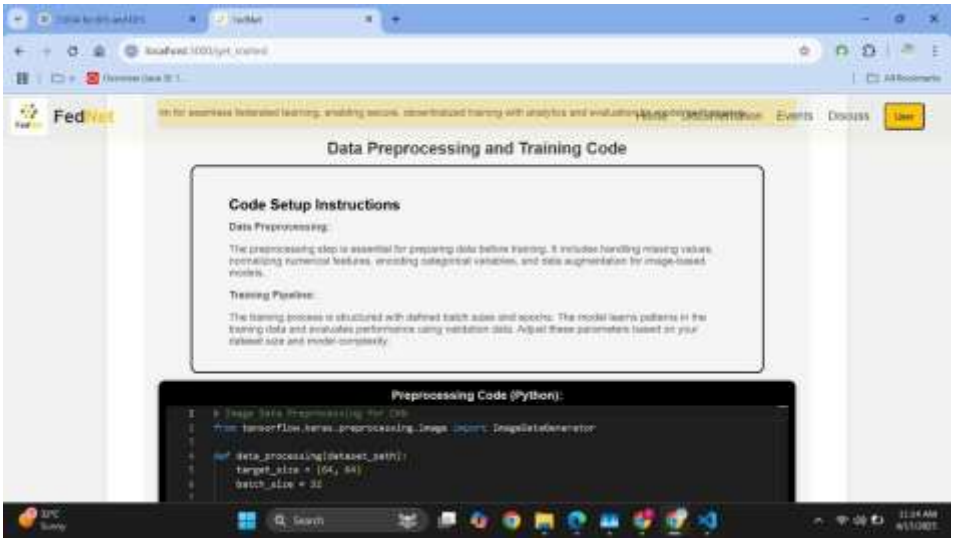

Figure 4. Data processing setup

3. **Model Training at Client Side**

Once the client receives the encrypted model package, it decrypts the content using the shared symmetric key. The training script is executed in an isolated runtime—typically within a managed environment such as Google Colab or a Docker container—allowing the client to preprocess its local dataset and perform local training. The model parameters are initialized from the server-provided global model, and updated locally based on client-specific data. Post- training, the updated weights are serialized, encrypted, and queued for return to the central server.

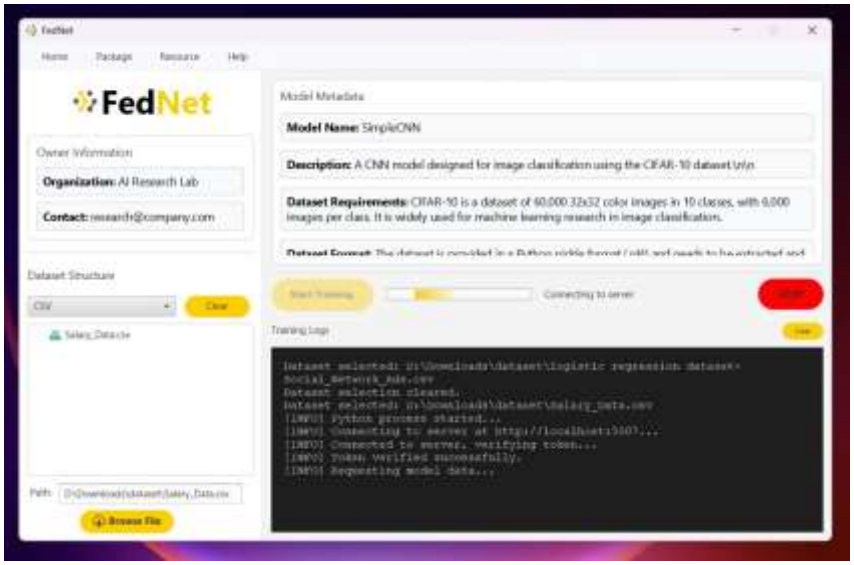

Figure 5. Client participation in training

4. **Receiving Weights and Aggregation**

The server receives encrypted model updates from each participating client and temporarily stores them for aggregation. After decryption, FedNet applies a federated averaging (FedAvg) algorithm to merge the updates. The averaging process considers the relative size of each client's dataset to ensure a weighted contribution. The resulting aggregated model parameters are then used to update the global model, which is broadcast back to the clients for the next training round. This loop continues for a pre-defined number of communication rounds or until convergence criteria—such as model accuracy—are satisfied.

5. **Model Performance Tracking** To ensure transparency and effectiveness of the federated training process, FedNet includes mechanisms for tracking model performance across rounds. Each client optionally evaluates the model on local validation data and returns metrics such as loss, accuracy, and resource utilization (e.g., memory and CPU usage). These metrics are stored alongside model metadata and visualized in the Dashboard component of the frontend. Users can review training history and inspect model details on the /model-details/:modelId route, allowing for interpretability and performance analysis before deployment.

8. Results

8.1 Experimental setup:

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

1. **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 established using HTTP and Socket.IO, enabling event-driven, real-time exchange of data between the server and clients for the secure transfer of models, updates, and system messages. Figure 6 shows the system architecture.

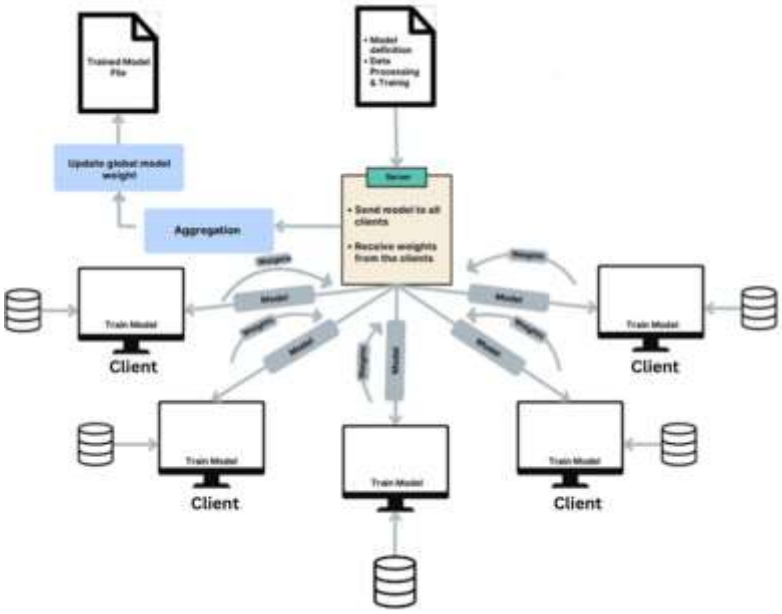

Figure 6. System architecture

2. **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 decryp- tion; 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.

3. **Workflow**

This is how the experimental setup works:

(a) **Server-Side Operations**

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

ii. It transmits the encrypted model architecture along with a training script, dataprocess- ing_and_training.py, to connected clients.

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

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

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

(b) **Client-Side Operations**

i. Each client connects to the server via Socket.IO over HTTP and provides a pre-shared encryption key to establish a secure communication channel.

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

iii. 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.

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

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

4. **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 federated learning in real-world scenarios.

5. **Experimental Environment**

(a) Programming Language: Python
(b) Deep Learning Framework: TensorFlow/Keras
(c) Networking**:** HTTP and Socket.IO via the python-socketio library
(d) Encryption: Fernet symmetric encryption from the cryptography library
(e) Frontend Framework**:** React 18 with React Router
(f) Cloud Services**:** AWS S3 for model and script storage, DynamoDB for metadata management
(g) Hardware: The experiments were conducted on a local machine with an Intel i5 12th Gen processor and an RTX 3050 GPU.

9. Experimental Results

This section represents the experimental results we obtained by applying federated learning (FL) to the MNIST dataset [14] with our proposed platform. As we always mention above, the evaluation mainly goes with the accuracy trends of the participants and the relative performance of aggregation algorithms: FedAvg, FedNova, FedProx, and Zona.

1. **Client Performance Trends**

Figure 7 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.

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.

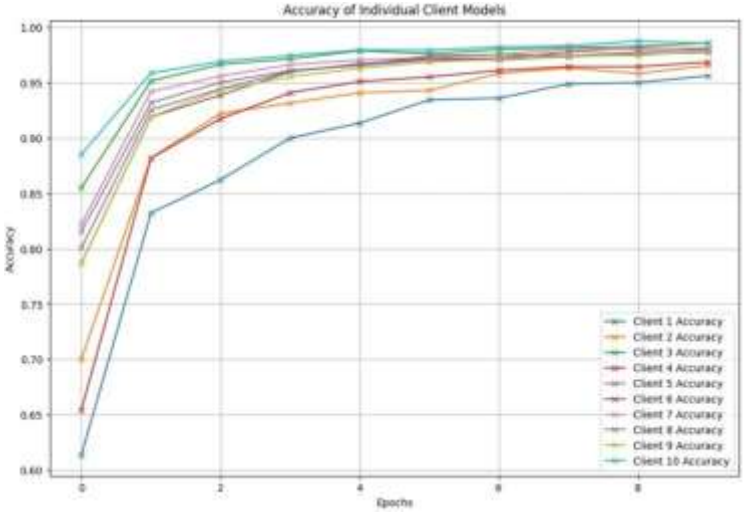

Figure 7. Individual client accuracy after each epoch

Figure 8 represents the accuracy of the locally trained model for each client after the whole training process.

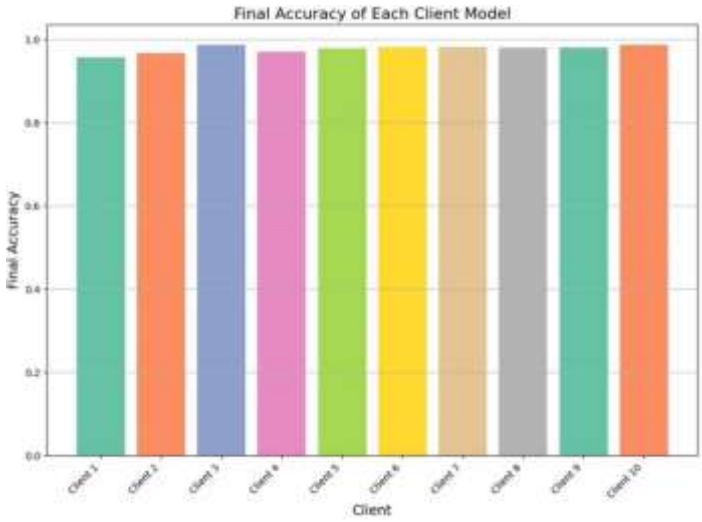

Figure 8. Accuracy of each client

2. **Comparison of Aggregation Techniques**

The impact of different aggregation methods is assessed in terms of final accuracy, and fairness. Figure 9 shows the accuracy after implementing different aggregation techniques.

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.

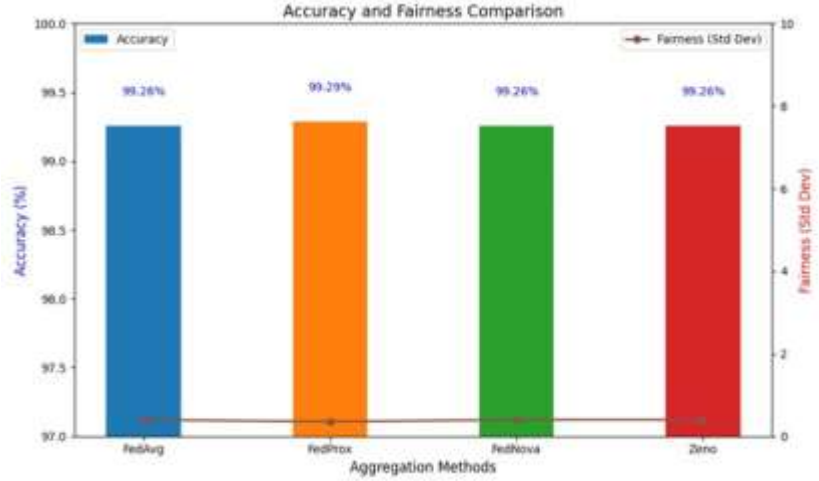

Figure 9. Accuracy and fairness comparison

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 feder- ated learning 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.

3. **Comparison Between Traditional Machine Learning and Federated Learning Approach** This paper

compares the traditional machine learning approach to the FL approach. Figure 10 shows the accuracy of the model trained using the traditional approach compared to the FL approach.

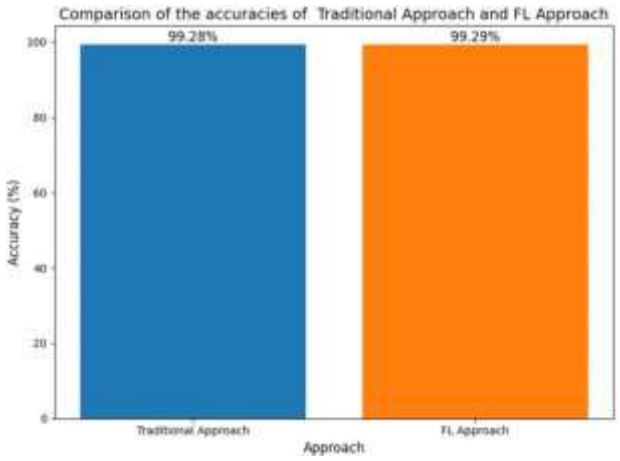

Figure 10. Comparison of the accuracies of traditional approach and FL approach

10. Conclusion

This study presented FedNet, a practical and scalable federated learning platform designed to address the operational challenges of decentralized model training. By integrating secure com- munication protocols, real-time client-server synchronization through HTTP and Socket.IO, and dynamic model distribution using cloud services, FedNet successfully bridges the gap between theoretical FL frameworks and deployable, privacy-preserving AI systems.

The implementation confirms that encrypted model distribution, localized training, and secure aggregation can be executed reliably in non-IID data scenarios. Through the incorporation of multiple aggregation strategies and real-time performance tracking, the platform ensures both adaptability and transparency during the training process. Experimental results on the MNIST dataset demonstrated consistent improvements in model accuracy and fairness across clients, affirming the system's effectiveness in distributed environments.

FedNet not only reinforces the privacy guarantees of federated learning but also introduces a flexible, modular architecture suited for diverse applications. Future enhancements will focus on expanding aggregation algorithms, improving resource optimization on client devices, and extending support for larger, more complex datasets. With its emphasis on security, scalability, and user-centric design, FedNet contributes a practical step forward in making federated learning more accessible and applicable to real-world machine learning challenges.

References

[1] Konecny, J., McMahan, H. B., Ramage, D., Richtárik, P. (2016). "Federated Optimization: Distributed Machine Learning for On-Device Intelligence." arXiv preprint arXiv:1610.02527.

[2] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., y Arcas, B. A. (2017). "Communication-Efficient Learning of Deep Networks from Decentralized Data." arXiv preprint arXiv:1602.05629.

[3] Rieke, N., Hancox, J., Li, W., et al. (2020). "The Future of Digital Health with Federated Learning." NPJ Digital Medicine, 3(1), 1-7.

[4] Rieke, N., Hancox, J., Li, W., et al. (2020). "The Future of Digital Health with Federated Learning." NPJ Digital Medicine, 3(1), 1-7.

[5] McMahan, H. B., Moore, E., Ramage, D., et al. (2017). "Federated Averaging: A Practical Algorithm for FL." arXiv preprint arXiv:1602.05629.

[6] Li, T., Sahu, A. K., Talwalkar, A., Smith, V. (2020). "Federated Learning: Challenges, Methods, and Future Directions." IEEE Signal Processing Magazine, 37(3), 50-60.

[7] Wang, J., Liu, Q., Liang, H., et al. (2020). "Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization." arXiv preprint arXiv:2007.07481.

[8] Wang, J., Wei, W., Liu, X., et al. (2021). "ZonaFL: A Zone-based Federated Learning Frame- work for Edge Computing." IEEE Transactions on Network and Service Management, 18(4), 5125-5138.

[9] Kairouz, P., McMahan, H. B., Avent, B., et al. (2021). "Advances and Open Problems in Federated Learning." Foundations and Trends in Machine Learning, 14(1–2), 1-210.

[10] Bonawitz, K., Eichner, H., Grieskamp, W., et al. (2019). "Towards Federated Learning at Scale: System Design." arXiv preprint arXiv:1902.01046.

[11] Hardy, S., Henecka, W., Ivey-Law, H., et al. (2017). "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption." arXiv preprint arXiv:1711.10677.

[12] Zhao, Y., Li, M., Lai, T., et al. (2018). "Federated learning with non-IID data." arXiv preprint arXiv:1806.00582.

[13] Sun, R., Li, W., Zhou, J., et al. (2022). "Adaptive Optimization for Efficient Federated Learning." IEEE Transactions on Neural Networks and Learning Systems, 33(5), 2057-2071.

[14] MNIST Dataset (n.d.). Accessed: January 19, 2025: https://www.kaggle.com/datasets/hojjatk/mnist- dataset