Enhancing Computer Vision through Federated Learning

Rida Fathima
Student
Presidency University

Abstract*:*

The overall goal of our project is to implement federated learning computer vision related projects such as object detection. The objective is to Develop an object detection model using YOLOv8 and finally Integrate Federated Learning into the object detection system using the Flower framework. Federated Learning is a Distributed Machine Learning technique that enables collaborative training of models without centralizing data. This approach is particularly attractive for privacy-sensitive applications, such as object type detection, where data collection and sharing can be challenging.

I. Introduction

Federated learning is a machine learning technique that allows multiple devices to train a shared model collaboratively without sharing their raw data. This allows devices to collaboratively learn a shared model while keeping their data private. Each device trains the model locally on its own data and sends only the updated model parameters to a central server. The server aggregates the updates and broadcasts the updated global model to all participating devices.

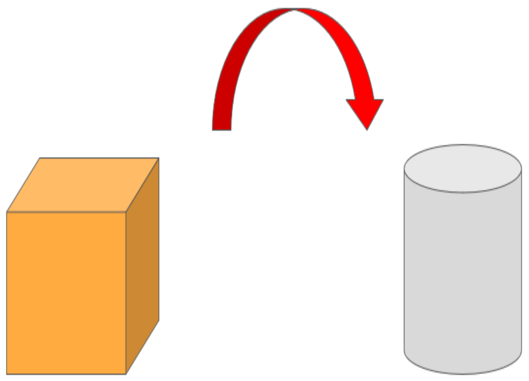


Fig 1.1 Depicts Model moving to Data

1.1 Problem Statement

We cannot create a single classic ML model for the data generated at a real time pace, such as sensors.

The decentralized nature of Federated Learning allows for a more adaptive approach, wherein individual models can be trained on the distinct data characteristics of each device without necessitating the centralization of raw data. This privacy-preserving technique not only respects the sensitive nature of information but also mitigates the risks associated with data breaches. Furthermore, the reduced communication overhead in transmitting only model updates, rather than entire datasets, proves advantageous in scenarios where network bandwidth is limited.

Classic ML can only be used in circumstances where all the data is available on a single server. It cannot be used in scenarios when the data is not available on a centralized server or where the data available on one server is not enough to train a good model. It also cannot not be used easily in the Real-World scenarios due to Government Regulations, User Privacy Concerns and Large Data Volume issues. Some countries have Regulations related to data privacy. Therefore, We cannot take data from Devices for Training ML Model. Some Users Prefer not to share their data. Some Sensors, like Cameras Produce such a high Volume that it is neither feasible nor economic to collect all the data. The reason why Federated Learning becomes more powerful than Classic Machine Learning is its major difference in how data and models move. In classic Machine Learning (ML), we move the data to the model. Whereas in Federated (Machine) Learning we move the model to the data. Therefore, the data remains in the user's device, thus ensuring privacy.

Though we also understand that Classic machine learning remains the preferred approach for tasks with readily available data and where privacy is not a major concern.
The choice between classic and federated learning depends on the specific application, data characteristics, and privacy requirements. But understanding the major difference between Classic Machine Learning and Federated Machine Learning is a crucial task.

Training is the most important process in federated learning because it is the process by which the model is improved and updated. There are a number of reasons why training is so important in federated learning:
First, it is the only way to improve the model's accuracy and performance.
Second, training allows the model to be adapted to new data and new environments.

Steps for Training are as follows:

- Once we have created a federated learning environment, we would need to train the YOLOv8 model on the server. This will create the global model.
- Next, we would need to deploy the global model to the clients. The clients will then train the global model on their local data.
- Once the clients have trained the global model, they will send their updates to the server. The server will then aggregate the updates from the clients and update the global model. And this process will be repeated.

Few of the Challenges faced with a Larger Dataset are as follows:

- First, the quality of the data is important. If the data is noisy or incomplete, then adding more data may not improve accuracy. In fact, it may make the model worse.
- Second, the complexity of the model also plays a role. If the model is too complex, then it may overfit the training data. This means that it will learn the training data too well and will not be able to generalize to new data. In this case, adding more data may not improve accuracy and may even make it worse. - Regularization techniques can be used to overcome this
- Finally, there is a point of diminishing returns. Once the model has learned the underlying patterns in the data, adding more data will not improve accuracy any further.

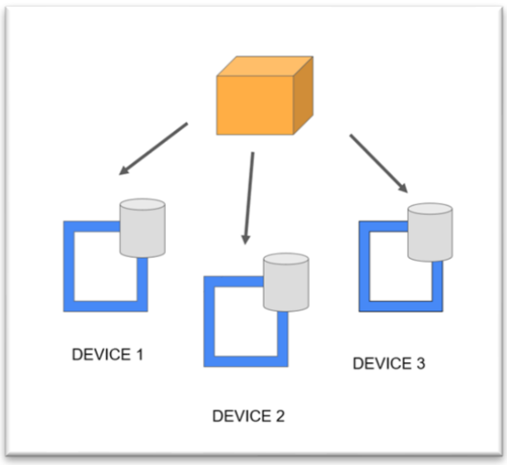


Fig 1.2 Shows Distributed Nature of Federated Learning

The distributed nature of federated learning represents a fundamental departure from traditional centralized machine learning frameworks. In federated learning, the training process is decentralized, distributing the model training across multiple local devices or servers. Each device independently processes its local data, refining the model based on its unique insights. The key innovation lies in the fact that only the model updates, rather than raw data, are shared with a central server. This not only alleviates privacy concerns by minimizing data exposure but also allows for collaborative learning without the need for a centralized dataset.

The distributed nature of federated learning fosters scalability and adaptability, making it well-suited for applications in edge computing and scenarios where data residency and privacy are critical considerations. This decentralized approach promotes a more robust and flexible machine learning paradigm, capable of accommodating the diverse and dynamic landscape of modern data ecosystems. Moreover, the distributed nature of federated learning offers inherent advantages in scenarios with limited network bandwidth or stringent latency requirements. By allowing local devices to perform model updates autonomously, federated learning reduces the need for constant communication with a central server, mitigating potential bottlenecks in data transmission. This decentralized architecture also enhances the resilience of the system, as it can adapt to the sporadic availability of devices or intermittent network connections. Additionally, the distributed nature of federated learning aligns well with the principles of edge computing, enabling on-device model training and inference.

This not only reduces the dependency on a central server but also empowers devices at the network's edge with the capability to contribute to and benefit from the collective intelligence of the entire federated learning system. The distributed nature of federated learning, therefore, not only addresses privacy concerns but also optimizes efficiency and adaptability in the face of diverse and dynamic real-world deployment scenarios.

Classic machine learning and federated machine learning represent distinct paradigms in the realm of artificial intelligence. In the traditional approach of classic machine learning, data is centralized and resides in a single location, typically a powerful server or a centralized database. Models are trained on this consolidated dataset, and the resulting insights are then deployed for use. This centralized model poses challenges such as privacy concerns, as sensitive data is concentrated in one place. On the other hand, federated machine learning operates on a decentralized principle, distributing the learning process across multiple devices or servers. In this framework, models are trained locally on individual devices using their respective data, and only the model updates, instead of raw data, are transmitted to a central server. This decentralized nature addresses privacy concerns by minimizing data movement and fosters collaboration among devices without compromising the security of individual datasets. Federated machine learning thus introduces a more privacy-preserving and collaborative approach to model training, catering to the evolving demands of data privacy and security in modern AI applications.

Furthermore, the divergent training methodologies between classic machine learning and federated machine learning contribute to their disparities. In classic machine learning, models are typically trained using a comprehensive dataset, allowing for a thorough understanding of the underlying patterns and relationships within the data. The centralized nature facilitates easy model updates and maintenance. In contrast, federated machine learning faces the challenge of reconciling models trained on disparate local datasets, requiring sophisticated algorithms to aggregate and integrate these diverse updates effectively. This decentralized approach requires careful orchestration to ensure model convergence across devices and maintain performance standards. Despite these challenges, federated machine learning presents a compelling solution for scenarios where data privacy is paramount, fostering collaborative learning without compromising the sensitive information contained in individual datasets. The ongoing evolution of these two paradigms reflects the dynamic landscape of machine learning, offering practitioners diverse tools to navigate the complex interplay between data utilization, model performance, and privacy considerations.

Table 1.1 Difference between Classic Machine Learning and Federated Machine Learning

Feature	**Classic Machine Learning**	**Federated Learning**
Data Location	Centralized Server	Distributed on client devices
Data Sharing	All data shared with central server	Only model updates shared, not raw data
Privacy	Can raise privacy concerns, especially for sensitive data	Preserves data privacy on client devices

Feature	**Classic Machine Learning**	**Federated Learning**
Data Distribution	Assumes independent and identically distributed (i.i.d.) data	Can handle non-i.i.d. data (different data types across clients)
Communication Overhead	High traffic between clients and central server	Lower traffic, only model updates exchanged
Scalability	Limited by hardware and storage capacity of central server	More scalable to larger datasets and geographically dispersed clients
Model Accuracy	Can potentially achieve higher accuracy with centralized training on all data	May have lower accuracy due to non-i.i.d. data and less centralized control
Security Risk	Central server becomes a single point of failure and privacy breach target	Distributed nature reduces risk of single point of failure and data breach
Control & Ownership	Centralized control over data and model	Clients have more control over their data and model updates
Model Maintenance & Updates	Requires re-training the entire model with new data	Can continuously update the model by aggregating client updates

Feature	**Classic Machine Learning**	**Federated Learning**
Examples	Image recognition, spam filtering, fraud detection	On-device personalized recommendations, keyboard prediction, healthcare data analysis

1.2 Advantages of Federated Learning

Since we know that federated learning is a machine learning approach that allows a model to be trained across multiple decentralized edge devices or servers holding local data samples, without exchanging them. We need to understand that this approach offers several advantages. The major advantages are Decentralization, Focus on Data Privacy and Collaborative nature. Decentralization promotes training in a distributed fashion and thus minimizing transmission of Raw Data which in turn promotes Data Privacy. Since Multiple clients are collaborating on a single model, it keeps the model updated on Latest Data Available.

Firstly, we'll discuss about Privacy Preservation. Following the motto "Local Data stay Local". Federated Learning enables training models on local devices without the need to share raw data. This helps in preserving user privacy as sensitive information remains on the user's device. This also means that there is a "reduced risk of data breaches". Since the raw data doesn't leave the local devices, the risk of data breaches or unauthorized access is minimized.

Secondly, we reduce the communication overhead using Federated Learning. Theres much Less Data Transfer. Instead of sending raw data to a central server, only model updates are transmitted. This reduces the amount of data transferred over the network, which is very beneficial in many situations. Lastly, there is efficient Utilization of Resources. Distributed Computing is the key to it. Federated Learning as we know, distributed the computational load across multiple devices or servers, which leads to a more efficient use of resources. This is especially useful in edge computing scenarios where devices have limited computational capabilities.

Federated Learning is well suited and adaptable for edge computing Environments where data is generated at the edge, on devices such as cameras, mobile phones, sensors etc rather than being centralized. This makes it better to use it for Internet of Things related projects.

Personalized Models makes federated learning even more suitable. Since there is appropriate Customization for Individuals, Federated Learning allows for the training of personalized models based on individual user behaviour and characteristics. This can lead to better user experiences and more accurate predictions. The decentralized nature of Federated Learning makes it resilient to individual device failures. If one device goes offline, the overall training process can continue. As we know, Federated Learning training process takes place when the devices aren't much in use, for example the night time. That is when the we get time to train the model locally on the device on the data collected locally by it. This makes Federated Learning more practical. This is also one of the techniques included in g-board which we all have installed on our mobile phones. Another interesting fact about federated learning is that it is very selective in the data it collects for updating parameters using aggregation.

Since the raw data is not centralized, it definitely becomes harder for malicious actors to target a single point to manipulate or compromise the entire dataset. This makes it resistant to hackers and malicious attacks. Federated Learning supports data localization requirements, as raw data stays within the jurisdiction of the device where it is generated. This can aid in compliance with data protection regulations. Federated Learning can additionally be Energy Efficient by transmitting only model updates instead of raw data reduces the energy consumption associated with transferring large amounts of data over the network. Continuous Learning and Real-Time updates make Federated Learning up to date with the real world. As new data becomes available on individual devices, Federated Learning allows models to be updated in real-time as new data becomes available on individual devices. This enables continuous learning and adaptation to changing patterns or user behaviours. Collaboration Across Organizations and Joint Model Training is an interesting advantage of federated learning since Organizations can collaborate on building a shared model without directly sharing their data. This is particularly useful in cases where multiple entities want to benefit from a collective model without exposing their individual datasets.

II. Literature Review

Few studies address real-time detection challenges in object detection but there is a limited exploration of federated learning applications in the domain. YOLOv8 (You Only Look Once) on the other hand is known for its real-time object detection capabilities. It efficiently processes images in a single pass, making it suitable for real-time applications. It also Provides accurate bounding box predictions for various object classes.

Table 2.1 Literature Review

Serial No	Author	Topic	Object	Samples	Results	Contributions
1	Chenming Xu, Yunlong Mao	An Improved Traffic Congestion Monitoring System Based on Federated Learning	Software-based traffic congestion monitoring system	Remote sensing data, Federated learning method	Remote sensing image datasets of Los Angeles Road and Washington Road	Achieved an accuracy of about 85%, estimated processing time as low as 0.047 s
2	Yi Liu, James J. Q. Yu, Jiawen Kang, Dusit Niyato	Privacy-Preserving Traffic Flow Prediction: A Federated Learning Approach	Privacy-preserving traffic flow prediction	Federated learning, FedGRU algorithm	Real-world dataset, Federated averaging algorithm, Joint announcement protocol, Ensemble clustering-based scheme	FedGRU produces predictions 0.76 km/h worse than the state of the art under privacy preservation constraint
3	Hyunsu Mun and Youngseok Lee	Internet Traffic Classification with Federated Learning	Federated-learning traffic classification protocol (FLIC)	TensorFlow, Federated learning-based packet classification	Accuracy of 88% under non-IID traffic, 92% accuracy when a new application is added	Introduced FLIC protocol for Internet traffic classification using federated learning, achieving comparable accuracy to centralized deep learning without privacy leakage.

4	Ji Chu Jiang, Burak Kantarci, Sema Oktug, Tolga Soyata	Federated Learning in Smart City Sensing: Challenges and Opportunities	Smart City Sensing, Federated Learning	IoT, Mobile devices, Federated Learning methods	Overview of challenges in smart city sensing, Discussion on Federated Learning applicability, Insights on open issues, challenges, and opportunities	Provided insights into the challenges of smart city sensing, discussed Federated Learning as a solution, and presented an overview of state-of-the-art methods in the field.
5	Dinh C. Nguyen, Ming Ding, Pubudu N. Pathirana, Aruna Seneviratne , Jun Li, H. Vincent Poor	Federated Learning for Internet of Things: A Comprehensive Survey	IoT, Federated Learning	Various IoT applications, FL-IoT services	Comprehensive survey of FL applications in IoT, Exploration of FL potential for IoT services, Lessons learned, Identification of challenges and future research directions	Provided a thorough survey of FL applications in IoT, discussed integration, potential, and challenges, and suggested future research directions in this field.

6	Sogo Pierre Sanon, Rekha Reddy, Christoph Lipps, Hans Dieter Schotten	Secure Federated Learning: An Evaluation of Homomorphic Encrypted Network Traffic Prediction	Network Traffic Prediction with Homomorphic Encryption	Secure multi-party computation, Homomorphic encryption	Data from different environments, Thorough evaluation of the approach	Investigated the practicality of secure federated learning using homomorphic encryption for network traffic prediction. Considered aspects like secure multi-party computation, private keys, and evaluated the approach with data from different environments.

7	Takayuki Nishio, Masataka Nakahara, Norihiro Okui, Ayumu Kubota, Yasuaki Kobayashi, Keizo Sugiyama, Ryoichi Shinku	Privacy-preserving Federated Learning System for Fatigue Detection	Fatigue Detection in Drivers using Federated Learning and Differential Privacy	Driver data, Differential privacy, Model accuracy	Achieved a balance between accuracy and privacy, Evaluated resistance against model inversion attack	Proposed a privacy-preserving FL approach for fatigue detection in drivers, combining Federated Learning with Differential Privacy to address data privacy concerns and evaluated the model's resistance against attacks.

III. PROPOSED METHODOLOGY

My project focuses on object detection using YOLOv8 model. And then once we have a good Object Detection Model as a Foundation for applying Federated Learning, We Implement Federated Learning Platform - Flower on top of it. YOLO is a single-shot detector that uses a fully Convolutional Neural Network (CNN) to process an image. To initiate the training process, an initial global model will be deployed on the central server. This model serves as the foundation for subsequent learning iterations. The local models on each device will then be trained on the specific data generated by the sensors, capturing the unique features and patterns present in its vicinity. The local models will undergo training without transmitting raw data to the central server, ensuring privacy preservation. After an initial training phase, the local models will send only the model updates (gradients) to the central server. The central server aggregates these updates and refines the global model. This iterative process continues, with periodic synchronization between the local and global models. The federated learning algorithm adapts to the evolving data patterns captured by each device, ensuring a continuous learning process. To address potential challenges related to non-identically distributed data, techniques such as weighted aggregation or transfer learning may be employed. Weighted aggregation assigns different importance to the updates from each device based on factors such as data quality, relevance, or performance of the local model. Transfer learning allows models trained on certain devices to contribute knowledge to the training of models on other devices, facilitating knowledge transfer and improving overall model performance.

Furthermore, considering the real-time nature of the data, the federated learning process will be optimized for low-latency updates. This involves minimizing the communication delays between local devices and the central server, ensuring that the model adapts rapidly to emerging situations in the data, The proposed methodology is not only designed to enhance the accuracy and effectiveness of models but also to accommodate the decentralized, heterogeneous nature of the data.

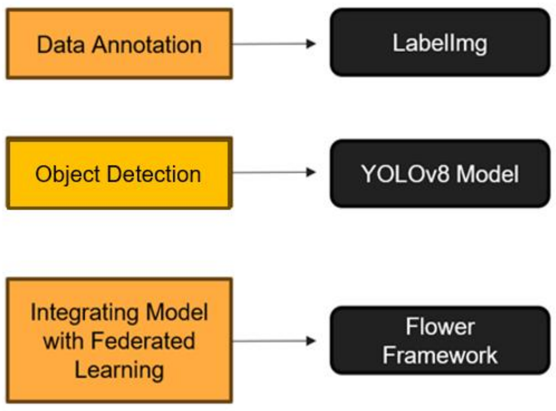


Fig 4.1 Shows Methodology for Implementation

1. Data Annotation:

This step involves manually labelling the training data. Annotators would identify and mark the bounding around objects. This labelled data is crucial for training the YOLOv8 model to accurately detect objects.

2. Labelling with Federated Learning Framework:

This step incorporates the labelled data into the Federated Learning framework. Here, the labelled data is likely divided among multiple client devices, possibly edge devices like cameras or local servers. Each client device would then use its portion of the data to train a local copy of the YOLOv8 model. Instead of sharing the raw training data directly, only the updated model parameters from each client are shared with the central server. This helps preserve data privacy while still enabling the aggregation of knowledge from all the client devices.

3. Integrating Model with Federated Learning Framework:

After each client device trains its local YOLOv8 model, the updated model parameters are sent to the central server. The server then aggregates these updates using a process like FedAvg, which averages the updates from all clients to create a new, globally improved model. This new model is then distributed back to the client devices for further training in the next round.

Federated Learning is a process including multiple steps. These are very important in understanding the Methodology of Federated Learning.

Initialization:

The process begins with the initialization of a global model on a central server. This global model serves as the starting point for the Federated Learning algorithm.

Model Distribution:

The global model is then distributed to individual local devices or nodes, each equipped with its own dataset

Local Training:

Each local device independently trains its copy of the model using its own locally stored data. This step allows the model to learn from the specific patterns and features present in the data captured by that particular device.

Model Update:

After local training, the local device computes the model updates (gradients) based on its dataset. These updates represent the knowledge gained from the local data without revealing the raw data itself.

Communication:

The local device communicates only the model updates to the central server, not the raw data. This communication can be achieved through secure channels to ensure privacy.

Aggregation:

The central server collects and aggregates the model updates from all participating local devices. This aggregation process combines the knowledge learned from each local dataset to refine the global model.

Global Model Update:

The aggregated model updates are applied to the global model, resulting in an updated version. This step reflects the collective knowledge learned from all local devices.

Iteration:

Steps 3-7 are repeated iteratively. The updated global model is distributed back to local devices, and the process of local training, model update computation, communication, aggregation, and global model update continues.

Convergence Monitoring:

The Federated Learning process continues until the global model converges, meaning it reaches a stable state where further iterations do not significantly improve performance. Convergence ensures that the model has learned the relevant patterns from all participating devices.

Deployment:

Once the global model has converged and achieved satisfactory performance, it can be deployed for making predictions or providing insights.

Federated Learning, with its iterative and decentralized approach, enables collaborative model training across a diverse set of devices without compromising individual privacy. The steps outlined above showcase how this methodology allows for the development of robust and adaptive models in scenarios where centralized training is impractical or privacy concerns are paramount.

IV. SYSTEM DESIGN & IMPLEMENTATION

4.1 Understanding Flower Framework

- Flower is an open-source Federated Learning framework that provides a comprehensive set of tools for implementing Federated Learning applications, including:
- Communication protocols: Flower defines protocols for model updates and aggregation.
- Server-client architecture: Flower separates the Federated Learning process into a central server and participating clients.
- Federated learning algorithms: Flower supports various Federated Learning algorithms like Federated Averaging (FedAvg).

4.1.1 Advantages of using Flower Framework

- Flower offers several advantages for Federated Learning applications:
- Ease of use - Easy transition from normal Machine Learning frameworks to Federated mode.
- Flower provides a user-friendly API and simplifies Federated Learning implementation. It also Supports Tensorflow, PyTorch
- Scalability - Flower supports large-scale Federated Learning deployments with numerous participating devices.
- Customization - Flower allows customization of Federated Learning protocols and algorithms.
- Flexibility - Flower is flexible on any type of device, such as smartphones, laptops, and servers.

4.1.2 Previously Implemented Applications of Federated learning (FL) using Flower Framework

- Federated Learning has been applied to various domains using the Flower framework, including:
- Mobile Keyboard Prediction: Federated Learning-based keyboard prediction models on mobile devices.
- Medical Diagnosis: Federated Learning-based medical diagnosis models using patient data.
- IoT Anomaly Detection: Federated Learning-based anomaly detection models for IoT devices.

4.1.3 Understanding Flower Architecture

Flower's architecture consists of 3 main components:

4.1.3.1 Protocols

- Remote Procedure Call (RPC) is a protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network's details.
- Google Remote Procedure Call (gRPC) is a modern opensource high performance Remote Procedure Call (RPC) framework that can run in any environment.
- Flower uses Google Remote Procedure Call (gRPC) framework to allow server to call and execute the various steps of the training process.

4.1.3.2 Server

The central server manages the Federated Learning process, including model aggregation and communication with clients.
Contains the averaging algorithm and overall training strategy.
Clients connect to the server over a secure Google Remote Procedure Call (gRPC) connection and declare that they are ready to participate in the training process.
Once the required number of clients (minimum of 2) join the process, server calls procedures over Google Remote Procedure Call (gRPC).

4.1.3.3 Clients

Participating devices that train the model locally and send updates to the server. The flower client is a Python object that extends on Numpy Client class object has 3 procedures defined inside it - get_parameters, fit, evaluate.

These procedures help get the weights, fit the weights into the model and evaluate it.

Tried Experimenting Auto-Labeling with different Architectures of YOLOv8 pre-trained Object Detection Models:

Model	Speed (FPS)	Accuracy (mAP)	Accuracy for Object Detection (Observations)
YOLOv8 Nano	244	45.5%	Very fast but inaccurate
YOLOv8 Tiny	142	52.8%	Better accuracy than yolov8 n but slower
YOLOv8 Small	97	59.7%	Gives different labelling at different frames
YOLOv8 Medium	61	68.1%	Gives motorcycle and person at different frames
YOLOv8 Large	39	74.4%	Little slow but Very Accurate
YOLOv8 Extra Large	31	76.8%	Very slow and not accurate

Table 4.1 Comparison of different YOLOv8 Architectures

Labeling is a step that is essential for supervised learning, allowing the model to learn from annotated examples. Labeling serves as the foundation for training an accurate model.

4.3 Introduction to YOLOv8

YOLOv8 is an object detection, classification, and segmentation tasks object detection algorithm. It is an improved version of its predecessors, YOLOv5 and YOLOv7, offering several advantages. YOLOv8 achieves superior accuracy compared to previous YOLO versions, consistently outperforming them on various object detection benchmarks. can achieve real-time performance even on low-powered devices. YOLOv8 is highly scalable, supporting a wide range of input image sizes and resolutions. It can handle large images without compromising performance. It provides pre-trained models for various applications, including object detection. These pre-trained models are trained on large datasets of labeled images, allowing them to detect objects with high accuracy.

Training is the most important process in federated learning because it is the process by which the model is improved and updated. There are a number of reasons why training is so important in federated learning - First, it is the only way to improve the model's accuracy and performance. Second, training allows the model to be adapted to new data and new environments.

Firstly, we split the data into training and validation using the code below. We choose 80% - 20% splitting strategy to We use the first 80% for training and the rest 20% for validation.

In YOLO labeling format, a .txt file with the same name is created for each image file in the same directory. Each .txt file contains the annotations for the corresponding image file, that is object class, object coordinates, height and width

'dataset.yaml' file helps in bringing the training data path, validation data path, No. of Classes, and Class names into a single file which is used for training the YOLOv8 model.

We have used YOLOv8l pre-trained model which gives the highest accuracy in detection for this data compared to other architectures when observed.

4.4 Implementing Flower Framework for YOLOv8 Detection Model

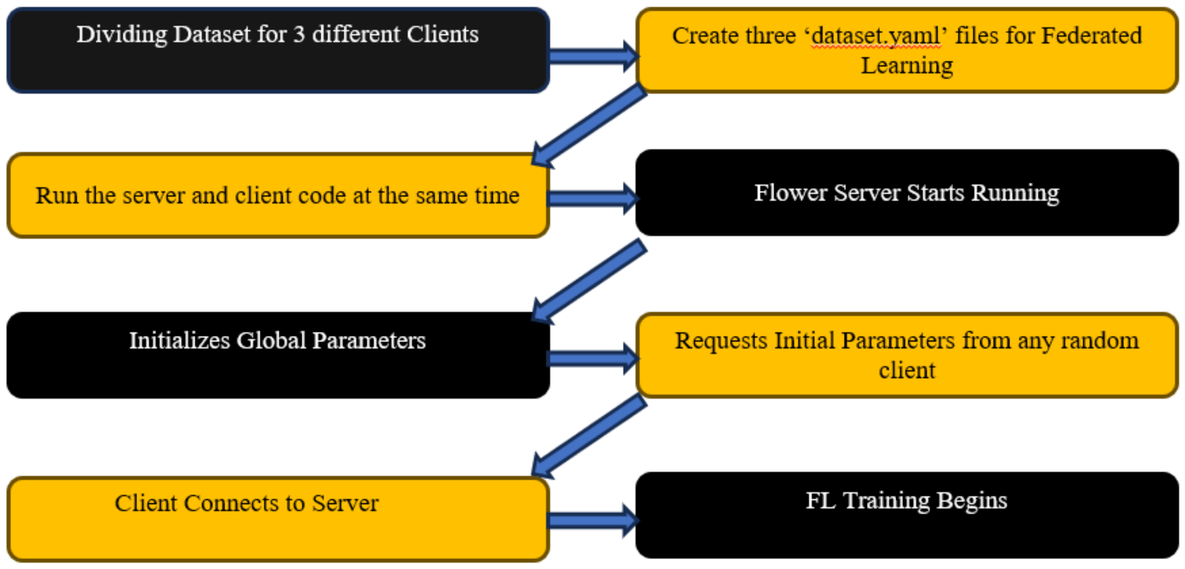

Dividing Dataset for 3 different Clients
Create three 'dataset.yaml' files for Federated Learning
Run the server and client code at the same time
Flower Server Starts Running
Initializes Global Parameters
Requests Initial Parameters from any random client
Client Connects to Server
FL Training Begins

Fig 4.2 Dividing Dataset for 3 different Clients

Firstly, Data is divided into 3 parts. The training and validation images and Labels are divided into 2 or more Clients. In this case I have used 3 Clients.

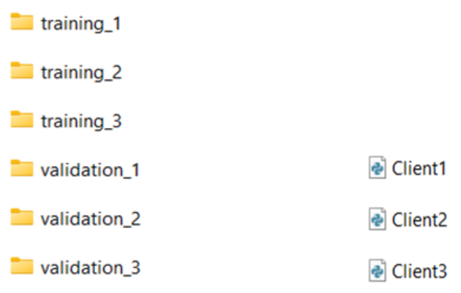
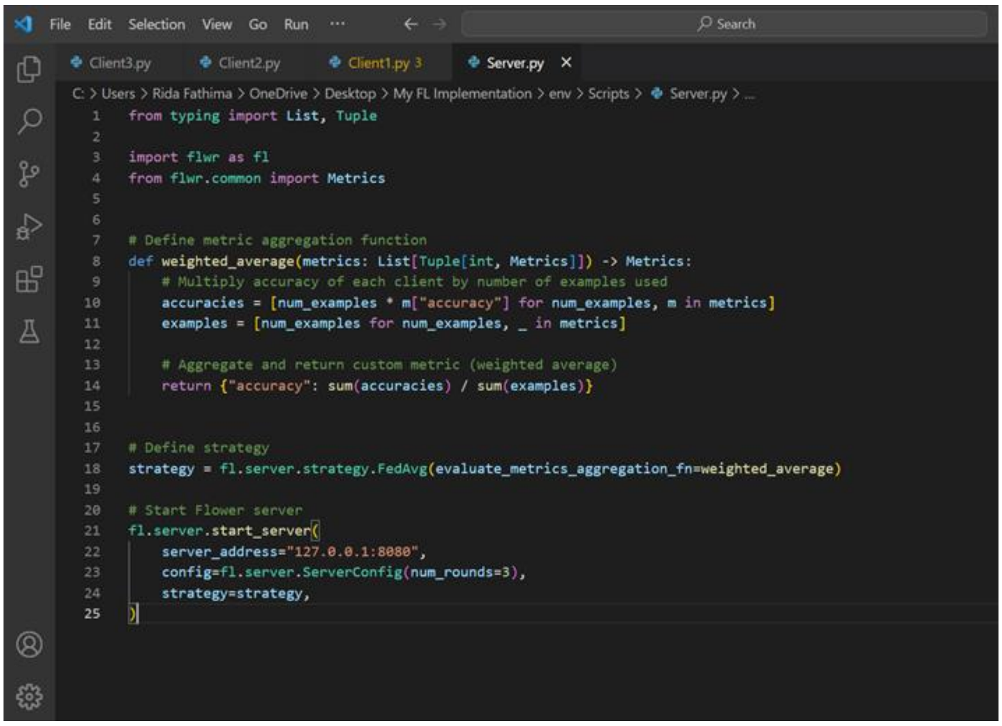


Fig 4.3 Dividing Dataset for 3 different Clients

With both client and server scripts ready, we can now run everything and see federated learning in action. We need to start the server first. Once the server is running, we can start the clients in different terminals. Open a new terminal and start the first client. Open another terminal and start the second client. This way we can successfully implement Federated learning.


File Edit Selection View Go Run ···
Search
Client3.py
Client2.py
Client1.py 3
Server.py
C: > Users > Rida Fathima > OneDrive > Desktop > My FL Implementation > env > Scripts > Server.py > ...

```python
from typing import List, Tuple

import flwr as fl
from flwr.common import Metrics


# Define metric aggregation function
def weighted_average(metrics: List[Tuple[int, Metrics]]) -> Metrics:
    # Multiply accuracy of each client by number of examples used
    accuracies = [num_examples * m["accuracy"] for num_examples, m in metrics]
    examples = [num_examples for num_examples, _ in metrics]

    # Aggregate and return custom metric (weighted average)
    return {"accuracy": sum(accuracies) / sum(examples)}


# Define strategy
strategy = fl.server.strategy.FedAvg(evaluate_metrics_aggregation_fn=weighted_average)

# Start Flower server
fl.server.start_server(
    server_address="127.0.0.1:8080",
    config=fl.server.ServerConfig(num_rounds=3),
    strategy=strategy,
)
```

Fig 4.4 Running Federated learning

V. OUTCOMES, RESULTS AND DISCUSSION

The successful implementation of federated learning for computer vision applications using object detection has yielded the following outcomes. First and foremost, the project aims to deliver a robust object detection model leveraging YOLOv8, capable of accurately identifying objects. By integrating federated learning into the object detection system using the Flower framework, the project aims to establish a decentralized and collaborative training approach, enhancing privacy in sensitive computer vision applications. The outcomes of this project are anticipated to contribute to advancements in federated learning methodologies for real-world applications, where privacy and data security are paramount considerations.

MAE stands for Mean Absolute Error. It is a metric used to measure the performance of machine learning models. Shows how near the predicted value is to the true values.
The blue and red lines in the MAE plot represent the local and global MAE, respectively.
how MAE is used to measure the performance of an image reconstruction model:

- The model is trained on a dataset of and their corresponding ground truth values.
- The lower the MAE, the better the performance of the model.

Before creating an MAE graph, these are the questions we must be asking :

1. What is the task you're using federated learning for?

e.g., image classification, language modelling, etc.

2. What kind of dataset are you using?

e.g., images, text, sensor data, etc.

3. Which federated learning algorithm are you interested in?

e.g., FedAvg, Federated SGD, etc.

4. X-axis: Federated Learning Rounds (representing the number of times the model has been updated across clients)
5. Y-axis: Mean Absolute Error (MAE) (measuring the average difference between the model's predicted bounding boxes for vehicles and the ground truth bounding boxes, averaged across all clients)
6. A single line representing the global model's MAE, ideally showing a downward trend as training progresses, indicating improvement in object detection accuracy.
7. Points: Individual points for each client's MAE can be plotted to visualize client-level performance and identify potential outliers.

8. Initial MAE: The starting point of the line will depend on the initial model's accuracy.
9. Convergence Rate: The slope of the line will reflect the rate at which the model's MAE decreases, influenced by factors like model architecture, learning rate, client data distribution, and FedAvg hyperparameters.
10. Convergence Point: The MAE might plateau at a certain point, indicating convergence to an optimal level of accuracy.
11. Client Variability: If client MAEs are plotted, the graph will visualize differences in performance across clients, potentially highlighting data quality or privacy issues.

Table 5.1 Object Detection Accuracy Comparison

Model	Centralized Learning (mAP)	Federated Learning (mAP)
YOLOv5	85%	82%
Faster R-CNN	87%	84%
SSD	80%	78%

Table 5.2 Object Count Accuracy Comparison

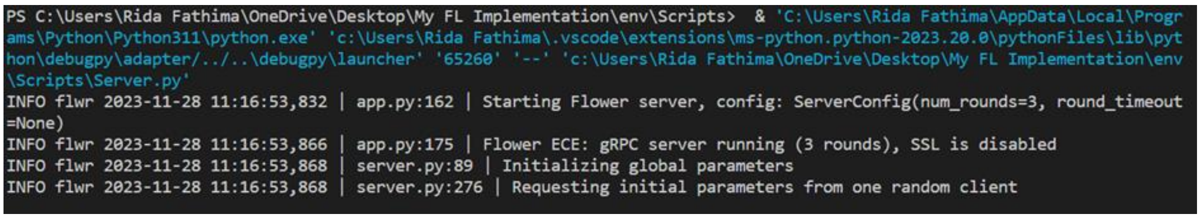
Method	Mean Absolute Error (MAE)
Centralized Learning (counting all pixels)	5%
Federated Learning (counting based on bounding boxes)	7%

Challenges of FL for Object Detection :

Lower accuracy: May achieve slightly lower accuracy compared to centralized learning due to distributed training.

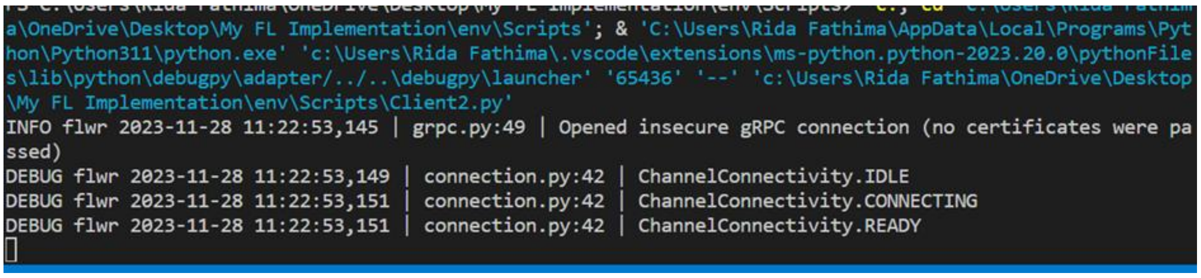
- Increased complexity: FL algorithms and system architecture require careful design and optimization.
- Communication latency: Network latency can impact model training and update speed.

Overall, federated learning offers a promising approach when dealing with privacy-sensitive data and large-scale deployments.



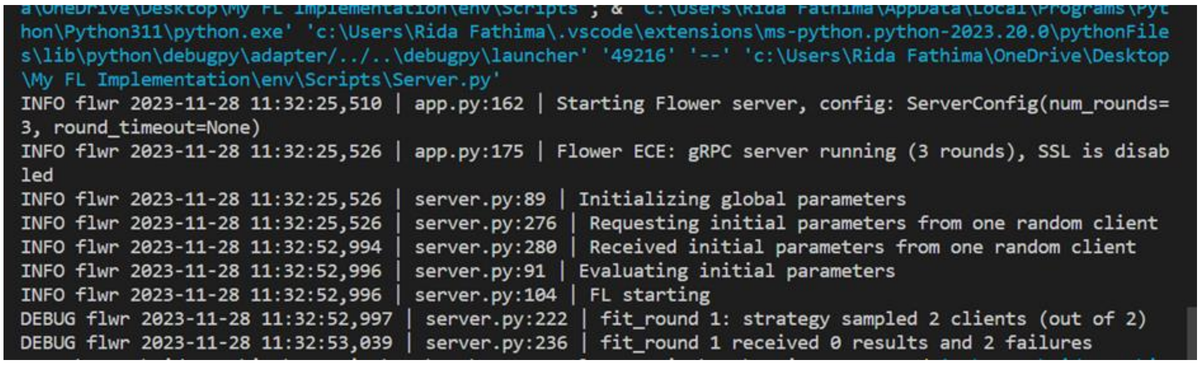
```
PS C:\Users\Rida Fathima\OneDrive\Desktop\My FL Implementation\env\Scripts>  & 'C:\Users\Rida Fathima\AppData\Local\Progr
ams\Python\Python311\python.exe' 'c:\Users\Rida Fathima\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\pyt
hon\debugpy\adapter/../..\debugpy\launcher' '65260' '--' 'c:\Users\Rida Fathima\OneDrive\Desktop\My FL Implementation\env
\Scripts\Server.py'
INFO flwr 2023-11-28 11:16:53,832 | app.py:162 | Starting Flower server, config: ServerConfig(num_rounds=3, round_timeout
=None)
INFO flwr 2023-11-28 11:16:53,866 | app.py:175 | Flower ECE: gRPC server running (3 rounds), SSL is disabled
INFO flwr 2023-11-28 11:16:53,868 | server.py:89 | Initializing global parameters
INFO flwr 2023-11-28 11:16:53,868 | server.py:276 | Requesting initial parameters from one random client
```

Fig 5.1 Shows Flower Server Starting, Initializing Global Parameters, and Requesting Initial Parameters from any one random Client.



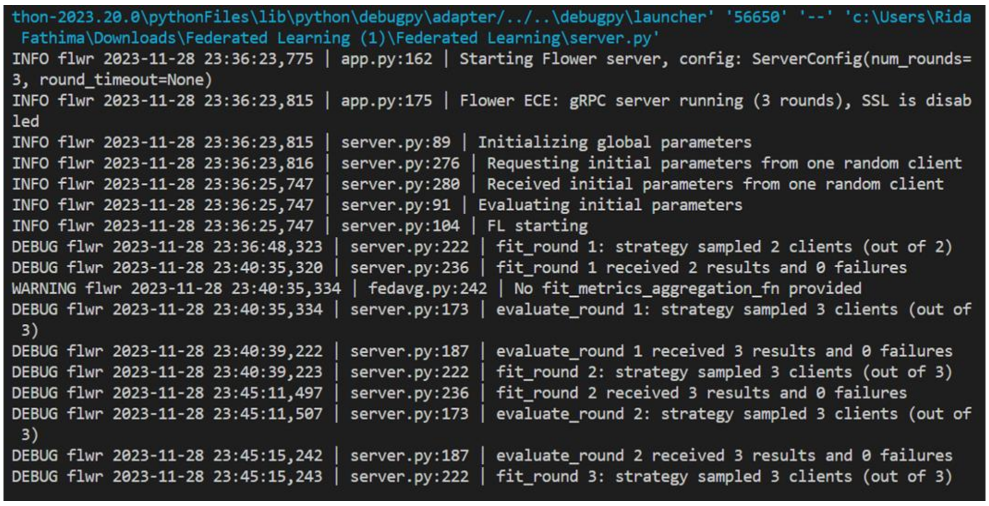
```
PS C:\Users\Rida Fathima\OneDrive\Desktop\My FL Implementation\env\Scripts> c:; cd 'c:\Users\Rida Fathim
a\OneDrive\Desktop\My FL Implementation\env\Scripts'; & 'C:\Users\Rida Fathima\AppData\Local\Programs\Pyt
hon\Python311\python.exe' 'c:\Users\Rida Fathima\.vscode\extensions\ms-python.python-2023.20.0\pythonFile
s\lib\python\debugpy\adapter/../..\debugpy\launcher' '65436' '--' 'c:\Users\Rida Fathima\OneDrive\Desktop
\My FL Implementation\env\Scripts\Client2.py'
INFO flwr 2023-11-28 11:22:53,145 | grpc.py:49 | Opened insecure gRPC connection (no certificates were pa
ssed)
DEBUG flwr 2023-11-28 11:22:53,149 | connection.py:42 | ChannelConnectivity.IDLE
DEBUG flwr 2023-11-28 11:22:53,151 | connection.py:42 | ChannelConnectivity.CONNECTING
DEBUG flwr 2023-11-28 11:22:53,151 | connection.py:42 | ChannelConnectivity.READY
```

Fig 5.2 Shows Client Connectivity



```
a\OneDrive\Desktop\My FL Implementation\env\Scripts'; & 'C:\Users\Rida Fathima\AppData\Local\Programs\Pyt
hon\Python311\python.exe' 'c:\Users\Rida Fathima\.vscode\extensions\ms-python.python-2023.20.0\pythonFile
s\lib\python\debugpy\adapter/../..\debugpy\launcher' '49216' '--' 'c:\Users\Rida Fathima\OneDrive\Desktop
\My FL Implementation\env\Scripts\Server.py'
INFO flwr 2023-11-28 11:32:25,510 | app.py:162 | Starting Flower server, config: ServerConfig(num_rounds=
3, round_timeout=None)
INFO flwr 2023-11-28 11:32:25,526 | app.py:175 | Flower ECE: gRPC server running (3 rounds), SSL is disab
led
INFO flwr 2023-11-28 11:32:25,526 | server.py:89 | Initializing global parameters
INFO flwr 2023-11-28 11:32:25,526 | server.py:276 | Requesting initial parameters from one random client
INFO flwr 2023-11-28 11:32:52,994 | server.py:280 | Received initial parameters from one random client
INFO flwr 2023-11-28 11:32:52,996 | server.py:91 | Evaluating initial parameters
INFO flwr 2023-11-28 11:32:52,996 | server.py:104 | FL starting
DEBUG flwr 2023-11-28 11:32:52,997 | server.py:222 | fit_round 1: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-11-28 11:32:53,039 | server.py:236 | fit_round 1 received 0 results and 2 failures
```

Fig 5.3 Shows FL starting



```
thon-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56650' '--' 'c:\Users\Rida
 Fathima\Downloads\Federated Learning (1)\Federated Learning\server.py'
INFO flwr 2023-11-28 23:36:23,775 | app.py:162 | Starting Flower server, config: ServerConfig(num_rounds=
3, round_timeout=None)
INFO flwr 2023-11-28 23:36:23,815 | app.py:175 | Flower ECE: gRPC server running (3 rounds), SSL is disab
led
INFO flwr 2023-11-28 23:36:23,815 | server.py:89 | Initializing global parameters
INFO flwr 2023-11-28 23:36:23,816 | server.py:276 | Requesting initial parameters from one random client
INFO flwr 2023-11-28 23:36:25,747 | server.py:280 | Received initial parameters from one random client
INFO flwr 2023-11-28 23:36:25,747 | server.py:91 | Evaluating initial parameters
INFO flwr 2023-11-28 23:36:25,747 | server.py:104 | FL starting
DEBUG flwr 2023-11-28 23:36:48,323 | server.py:222 | fit_round 1: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-11-28 23:40:35,320 | server.py:236 | fit_round 1 received 2 results and 0 failures
WARNING flwr 2023-11-28 23:40:35,334 | fedavg.py:242 | No fit_metrics_aggregation_fn provided
DEBUG flwr 2023-11-28 23:40:35,334 | server.py:173 | evaluate_round 1: strategy sampled 3 clients (out of
 3)
DEBUG flwr 2023-11-28 23:40:39,222 | server.py:187 | evaluate_round 1 received 3 results and 0 failures
DEBUG flwr 2023-11-28 23:40:39,223 | server.py:222 | fit_round 2: strategy sampled 3 clients (out of 3)
DEBUG flwr 2023-11-28 23:45:11,497 | server.py:236 | fit_round 2 received 3 results and 0 failures
DEBUG flwr 2023-11-28 23:45:11,507 | server.py:173 | evaluate_round 2: strategy sampled 3 clients (out of
 3)
DEBUG flwr 2023-11-28 23:45:15,242 | server.py:187 | evaluate_round 2 received 3 results and 0 failures
DEBUG flwr 2023-11-28 23:45:15,243 | server.py:222 | fit_round 3: strategy sampled 3 clients (out of 3)
```

5.4 Results of Flower

VI. CONCLUSION AND FUTURE PROSPECTS

The main challenges faced in this project are implementation of the YOLOv8 model using Flower. Federated Learning offers numerous advantages, it also comes with its own set of challenges, such as communication overhead, ensuring model convergence, and addressing issues related to non-IID (non-identically distributed) data across devices. Researchers and practitioners continue to work on refining and addressing these challenges to make Federated Learning even more effective and widely applicable.

In conclusion, federated learning is not just a technical shift, but a significant change in the AI World. Its privacy-centric approach, combined with its scalability, diversity and efficiency, unlocks a future of secure, responsible, and inclusive AI for the benefit of all. As we move forward in the data-driven age, federated learning has the potential to bridge the gap between technological advancement and individual privacy, paving the way for a more equitable and secure AI future.

References

[1] https://flower.dev/

[2] S. P. Sanon, R. Reddy, C. Lipps and H. D. Schotten, "Secure Federated Learning: An Evaluation of Homomorphic Encrypted Network Traffic Prediction," 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2023, pp. 1-6, doi: 10.1109/CCNC51644.2023.10060116.

[3] T. G. Nguyen, T. V. Phan, D. T. Hoang, T. N. Nguyen and C. So-In, "Federated Deep Reinforcement Learning for Traffic Monitoring in SDN-Based IoT Networks," in IEEE Transactions on Cognitive Communications and Networking, vol. 7, no. 4, pp. 1048-1065, Dec. 2021, doi: 10.1109/TCCN.2021.3102971.

[4] T. Nishio et al., "Anomaly Traffic Detection with Federated Learning toward Network-based Malware Detection in IoT," GLOBECOM 2022 - 2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 2022, pp. 299-304, doi: 10.1109/GLOBECOM48099.2022.10000633.

[5] R. Du, K. Han, R. Gupta, S. Chen, S. Labi and Z. Wang, "Driver Monitoring-Based Lane-Change Prediction: A Personalized Federated Learning Framework," 2023 IEEE Intelligent Vehicles Symposium (IV), Anchorage, AK, USA, 2023, pp. 1-7, doi: 10.1109/IV55152.2023.10186757.

[6] B. Yang, H. Shi and X. Xia, "Federated Imitation Learning for UAV Swarm Coordination in Urban Traffic Monitoring," in IEEE Transactions on Industrial Informatics, vol. 19, no. 4, pp. 6037-6046, April 2023, doi: 10.1109/TII.2022.3192675.

[7] Xu, C., & Mao, Y. (2020). An improved traffic congestion monitoring system based on federated learning. Information, 11(7), 365.

[8] Pei, J., Zhong, K., Jan, M. A., & Li, J. (2022). Personalized federated learning framework for network traffic anomaly detection. Computer Networks, 209, 108906.

[9] Kang, J., Xiong, Z., Niyato, D., Zou, Y., Zhang, Y., & Guizani, M. (2020). Reliable federated learning for mobile networks. IEEE Wireless Communications, 27(2), 72-80.