FEDERATED LEARNING FOR TRAFFIC MONITORING

A PROJECT REPORT

Submitted by,

Ms. Rida Fathima - 20201CBC0042

Under the guidance of,

Dr. Srinivasan T R

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (BLOCKCHAIN)

At

PRESIDENCY UNIVERSITY, BENGALURU

JANUARY 2024

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report **"FEDERATED LEARNING FOR TRAFFIC MONITORING"** being submitted by "RIDA FATHIMA" bearing roll number "20201CBC0042" in partial fulfillment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering (Blockchain) is a bonafide work carried out under my supervision.

Dr. Srinivasan T R
Project Supervisor,
Professor
School of CSE&IS
Presidency University

Dr. S Senthilkumar
Professor & HOD
School of CSE&IS
Presidency University

Dr. C. KALAIARASAN
Associate Dean
School of CSE&IS
Presidency University

Dr. L. SHAKKEERA
Associate Dean
School of CSE&IS
Presidency University

Dr. MD. SAMEERUDDIN KHAN
Dean
School of CSE&IS
Presidency University

Name and Signature of the Examiners

1)

2)

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **FEDERATED LEARNING FOR TRAFFIC MONITORING** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering (Blockchain)**, is a record of our own investigations carried under the guidance of **Dr. Srinivasan TR, Professor, School of Computer Science and Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Rida Fathima
20201CBC0042

ABSTRACT

The overall goal of our project is to implement federated learning for traffic monitoring using vehicle detection and counting across CCTV video footage collected by SP8. The main objective is to identify and count different vehicle types, such as cars, trucks, motorcycles, etc. The data collected from CCTV belongs to the Chiayi government. The objective is to Develop a robust vehicle detection model using YOLOv8 and finally Integrate Federated Learning into the vehicle 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 vehicle type detection, where data collection and sharing can be challenging.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. Kalaiarasan C and Dr. Shakkeera L,** School of Computer Science Engineering & Information Science, Presidency University and Dr. S Senthilkumar, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Dr. Srinivasan T R, Professor**, School of Computer Science Engineering & Information Science, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators **Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS** and also the department Project Coordinators **Dr. Srinivasan T R** .

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Rida Fathima

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER-1
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.

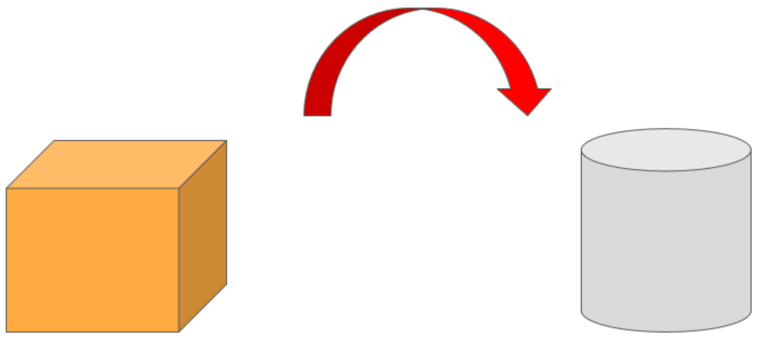


Fig 1.1 Depicts Model moving to Data

Let us understand Federated Learning using an Analogy.

- Imagine a group of students who are working on a class project.
- Each student has their own set of data, and they need to work together to create a report.
- Instead of sharing their data with each other, they each create their own report based on their own data.
- Then, they send their reports to a teacher, who combines them into a single report.
- The teacher then sends the combined report back to the students, and they can all review it and make suggestions.
- This process is repeated until the report is complete.

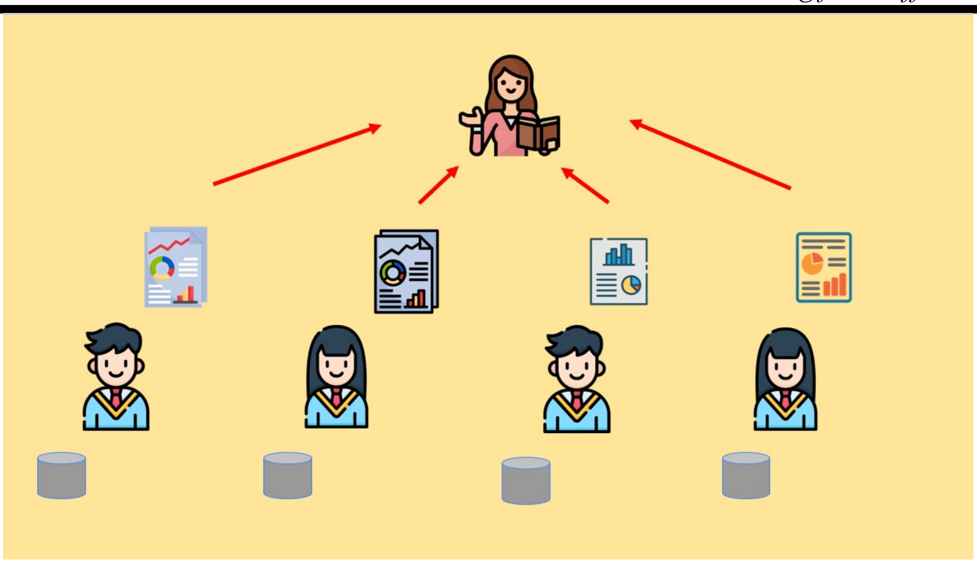


Fig 1.2 Federated Learning analogy of Student and Teacher

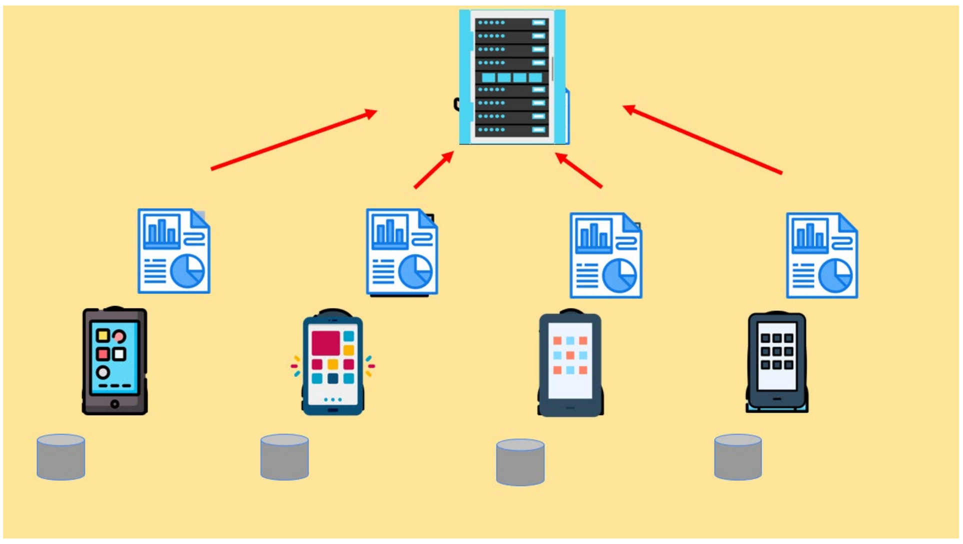


Fig 1.3 Federated Learning devices, data, and Central Server

1.1 Problem Statement

Using Federated Learning to solve a Real Time Problem with **Real Time data** taken from Chiayi County CCTV Footages. **Federated Learning** is used because we cannot create a single classic ML model for the whole city since the data on each CCTV varies from one another. This is because it is a big County.

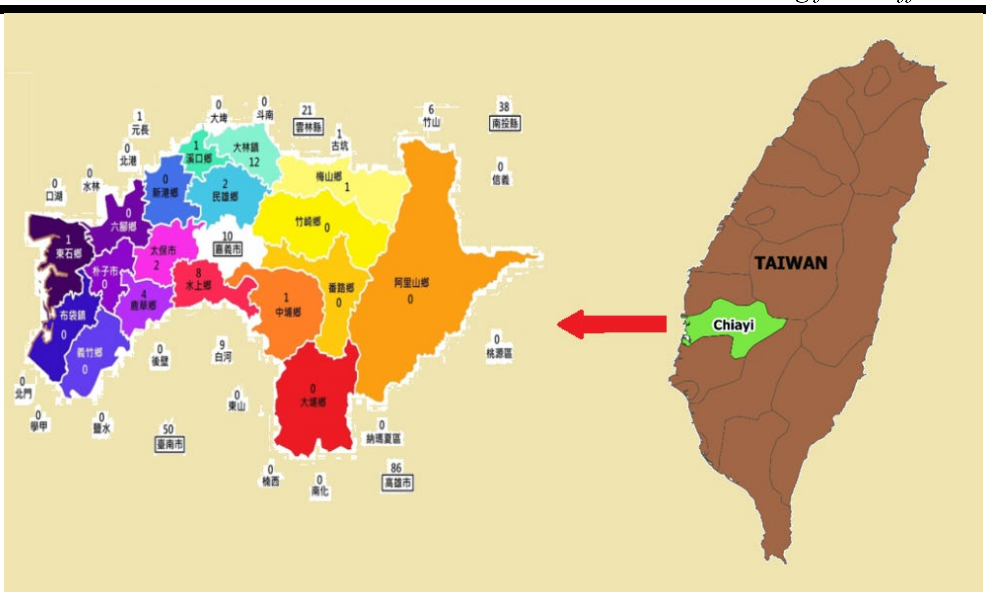


Fig 1.4 Chiayi Country Representation

In addressing the dynamic challenges posed by the surveillance footage from various CCTV cameras scattered across the expansive Chiayi County, the implementation of Federated Learning emerges as a crucial solution. The enormity and diversity of the county make it impractical to construct a singular conventional machine learning model that can effectively encapsulate the varied data streams from each CCTV device. 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 camera without necessitating the centralization of raw data. This privacy-preserving technique not only respects the sensitive nature of surveillance 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. By embracing Federated Learning, the aim is to derive actionable insights from real-time CCTV footage, fostering a collaborative and decentralized model that seamlessly adapts to the unique attributes of each location within Chiayi County. This approach not only enhances the accuracy of predictions but also ensures continuous learning and adaptation to evolving scenarios captured by the extensive network of surveillance cameras distributed throughout the county.

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.

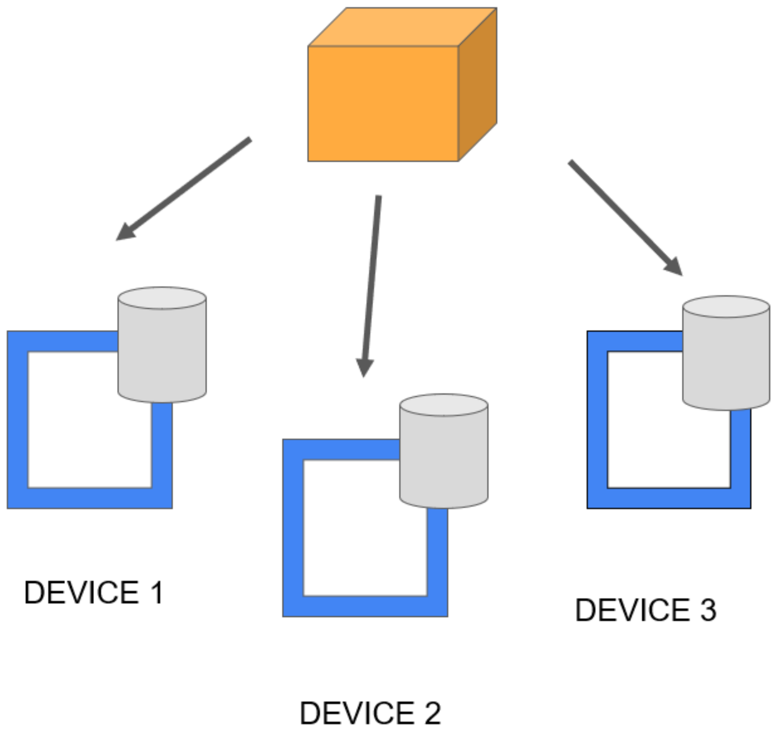


Fig 1.5 Shows Distributed Nature of Federated Learning

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
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

Feature	**Classic Machine Learning**	**Federated Learning**
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
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 like cameras.

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.

CHAPTER-2

LITERATURE SURVEY

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.

8	Not provided	Differentially Private Online Federated Learning With Personalization and Fairness	Federated Learning with Differential Privacy	Personalization, Fairness, Regret bounds, Simulation result	Proposed DIPOFEL and DIPOFLAIR Algorithms, Addressed challenges of centralized server congestion, time-varying data distribution, task specificity, and unfairness.	Introduced DIPOFEL and DIPOFLAIR Algorithms for online federated learning with differential privacy, addressing challenges in personalized task optimization and fairness in a peer-to-peer federated learning setting.
9	Not provided	A Survey on Attack Detection and Resilience for Connected and Automated Vehicles: From Vehicle Dynamics and Control Perspective	Attack Detection and Resilience for Connected and Automated Vehicles (CAVs)	Vehicle dynamics, Control frameworks, Attack and anomaly detection, Resilience strategies	Not specified	Reviewed recent advances in attack/anomaly detection and resilience strategies for CAVs from a vehicle dynamics and control perspective. Identified potential research directions and challenges.

10	Not provided	Pothole Detection on Urban Roads Using YOLOv8	Pothole detection on urban roads using YOLOv8	Urban road images, Pothole detection, YOLOv8 algorithm	Transfer learning, Average precision of 0.92, Recall of 0.89	Proposed a YOLOv8-based approach for efficient pothole detection on urban roads, achieving high precision and recall. Outperformed state-of-the-art object detection algorithms in terms of accuracy and speed.
11	Not provided	An improved fire detection approach based on YOLO-v8 for smart cities	Fire detection in smart cities using YOLOv8	Smart fire detection system (SFDS), YOLOv8 algorithm, Deep learning	Fog and Cloud computing, IoT layer, Precision rate of 97.1% for all classes	Proposed SFDS for real-time fire detection in smart cities using YOLOv8, leveraging Fog and Cloud computing, achieving state-of-the-art perf

CHAPTER-3
RESEARCH GAPS OF EXISTING METHODS

Federated learning in traffic monitoring is an evolving field with several ongoing research efforts. Previous work on vehicle detection demonstrates robustness against challenging weather and lighting conditions. But there is also a lack of consistency in performance across different datasets.

Few studies address real-time detection challenges in dynamic traffic scenarios 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, including vehicles. But there may be challenges in detecting small or distant vehicles accurately and sensitivity to variations in lighting conditions, affecting performance in diverse environments.

Here are some potential research gaps in existing methods of implementing federated learning in traffic monitoring:

Since the data is distributed across multiple edge devices, making sure of optimizing communication and robustness between edge devices and the central server to reduce the overall communication overhead is very important.

Heterogeneity in Edge Devices is another research gap found. It is important to investigate methods to handle the diversity of data collected in different Cameras used in traffic monitoring. Heterogeneity in data can cause issues in the training of the local devices causing changes in overall training result parameters given to the central server.

Examining the impact of edge device failures or disconnections on the federated learning process is another gap identified in existing methods.

In addition to the aforementioned challenges and research gaps in federated learning for traffic monitoring, another critical aspect is the consideration of privacy preservation. Given the sensitive nature of traffic-related data, ensuring robust privacy-preserving mechanisms becomes imperative. Federated learning inherently offers a decentralized approach that minimizes the need to share raw data with a central server. However, it is crucial to delve into advanced techniques for encryption, differential privacy, or secure multi-party computation to safeguard the privacy of information exchanged during the federated learning process. This becomes particularly pertinent as traffic monitoring systems often involve the collection of personally identifiable information and real-time location data. Striking a balance between model accuracy and individual privacy is vital for the widespread adoption of federated learning in traffic monitoring applications. Addressing these multifaceted challenges requires collaborative efforts from researchers, practitioners, and policymakers to establish comprehensive frameworks that not only enhance the efficiency and accuracy of traffic monitoring systems but also uphold the ethical principles of data privacy and security.

CHAPTER-4
PROPOSED METHODOLOGY

My project focuses on vehicle type detection and counting of vehicle types using YOLOv8 model. And then once we have a good Vehicle Detection Model as a Foundation for applying Federated Learning, We Implement Federated Learning Platform - Flower on top of it.
Dataset: A dataset of traffic scenes, images or videos with labeled vehicle information is very important. This dataset should reflect the diversity of real-world traffic scenarios.
Model Architecture: The model used for traffic tracking should be designed to handle real-time data and exhibit good accuracy. Convolutional Neural Networks (CNNs) are commonly used for computer vision tasks, but their architecture may need to be customized for traffic tracking. YOLO is a single-shot detector that uses a fully Convolutional Neural Network (CNN) to process an image.
Security and Privacy: Given the sensitivity of traffic data, strong security and privacy measures must be in place to protect data during the Federated Learning process.

The proposed methodology for implementing Federated Learning in the context of Chiayi County's real-time surveillance data involves multiple steps designed to harness the distributed nature of the CCTV network effectively. Firstly, a federated learning framework will be established, comprising a central server and the individual CCTV cameras distributed across various locations. Each camera will act as a decentralized node contributing to the collaborative learning process.

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 CCTV device will then be trained on the specific data generated by that camera, 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 CCTV device, ensuring a continuous learning process.

To address potential challenges related to non-identically distributed data across cameras, techniques such as weighted aggregation or transfer learning may be employed. Weighted aggregation assigns different importance to the updates from each camera based on factors such as data quality, relevance, or performance of the local model. Transfer learning allows models trained on certain cameras to contribute knowledge to the training of models on other cameras, facilitating knowledge transfer and improving overall model performance.

Furthermore, considering the real-time nature of the surveillance 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 surveillance footage.

The proposed methodology is not only designed to enhance the accuracy and effectiveness of the surveillance models but also to accommodate the decentralized, heterogeneous nature of the CCTV data in Chiayi County. By leveraging Federated Learning, the project aims to create a collaborative and adaptive system capable of providing real-time insights while respecting privacy, addressing challenges related to diverse data sources, and ensuring efficient and timely updates to the surveillance models deployed across the county.

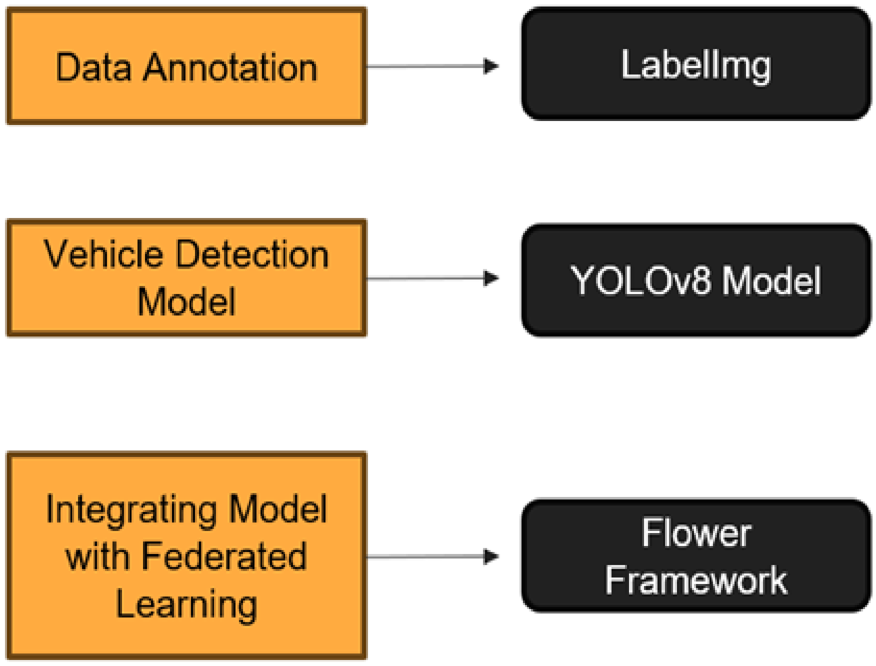


Fig 4.1 Shows the Models, Software and Frameworks used in the Project

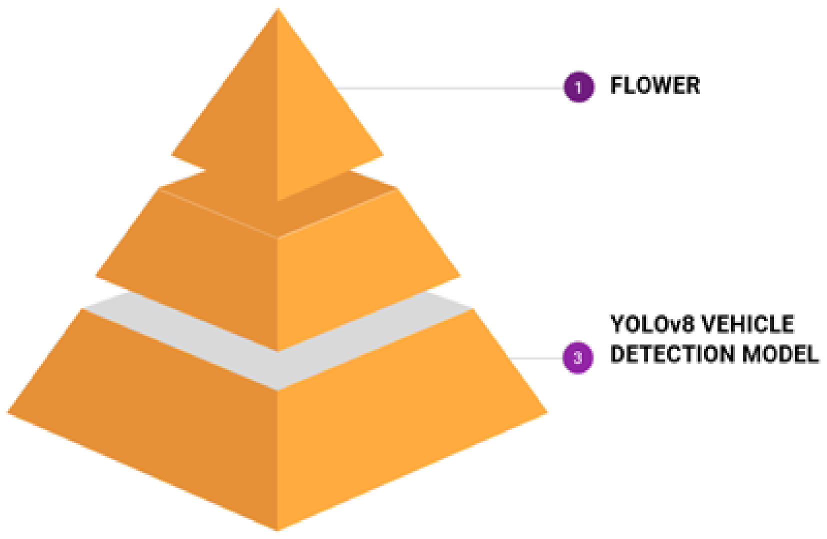


Fig 4.2 Shows Implementation of Federated Learning on top of YOLOv8 model

1. Data Annotation:

This step involves manually labelling the training data, which in this case, would be CCTV camera footage. Annotators would identify and mark the bounding boxes around each vehicle present in each frame of the video footage. This labelled data is crucial for training the YOLOv8 model to accurately detect vehicles in unseen CCTV footage.

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. In the context of Chiayi County's CCTV project, each surveillance camera would be a local device.

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. In the case of Chiayi County's CCTV project, the deployed model can analyse real-time surveillance footage to detect patterns, anomalies, or other relevant information.

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.

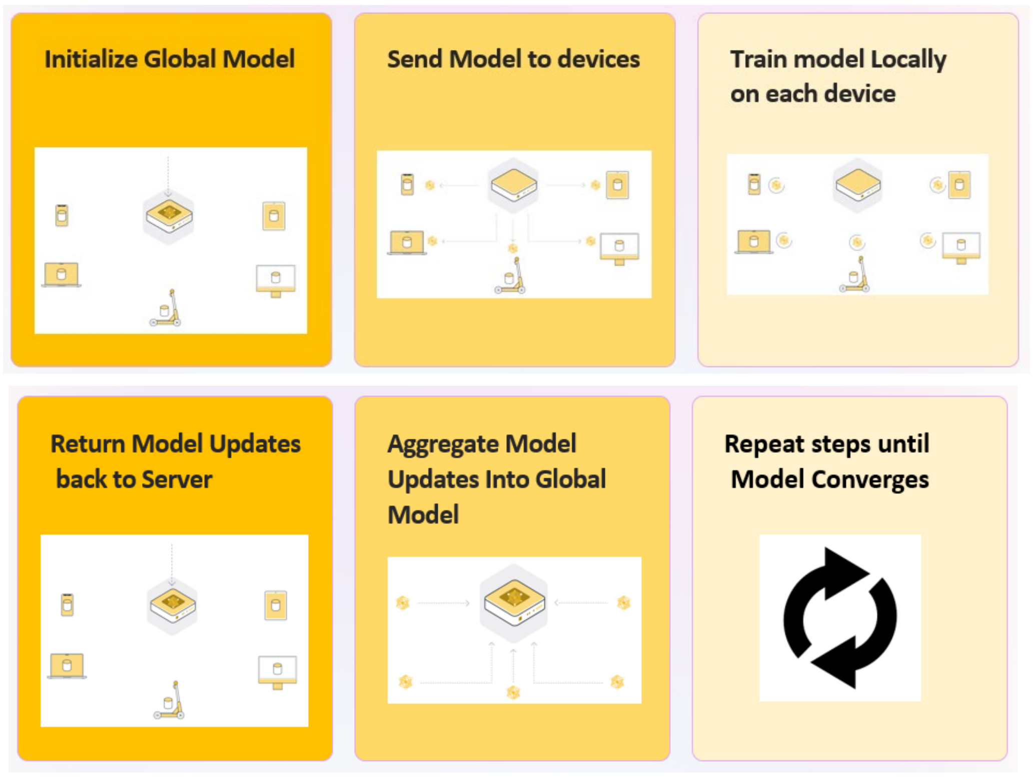

Initialize Global Model
Send Model to devices
Train model Locally on each device
Return Model Updates back to Server
Aggregate Model Updates Into Global Model
Repeat steps until Model Converges

Fig 4.3 Steps of Federated Learning

CHAPTER-5
OBJECTIVES

Enhancing YOLOv8 Model Performance: Investigate and implement modifications to YOLOv8 to improve small and distant vehicle detection, especially in challenging environmental conditions. Federated Learning Integration: Integrate Flower as a federated learning platform on top of the enhanced YOLOv8 model for collaborative model training while preserving data privacy. Optimizing Communication Overhead: Develop strategies to minimize communication overhead in federated learning, ensuring efficient model updates between devices without compromising accuracy.
Benchmarking Federated Learning for Vehicle Type Detection: Establish standardized benchmarks and evaluation metrics for federated learning in the context of vehicle type detection, facilitating comparisons and advancements in the field.

● Heterogeneous Data Handling: Devise methodologies to handle the diversity of data collected from different cameras used in traffic monitoring. As cameras may vary in resolution, field of view, and calibration, addressing heterogeneity ensures the robustness and generalizability of the federated learning model.

● Privacy-Preserving Techniques: Implement advanced privacy-preserving techniques such as differential privacy, secure multi-party computation, or homomorphic encryption to safeguard sensitive data during the federated learning process. Ensuring individual privacy is crucial, especially when dealing with real-time location and personally identifiable information.

● Edge Device Failure Resilience: Explore mechanisms to handle edge device failures or disconnections during federated learning. Robust strategies should be devised to manage interruptions in communication without compromising the overall efficiency and convergence of the federated learning process.

● Adaptive Learning Rates: Investigate adaptive learning rate strategies to optimize model training in federated learning. Adjusting learning rates dynamically based on the performance of individual devices can contribute to faster convergence and improved overall model accuracy.

● Cross-Domain Federated Learning: Explore the feasibility and challenges of extending federated learning across different domains of traffic monitoring. Understanding how federated learning models can adapt to diverse scenarios and data distributions enhances the versatility of the approach.

● Energy-Efficient Federated Learning: Develop techniques to optimize the energy consumption of edge devices during federated learning. As energy efficiency becomes a critical consideration for devices with limited resources, minimizing the computational burden is essential.

● Dynamic Model Aggregation: Investigate dynamic model aggregation techniques that adaptively weight model updates based on the relevance and quality of the contributed information from each device, improving the overall federated learning performance.

● Edge-Based Anomaly Detection: Implement edge-based anomaly detection mechanisms to identify and mitigate the impact of outliers or malicious contributions during federated learning, enhancing the security and reliability of the collaborative training process.

● Federated Learning for Traffic Flow Prediction: Extend the application of federated learning to traffic flow prediction, allowing for more comprehensive insights into urban mobility patterns and optimizing traffic management strategies.

● Edge Device Resource Allocation: Develop strategies for efficient resource allocation on edge devices during federated learning, considering factors such as memory constraints, processing power, and storage limitations.

● Multi-Modal Federated Learning: Extend federated learning to incorporate multi-modal data sources, such as combining video and sensor data, to enhance the robustness and richness of the model's understanding of traffic scenarios.

● Adaptive Compression Techniques: Investigate adaptive model compression techniques to reduce the size of model updates transmitted between devices, minimizing communication overhead while maintaining model accuracy.

● Federated Learning for Anonymized Traffic Studies: Explore the application of federated learning for anonymized traffic studies, allowing for comprehensive analysis without compromising individual privacy.

● Real-Time Model Evaluation: Implement mechanisms for real-time evaluation of federated learning models during the training process, enabling early detection of performance issues and facilitating timely adjustments.

CHAPTER-6
SYSTEM DESIGN & IMPLEMENTATION

6.1 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).

6.1.1 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.

6.1.2 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.

6.1.3 Understanding Flower Architecture

Flower's architecture consists of 3 main components:

6.1.3.1 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.

6.1.3.2 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).

6.1.3.3 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.

6.2 Data Understanding

An in-depth analysis of the CCTV footage obtained from SP8 Cameras was necessary to understand how the model would work.

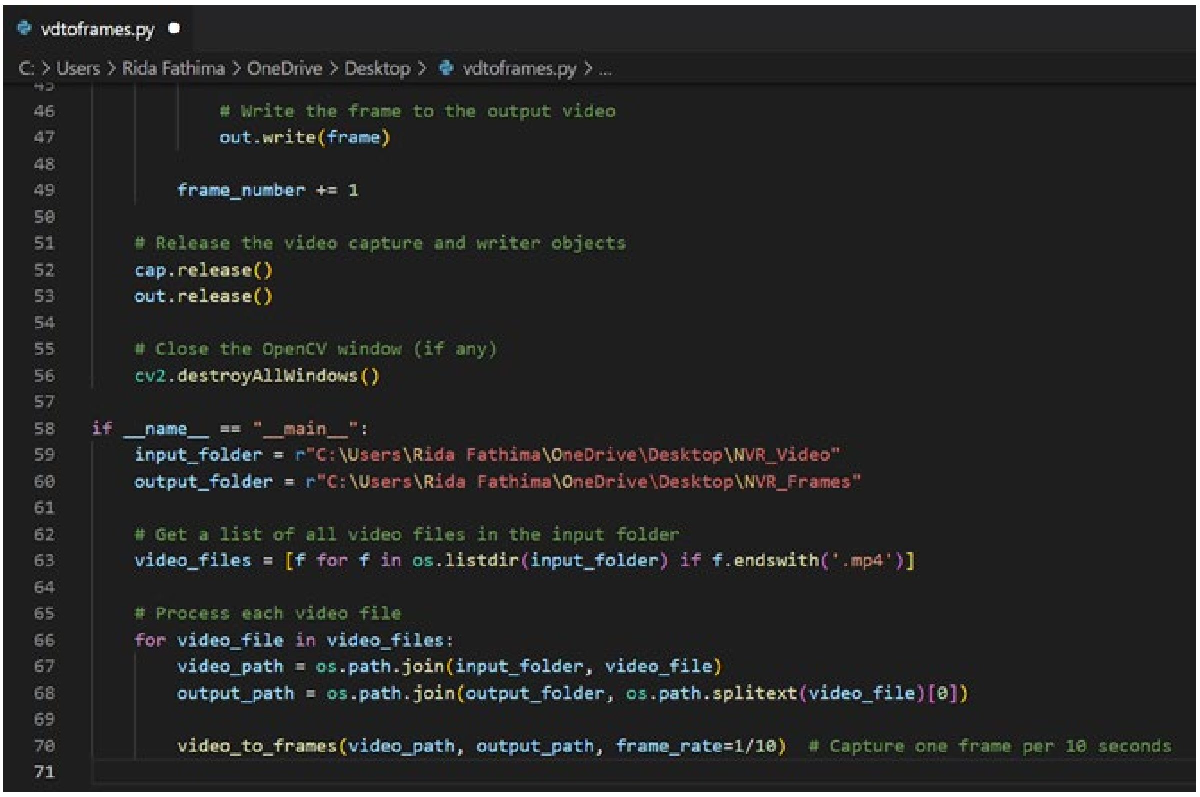
Analyzing the CCTV footage obtained from SP8 Cameras involved a comprehensive and detailed examination to gain a profound understanding of how the model would function in real-world scenarios. The in-depth analysis aimed to extract valuable insights into the intricacies of the surveillance footage, uncovering patterns, anomalies, and relevant features essential for the successful operation of the model. This process typically included scrutinizing various aspects such as the clarity of the footage, lighting conditions, camera angles, and the diversity of scenarios captured. Furthermore, it involved assessing the quality and quantity of data available, identifying potential challenges or limitations, and determining the specific characteristics of objects or events of interest within the footage. By delving deeply into the CCTV footage, researchers or practitioners could refine the model's design, optimize its parameters, and enhance its overall performance, ensuring that it is well-equipped to handle the complexities and variations present in real-world surveillance environments. This meticulous analysis provides a foundation for developing robust and effective models tailored to the unique requirements of the SP8 Camera system.

6.2.1 Data Scenes Comparison

The SP8 dataset consists of various traffic scenes, including highways, urban roads, and rural areas. The data also captures diverse vehicle types, lighting conditions, and Weather conditions.

6.2.2 Data Annotation Process

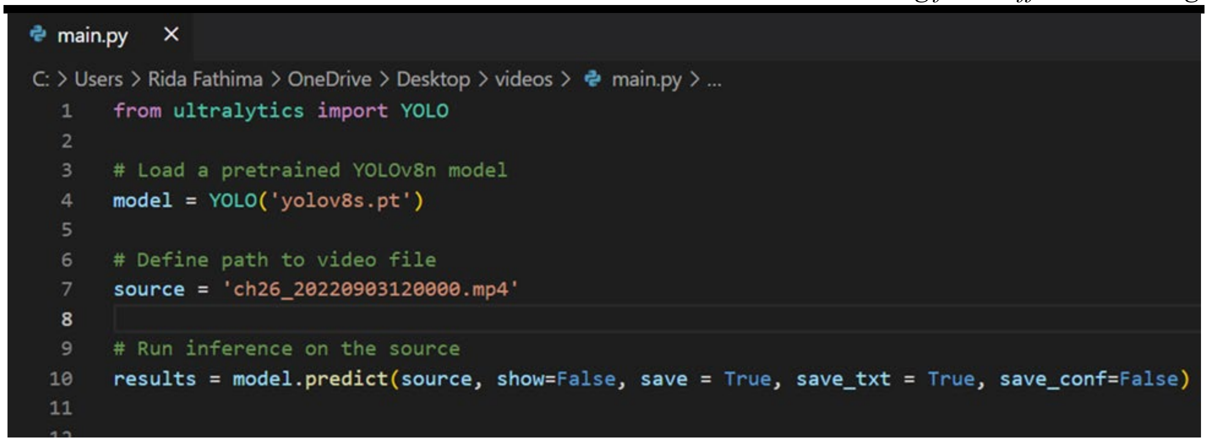
The raw video data was processed using Python code to divide it into frames. One frame per 10 seconds.


vdtoframes.py
C: > Users > Rida Fathima > OneDrive > Desktop > vdtoframes.py > ...

```python
            # Write the frame to the output video
            out.write(frame)

        frame_number += 1

    # Release the video capture and writer objects
    cap.release()
    out.release()

    # Close the OpenCV window (if any)
    cv2.destroyAllWindows()

if __name__ == "__main__":
    input_folder = r"C:\Users\Rida Fathima\OneDrive\Desktop\NVR_Video"
    output_folder = r"C:\Users\Rida Fathima\OneDrive\Desktop\NVR_Frames"

    # Get a list of all video files in the input folder
    video_files = [f for f in os.listdir(input_folder) if f.endswith('.mp4')]

    # Process each video file
    for video_file in video_files:
        video_path = os.path.join(input_folder, video_file)
        output_path = os.path.join(output_folder, os.path.splitext(video_file)[0])

        video_to_frames(video_path, output_path, frame_rate=1/10)  # Capture one frame per 10 seconds
```

Fig 6.1 Python Code for Video to Frames

We had many attempts at labeling the data to check which works best for our model, since data labeling is a crucial step in having a solid vehicle detection model as a foundation. The initial attempt was to try with the auto-labeling process which was done using the yolov8 pre4-trained vehicle detection model.



main.py ×

C: > Users > Rida Fathima > OneDrive > Desktop > videos > main.py > ...

```python
from ultralytics import YOLO

# Load a pretrained YOLOv8n model
model = YOLO('yolov8s.pt')

# Define path to video file
source = 'ch26_20220903120000.mp4'

# Run inference on the source
results = model.predict(source, show=False, save = True, save_txt = True, save_conf=False)

```

Fig 6.2 Code for auto-labeling and predicting CCTV footage using YOLOv8s architecture.

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

Table 6.1 Comparison of different YOLOv8 Architectures

Model	Speed (FPS)	Accuracy (mAP)	Accuracy for SP8 Vehicle Data (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

The second attempt was to combine auto label and manual labeling which was a very time-consuming process. Thus, we had to manually label the data using LabelImg. Manual Labeling process requires marking the vehicles with bounding boxes in each frame per 10 second that was extracted from the CCTV footages.

Labeling is a step that is essential for supervised learning, allowing the model to learn from annotated examples. It is crucial to create a reliable dataset that encompasses various scenarios and conditions similar to real-time traffic scenes. Labeling serves as the foundation for training an accurate model.

6.3 Introduction to YOLOv8

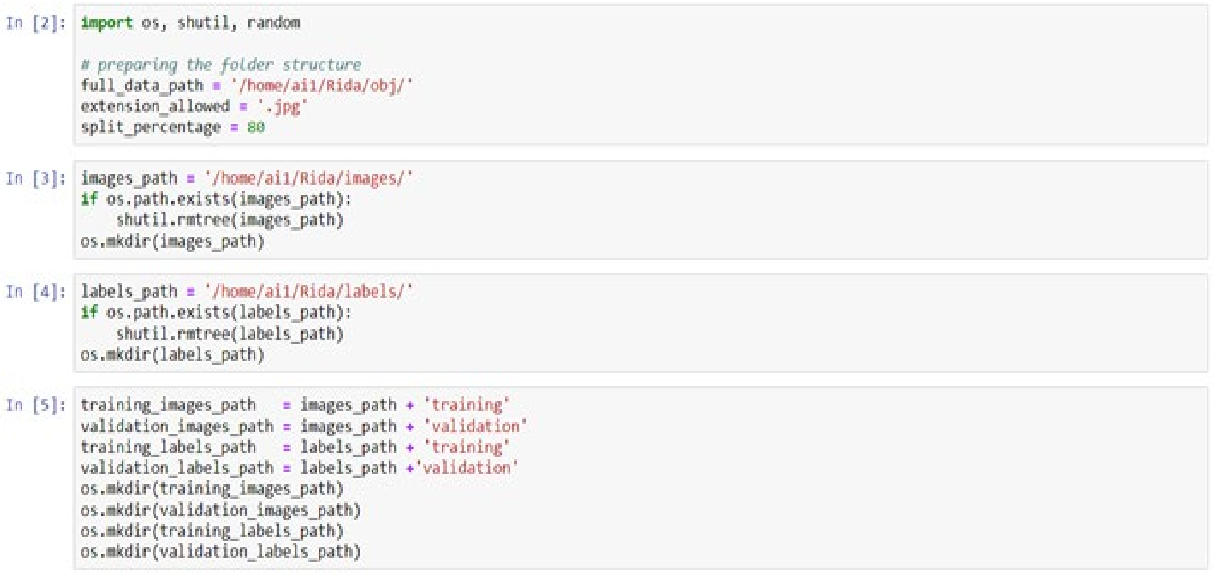
YOLOv8 is an object detection, classification, and segmentation tasks object detection algorithm. It is an improved version of its predecessors, YOLOv5 and YOLOv7, offering several advantages. YOLOv8 achieves superior accuracy compared to previous YOLO versions, consistently outperforming them on various 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 vehicle detection. These pre-trained models are trained on large datasets of labeled images, allowing them to detect objects with high accuracy.

6.3.1 Data for Training and Validation

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% -

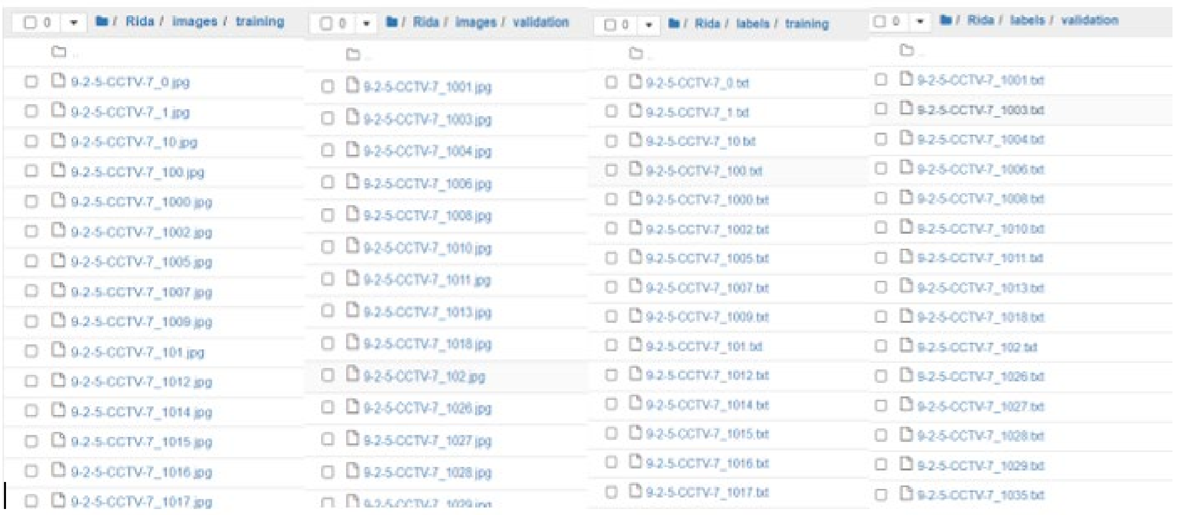
20% splitting strategy to We use the first 80% for training and the rest 20% for validation.



```
In [2]: import os, shutil, random

        # preparing the folder structure
        full_data_path = '/home/ai1/Rida/obj/'
        extension_allowed = '.jpg'
        split_percentage = 80

In [3]: images_path = '/home/ai1/Rida/images/'
        if os.path.exists(images_path):
            shutil.rmtree(images_path)
        os.mkdir(images_path)

In [4]: labels_path = '/home/ai1/Rida/labels/'
        if os.path.exists(labels_path):
            shutil.rmtree(labels_path)
        os.mkdir(labels_path)

In [5]: training_images_path   = images_path + 'training'
        validation_images_path = images_path + 'validation'
        training_labels_path   = labels_path + 'training'
        validation_labels_path = labels_path +'validation'
        os.mkdir(training_images_path)
        os.mkdir(validation_images_path)
        os.mkdir(training_labels_path)
        os.mkdir(validation_labels_path)
```

Fig 6.3 Splitting of Dataset into Training and 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



```
/ Rida / images / training
..
9-2-5-CCTV-7_0.jpg
9-2-5-CCTV-7_1.jpg
9-2-5-CCTV-7_10.jpg
9-2-5-CCTV-7_100.jpg
9-2-5-CCTV-7_1000.jpg
9-2-5-CCTV-7_1002.jpg
9-2-5-CCTV-7_1005.jpg
9-2-5-CCTV-7_1007.jpg
9-2-5-CCTV-7_1009.jpg
9-2-5-CCTV-7_101.jpg
9-2-5-CCTV-7_1012.jpg
9-2-5-CCTV-7_1014.jpg
9-2-5-CCTV-7_1015.jpg
9-2-5-CCTV-7_1016.jpg
9-2-5-CCTV-7_1017.jpg
```

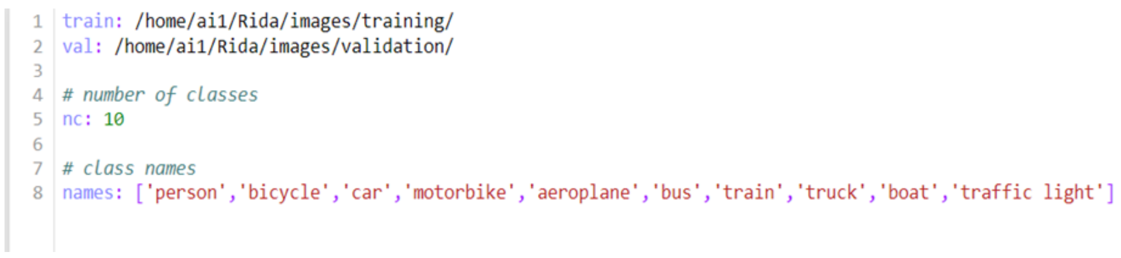
```
/ Rida / images / validation
..
9-2-5-CCTV-7_1001.jpg
9-2-5-CCTV-7_1003.jpg
9-2-5-CCTV-7_1004.jpg
9-2-5-CCTV-7_1006.jpg
9-2-5-CCTV-7_1008.jpg
9-2-5-CCTV-7_1010.jpg
9-2-5-CCTV-7_1011.jpg
9-2-5-CCTV-7_1013.jpg
9-2-5-CCTV-7_1018.jpg
9-2-5-CCTV-7_102.jpg
9-2-5-CCTV-7_1026.jpg
9-2-5-CCTV-7_1027.jpg
9-2-5-CCTV-7_1028.jpg
9-2-5-CCTV-7_1029.jpg
```

```
/ Rida / labels / training
..
9-2-5-CCTV-7_0.txt
9-2-5-CCTV-7_1.txt
9-2-5-CCTV-7_10.txt
9-2-5-CCTV-7_100.txt
9-2-5-CCTV-7_1000.txt
9-2-5-CCTV-7_1002.txt
9-2-5-CCTV-7_1005.txt
9-2-5-CCTV-7_1007.txt
9-2-5-CCTV-7_1009.txt
9-2-5-CCTV-7_101.txt
9-2-5-CCTV-7_1012.txt
9-2-5-CCTV-7_1014.txt
9-2-5-CCTV-7_1015.txt
9-2-5-CCTV-7_1016.txt
9-2-5-CCTV-7_1017.txt
```

```
/ Rida / labels / validation
..
9-2-5-CCTV-7_1001.txt
9-2-5-CCTV-7_1003.txt
9-2-5-CCTV-7_1004.txt
9-2-5-CCTV-7_1006.txt
9-2-5-CCTV-7_1008.txt
9-2-5-CCTV-7_1010.txt
9-2-5-CCTV-7_1011.txt
9-2-5-CCTV-7_1013.txt
9-2-5-CCTV-7_1018.txt
9-2-5-CCTV-7_102.txt
9-2-5-CCTV-7_1026.txt
9-2-5-CCTV-7_1027.txt
9-2-5-CCTV-7_1028.txt
9-2-5-CCTV-7_1029.txt
9-2-5-CCTV-7_1035.txt
```

Fig 6.4 Different Files of Images and Labels for Training and Validation

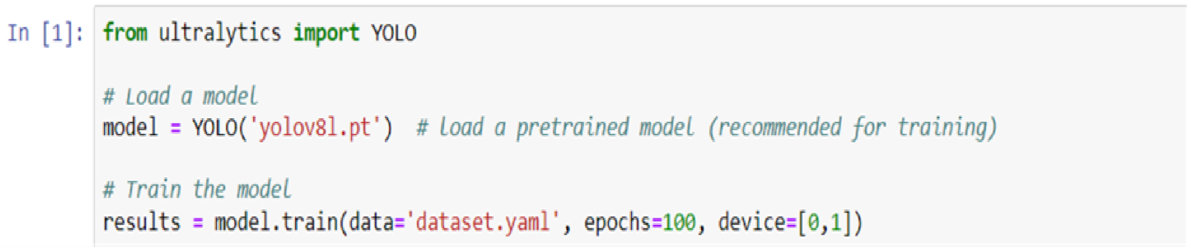
'dataset.yaml' file helps in bringing the training data path, validation data path, No. of Classes, and Class names into a single file which is used for training the YOLOv8 model.



```
train: /home/ai1/Rida/images/training/
val: /home/ai1/Rida/images/validation/

# number of classes
nc: 10

# class names
names: ['person','bicycle','car','motorbike','aeroplane','bus','train','truck','boat','traffic light']
```

Figure 6.5 ‘dataset.yaml’ file code

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



```
In [1]: from ultralytics import YOLO

        # Load a model
        model = YOLO('yolov8l.pt')  # load a pretrained model (recommended for training)

        # Train the model
        results = model.train(data='dataset.yaml', epochs=100, device=[0,1])
```

Fig 6.6 YOLOv8 Code to Load and Train a Model

6.3.2 YOLOv8 Model Evaluation metrics

The below graphs show the evaluation metrics of the YOLOv8 model Training. These graphs contain six lines, each representing a different class of vehicle:

Person
Car
Motorbike
Bus
Truck
All Classes (Combined)

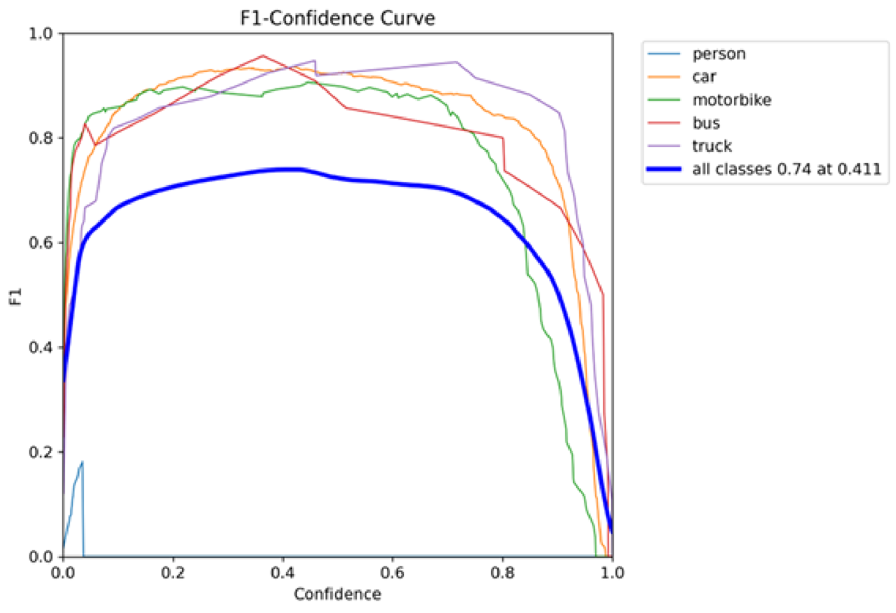


Fig 6.7 F1-Confidence Curve

Cars, motorbikes, buses, and trucks might have high F1 scores at certain thresholds, indicating good performance.

Persons may have a lower F1 score, suggesting challenges in detecting them accurately.

In this case, the model achieves an F1 score of 0.74 when the confidence threshold is set to 0.411.

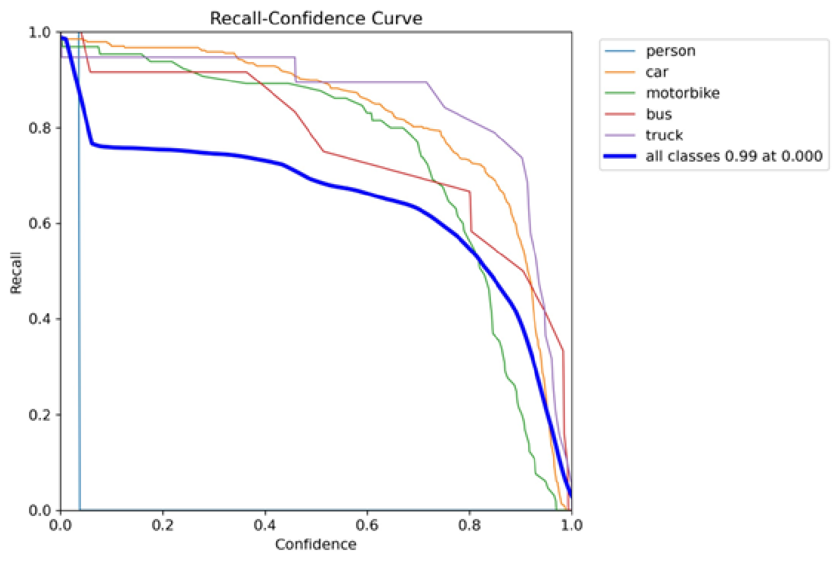


Fig 6.8 Recall-Confidence Curve

The recall decreases more slowly as the confidence increases.

This suggests that the model is more confident in detecting all classes of vehicles than it is in detecting individual classes.

A higher threshold results in fewer detections but potentially higher precision (fewer false positives). A lower threshold increases recall but may introduce more false positives.

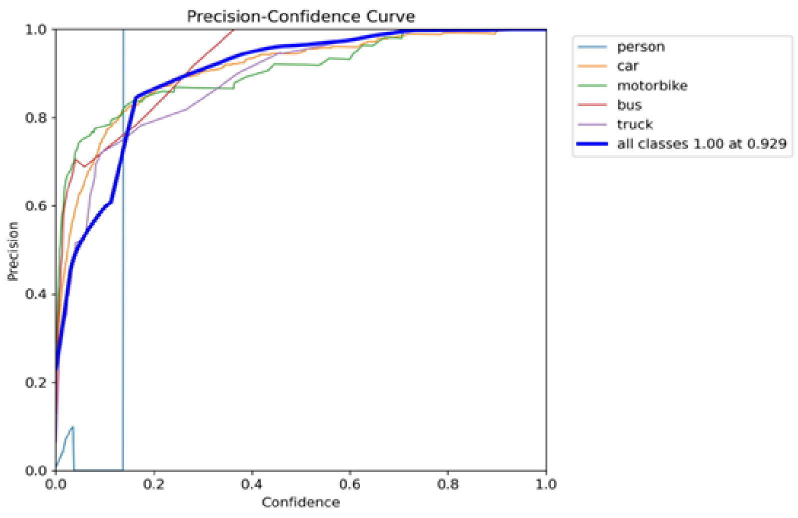


Fig 6.9 Precision-Confidence Curve

The highest precision achieved is at a confidence level of 1.0, with a precision value of 0.929.

This means that when the model is extremely confident (confidence level of 1.0), it correctly detects vehicles with an accuracy of 92.9%.

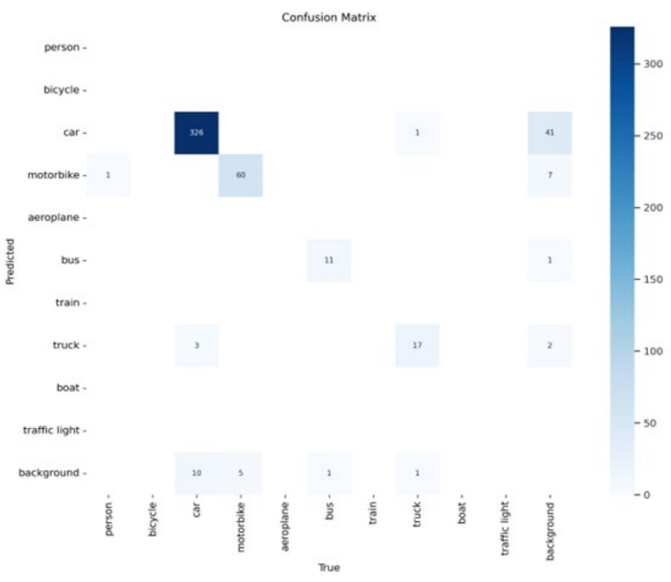


Fig 6.10 Confusion Matrix

The confusion matrix shows how many instances of each class were correctly or incorrectly classified by the model.

It shows the number of true positives, false positives, and false negatives for each class. A true positive is an instance where the model correctly predicted the presence of an object of a certain class. A false positive is an instance where the model predicted the presence of an object of a certain class, but there was no such object in the image. A false negative is an instance where the model failed to predict the presence of an object of a certain class, even though there was such an object in the image.

The diagonal line represents the true positives, while the off-diagonal elements represent the false positives and false negatives. The color scale on the right shows the number of instances for each class.

The color scale on the right shows the number of instances for each class. The darker the color, the more instances of that class were present in the dataset. In our case, we can see that car class is the darkest color which indicates more instances of cars present in our dataset.

6.4 Implementing Flower Framework for YOLOv8 Vehicle Detection Model

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.

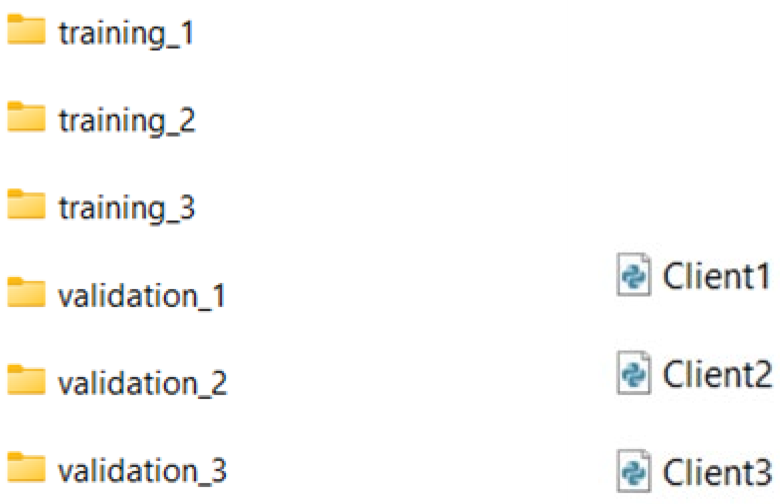
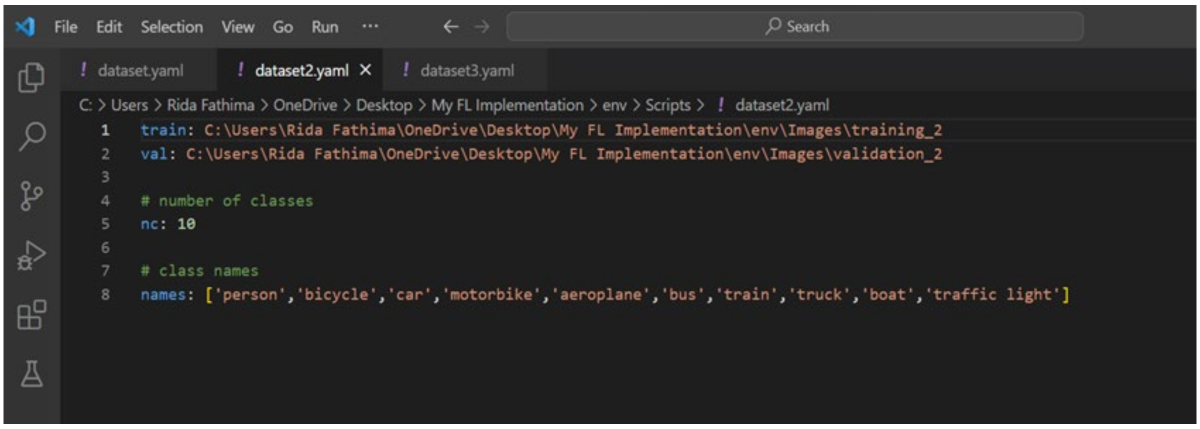


Fig 6.11 Dividing Dataset for 3 different Clients

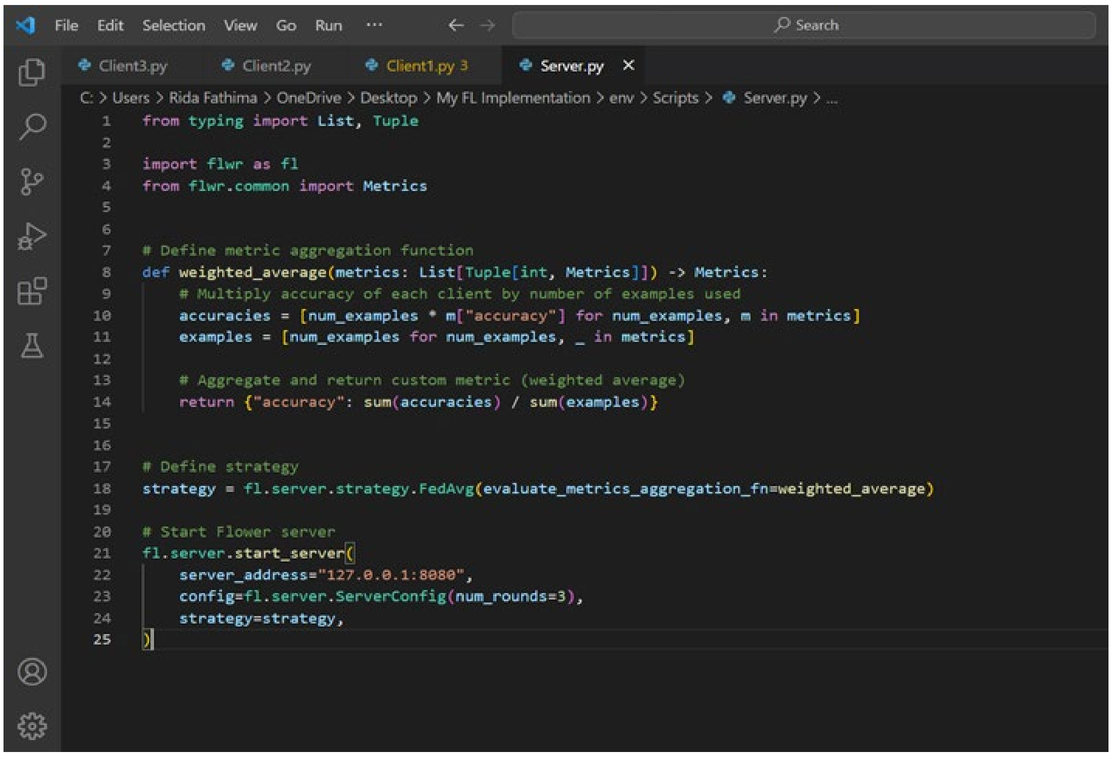
For federated learning, three ‘dataset.yaml’ files are created as shown below.


File Edit Selection View Go Run ···
Search
dataset.yaml
dataset2.yaml
dataset3.yaml
C: > Users > Rida Fathima > OneDrive > Desktop > My FL Implementation > env > Scripts > dataset2.yaml

```yaml
train: C:\Users\Rida Fathima\OneDrive\Desktop\My FL Implementation\env\Images\training_2
val: C:\Users\Rida Fathima\OneDrive\Desktop\My FL Implementation\env\Images\validation_2

# number of classes
nc: 10

# class names
names: ['person','bicycle','car','motorbike','aeroplane','bus','train','truck','boat','traffic light']
```

Fig 6.12 ‘dataset.yaml’ file for Federated Learning

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 6.13 Running Federated learning

CHAPTER-7
TIMELINE FOR EXECUTION OF PROJECT

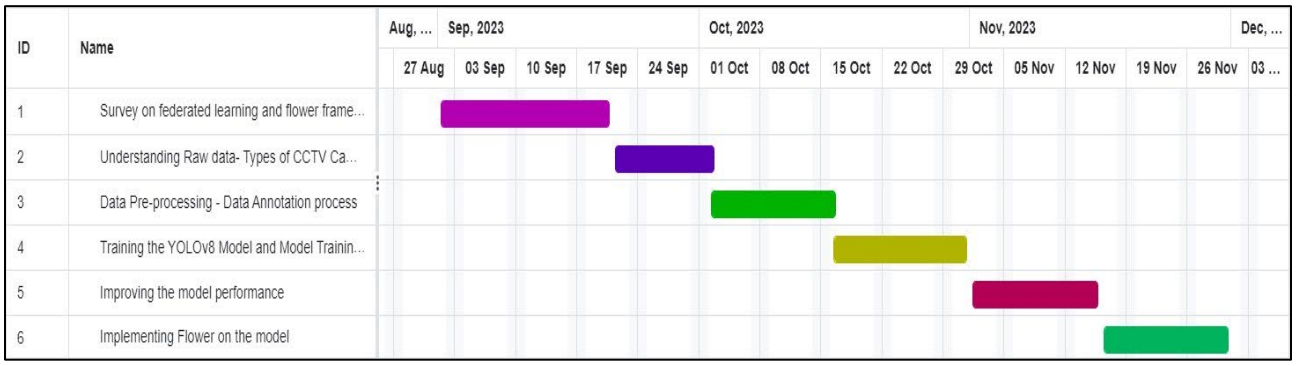


Fig 7.1 Timeline of Execution of Project (Gantt Chart)

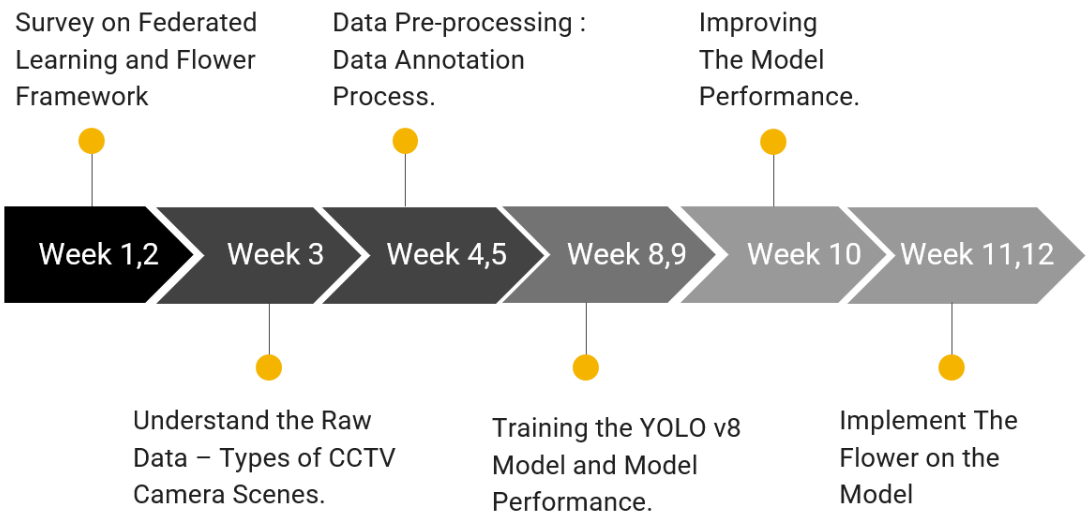

Survey on Federated Learning and Flower Framework
Data Pre-processing : Data Annotation Process.
Improving The Model Performance.
Week 1,2
Week 3
Week 4,5
Week 8,9
Week 10
Week 11,12
Understand the Raw Data – Types of CCTV Camera Scenes.
Training the YOLO v8 Model and Model Performance.
Implement The Flower on the Model

Fig 7.2 Project Timeline Depiction

CHAPTER-8
OUTCOMES

The successful implementation of federated learning for traffic monitoring using vehicle detection and counting through CCTV video footage collected by SP8 has yielded the following outcomes. First and foremost, the project aims to deliver a robust vehicle detection model leveraging YOLOv8, capable of accurately identifying and counting various vehicle types, including cars, trucks, motorcycles, among others, within the Chiayi government's CCTV data. By integrating federated learning into the vehicle detection system using the Flower framework, the project aims to establish a decentralized and collaborative training approach, enhancing privacy in sensitive applications like vehicle type detection. This decentralized model will allow for the collaborative improvement of the detection model without centralizing the sensitive CCTV data, addressing privacy concerns and challenges associated with data sharing. The outcomes of this project are anticipated to contribute to advancements in federated learning methodologies for real-world applications, particularly in the context of traffic monitoring, where privacy and data security are paramount considerations.

The Control and Command Center was where the Federated Learning system is to be deployed. Our Federated Learning system deploys the Federated Learning server on the platform using containerization. Communication between clients and servers is facilitated through the gRPC communication protocol.

Clients only upload their weights to the server, without uploading all the data, ensuring data privacy.

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.

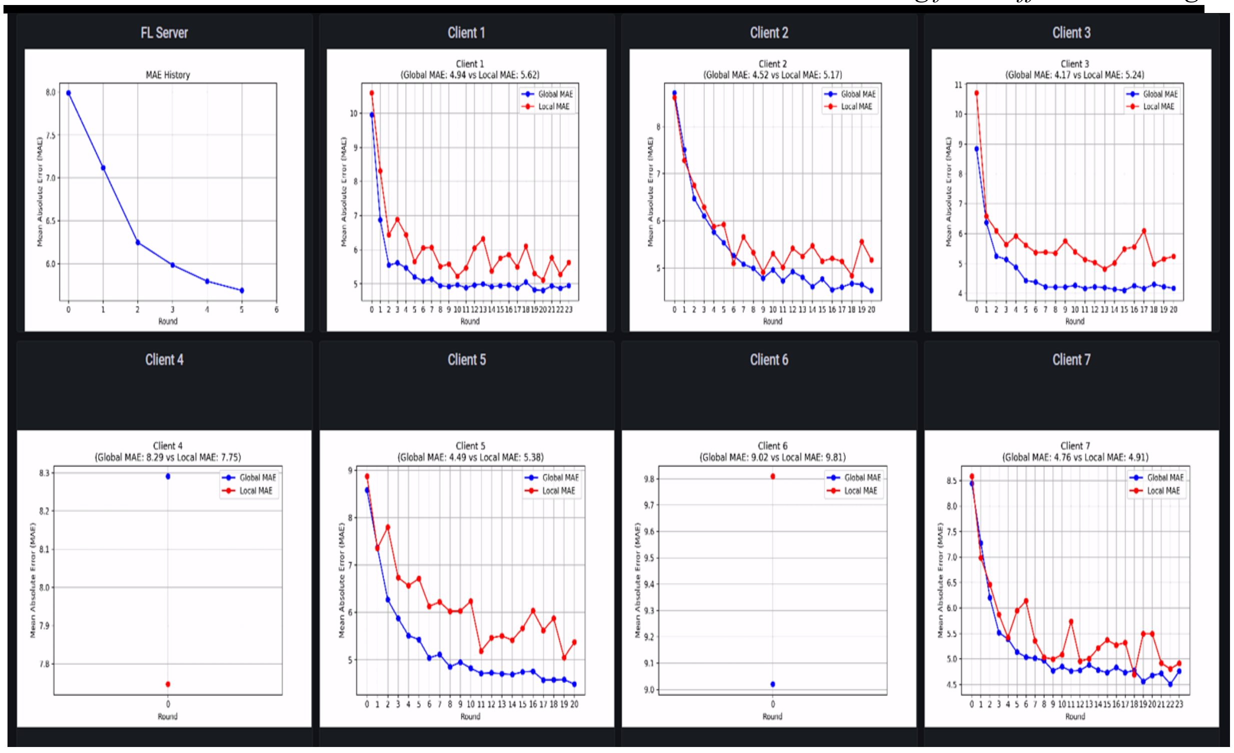


Fig 8.1 MAE Graph

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 vehicle 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 on unseen CCTV data.
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 8.1 Vehicle Detection Accuracy Comparison

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

Table 8.2 Vehicle Count Accuracy Comparison

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

Benefits of FL for Vehicle Detection and Count:

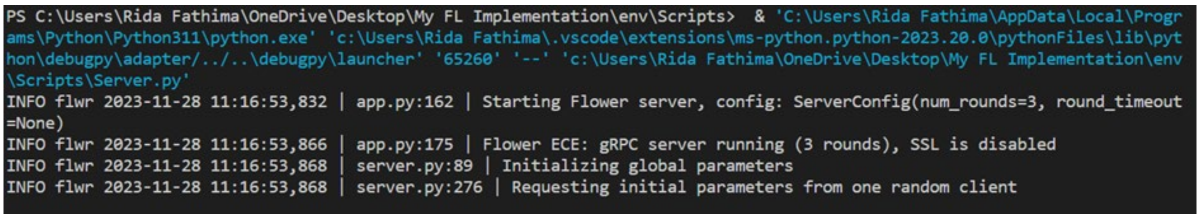
- Improved privacy: Data remains on client devices, reducing privacy concerns.
- Enhanced scalability: Can handle large datasets from geographically distributed cameras.
- Reduced communication overhead: Only model updates are shared, minimizing network traffic.
- Continuous model updates: Enables real-time adaptation to changing traffic patterns and environmental conditions.

Challenges of FL for Vehicle Detection and Count:

- 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.

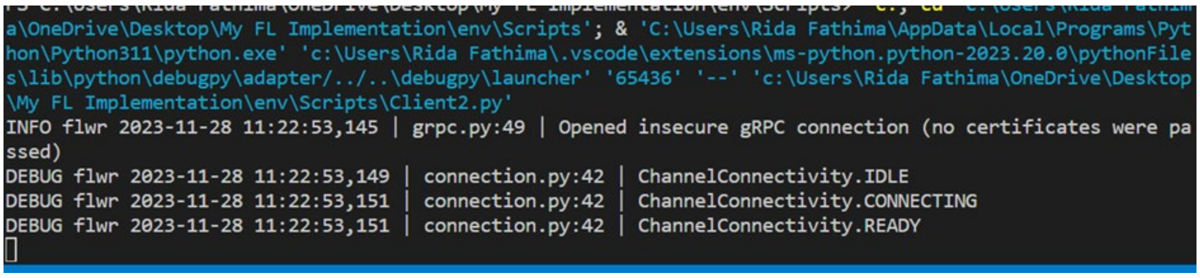
Overall, federated learning offers a promising approach for vehicle detection and count, especially when dealing with privacy-sensitive data and large-scale deployments. While challenges remain, the potential benefits of improved privacy, scalability, and continuous adaptation make FL a valuable tool for advancing smart traffic management and intelligent transportation systems.

CHAPTER-9
RESULTS AND DISCUSSIONS



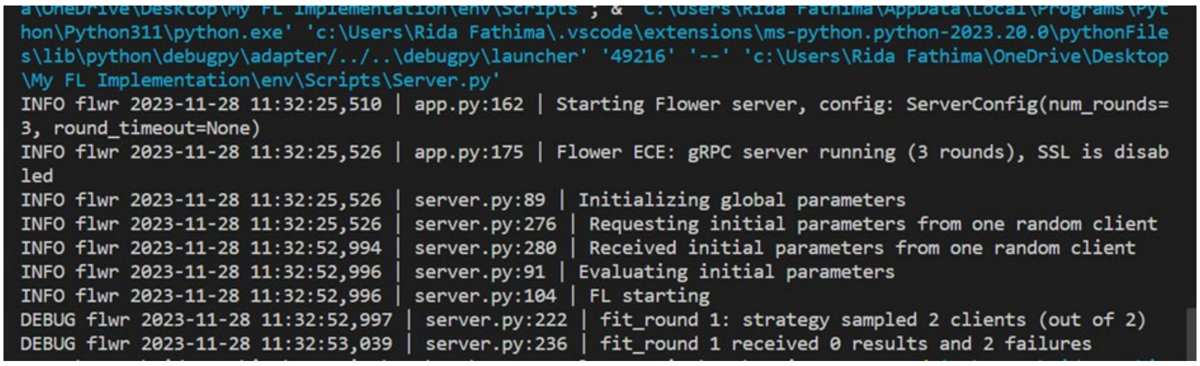
```
PS C:\Users\Rida Fathima\OneDrive\Desktop\My FL Implementation\env\Scripts>  & 'C:\Users\Rida Fathima\AppData\Local\Progr
ams\Python\Python311\python.exe' 'c:\Users\Rida Fathima\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\pyt
hon\debugpy\adapter/../..\debugpy\launcher' '65260' '--' '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 9.1 Shows Flower Server Starting, Initializing Global Parameters, and Requesting Initial Parameters from any one random Client.



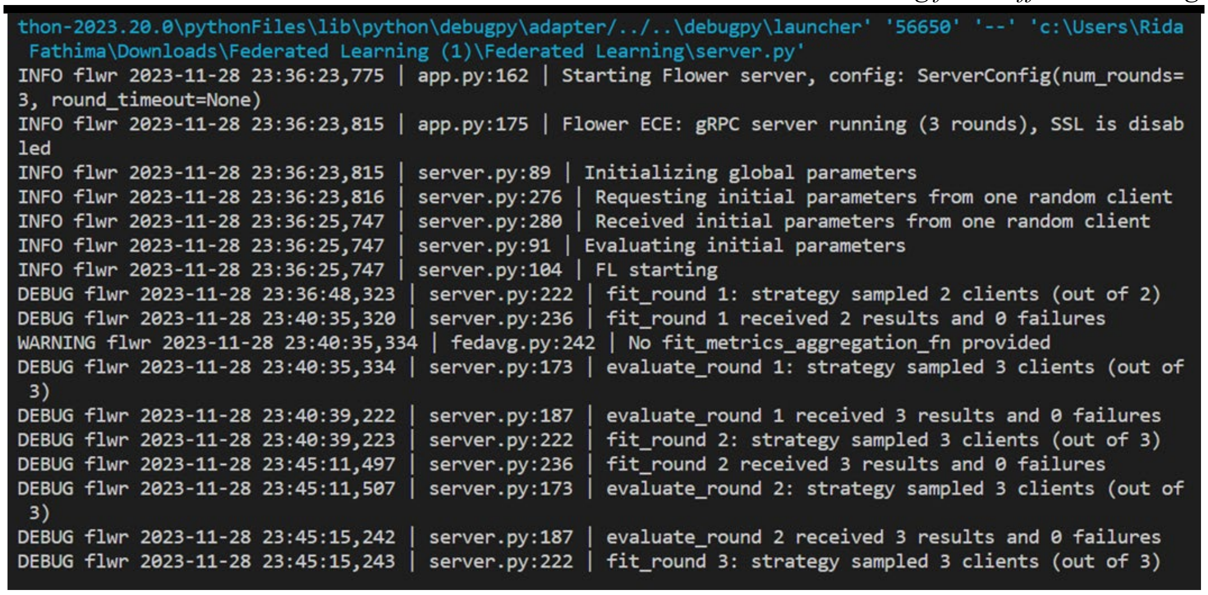
```
a\OneDrive\Desktop\My FL Implementation\env\Scripts'; & 'C:\Users\Rida Fathima\AppData\Local\Programs\Pyt
hon\Python311\python.exe' 'c:\Users\Rida Fathima\.vscode\extensions\ms-python.python-2023.20.0\pythonFile
s\lib\python\debugpy\adapter/../..\debugpy\launcher' '65436' '--' 'c:\Users\Rida Fathima\OneDrive\Desktop
\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 9.2 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 9.3 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)
```

Fig 9.4 Results of Flower

CHAPTER-10
CONCLUSION AND FUTURE PROSPECTS

The main challenges faced in this project are implementation of the YOLOv8 model using Flower. Other Challenges faced were lighting, & weather conditions of dataset, and accuracy in vehicle detection. The night traffic scenes reflecting lights makes it harder to detect the type of vehicle. During rainy weather, the camera is covered with water droplets making it difficult to detect the type of vehicle. To address these challenges, I incorporated a diverse range of scenarios, and including different types of scenarios in each client would help the model understand all types of scenarios.

The successful implementation of federated learning for traffic monitoring will help us enhance the accuracy of vehicle detection across multiple CCTV feeds, which contributes to improved traffic management and safety. Since different vehicles have different environmental impacts, Monitoring Vehicle Types also allows us to assess the Emissions and helps cities reduce air pollution, promoting sustainable transportation.

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 addition to the challenges mentioned, the integration of the YOLOv8 model into the federated learning framework using Flower posed unique implementation hurdles. Coordinating communication and synchronization between multiple clients and the central server while ensuring seamless integration with YOLOv8's real-time object detection capabilities required meticulous technical considerations. Moreover, handling the disparities in lighting and weather conditions within the dataset presented inherent difficulties. Night traffic scenes, characterized by reflective lights, posed a particular challenge for vehicle detection accuracy, while rainy weather conditions added another layer of complexity by obstructing camera lenses with water droplets. These challenges underscore the need for adaptive and resilient federated learning models capable of accommodating diverse environmental conditions for robust real-world deployment.

The successful implementation of federated learning in traffic monitoring holds significant promise in revolutionizing urban management and sustainability. By enhancing the accuracy of vehicle detection across multiple CCTV feeds, traffic management becomes more efficient, leading to safer roadways and reduced congestion. Beyond immediate safety concerns, the ability to monitor different vehicle types contributes to a broader environmental impact assessment. The classification of vehicles allows for the evaluation of emissions, enabling cities to make informed decisions to reduce air pollution and promote sustainable transportation solutions. While federated learning offers substantial benefits in terms of decentralized model training and privacy preservation, ongoing research and development efforts are essential to overcoming existing challenges. As federated learning becomes more pervasive, addressing issues like communication overhead, model convergence, and non-identically distributed data becomes paramount to ensuring its seamless integration and widespread applicability in diverse domains.

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.

APPENDIX-A
PSUEDOCODE

Algorithm 1 Server Pseudocode

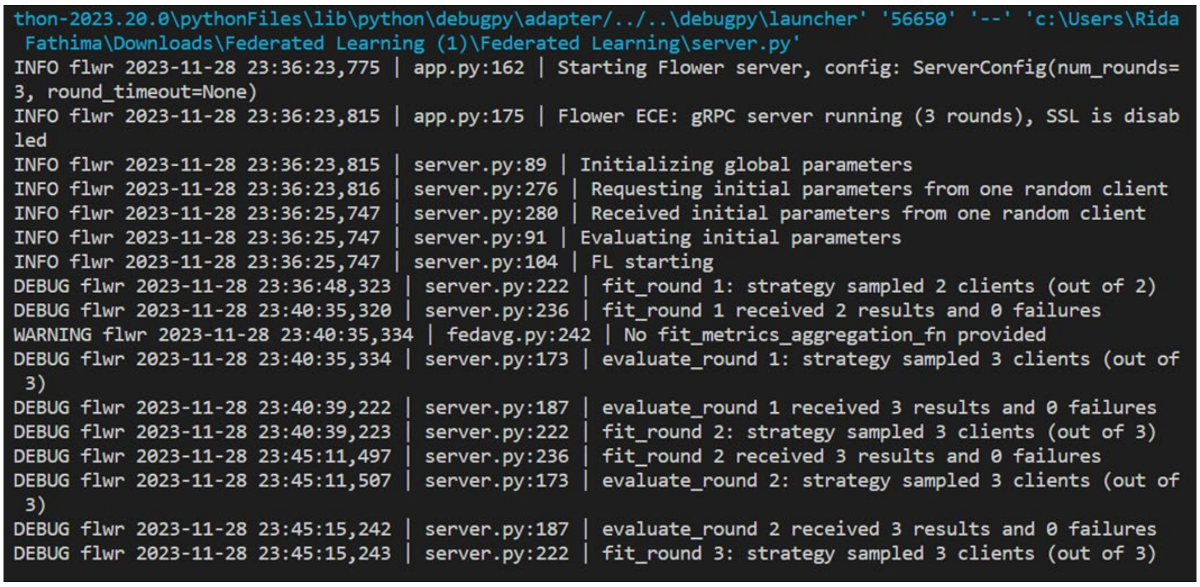
```
 1: procedure SERVERINITIALIZATION
 2:     Import necessary libraries                                ▷ flwr, torch, etc.
 3:     Define metric aggregation function                ▷ e.g., weighted_average
 4:     Define federated learning strategy                        ▷ e.g., FedAvg
 5:     Start Flower server with configured parameters and strategy           ▷
    fl.server.start_server
 6: end procedure
 7: procedure MAINLOOP
 8:     for i in range(number of rounds) do
 9:         Wait for connections from clients
10:         Receive models from connected clients
11:         Aggregate models using federated learning strategy
12:         Send aggregated model back to clients
13:     end for
14: end procedure
15: procedure SERVERCLEANUP
16:     Close server connection                          ▷ fl.server.stop_server
17: end procedure
```

Algorithm 2 Client Pseudocode

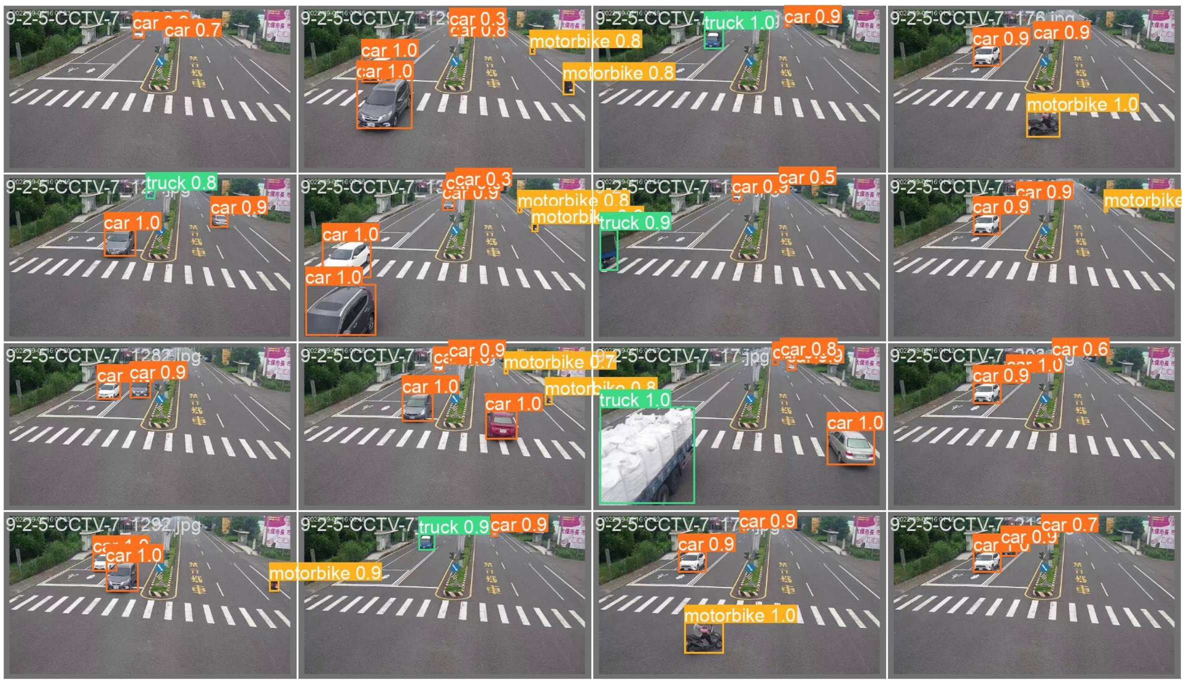
```
 1: procedure CLIENTINITIALIZATION
 2:     Import necessary libraries                 ▷ torch, torchvision, etc.
 3:     Define neural network model                         ▷ e.g., Net class
 4:     Load and preprocess dataset                          ▷ e.g., CIFAR-10
 5: end procedure
 6: procedure FLOWERCLIENTINITIALIZATION
 7:     Extend NumPyClient class                      ▷ fl.client.NumPyClient
 8:     Implement get_parameters, set_parameters, fit, and evaluate methods
 9: end procedure
10: procedure MAINLOOP
11:     Connect to Flower server                ▷ fl.client.start_numpy_client
12:     while server is accepting connections do
13:         Send local model parameters to the server
14:         Receive aggregated model from the server
15:         Perform local training (fit method)
16:         Evaluate local model (evaluate method)
17:         Send local evaluation results to the server
18:     end while
19: end procedure
20: procedure CLIENTCLEANUP
21:     Close client connection                         ▷ fl.client.stop_client
22: end procedure
```

APPENDIX-B

SCREENSHOTS

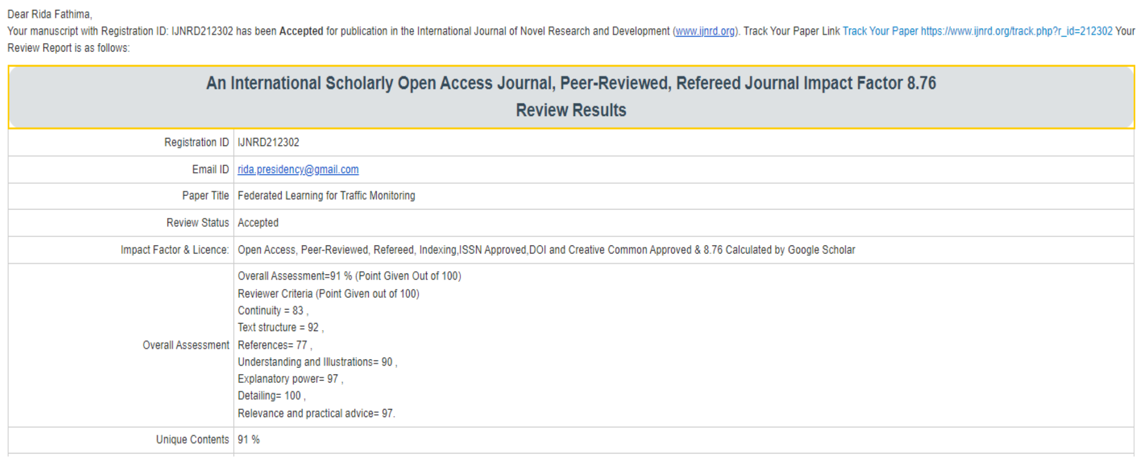


```
thon-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56650' '--' 'c:\Users\Rida
 Fathima\Downloads\Federated Learning (1)\Federated Learning\server.py'
INFO flwr 2023-11-28 23:36:23,775 | app.py:162 | Starting Flower server, config: ServerConfig(num_rounds=
3, round_timeout=None)
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)
```



APPENDIX-C

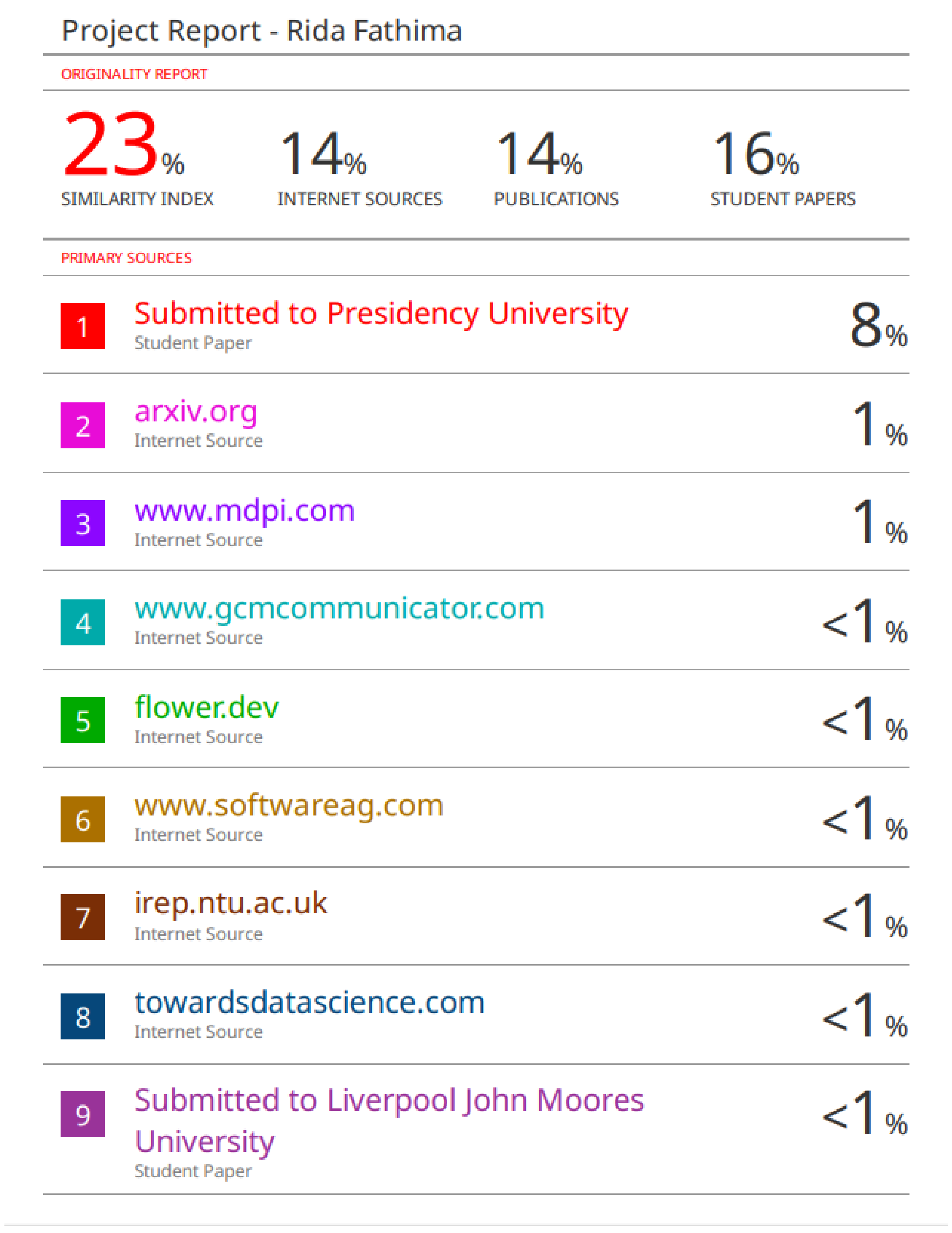
ENCLOSURES



Dear Rida Fathima,
Your manuscript with Registration ID: IJNRD212302 has been **Accepted** for publication in the International Journal of Novel Research and Development (www.ijnrd.org). Track Your Paper Link Track Your Paper https://www.ijnrd.org/track.php?r_id=212302 Your Review Report is as follows:

An International Scholarly Open Access Journal, Peer-Reviewed, Refereed Journal Impact Factor 8.76
Review Results

Registration ID	IJNRD212302
Email ID	rida.presidency@gmail.com
Paper Title	Federated Learning for Traffic Monitoring
Review Status	Accepted
Impact Factor & Licence:	Open Access, Peer-Reviewed, Refereed, Indexing,ISSN Approved,DOI and Creative Common Approved & 8.76 Calculated by Google Scholar
Overall Assessment	Overall Assessment=91 % (Point Given Out of 100) Reviewer Criteria (Point Given out of 100) Continuity = 83 , Text structure = 92 , References= 77 , Understanding and Illustrations= 90 , Explanatory power= 97 , Detailing= 100 , Relevance and practical advice= 97.
Unique Contents	91 %



Project Report - Rida Fathima

ORIGINALITY REPORT

23%	14%	14%	16%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Presidency University Student Paper	8%
2	arxiv.org Internet Source	1%
3	www.mdpi.com Internet Source	1%
4	www.gcmcommunicator.com Internet Source	<1%
5	flower.dev Internet Source	<1%
6	www.softwareag.com Internet Source	<1%
7	irep.ntu.ac.uk Internet Source	<1%
8	towardsdatascience.com Internet Source	<1%
9	Submitted to Liverpool John Moores University Student Paper	<1%

SUSTAINABLE DEVELOPMENT GOALS

SUSTAINABLE DEVELOPMENT GOALS
17 GOALS TO TRANSFORM OUR WORLD
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

Federated learning in traffic monitoring aligns with several Sustainable Development Goals (SDGs) by contributing to the development of smart and sustainable cities, efficient transportation systems, and environmental conservation. Here are some SDGs that are particularly relevant:

1. Goal 9: Industry, Innovation, and Infrastructure
  - Target 9.1: Develop quality, reliable, sustainable, and resilient infrastructure, including regional and transborder infrastructure, to support economic development and human well-being.

Federated learning in traffic monitoring contributes to the development of innovative technologies and infrastructure for more efficient traffic management.

2. Goal 11: Sustainable Cities and Communities
  - Target 11.2: Provide access to safe, affordable, accessible, and sustainable transport

systems for all.

- Target 11.6: Reduce the adverse per capita environmental impact of cities, including by paying special attention to air quality and municipal and other waste management.

Federated learning helps in optimizing traffic flow, reducing congestion, and improving air quality by enabling more intelligent traffic monitoring and management.

3. Goal 13: Climate Action

- Target 13.2: Integrate climate change measures into national policies, strategies, and planning.

By optimizing traffic flow and reducing congestion, federated learning can contribute to lower emissions and help address climate change concerns related to transportation.

4. Goal 17: Partnerships for the Goals

- Target 17.6: Enhance North-South, South-South, and triangular regional and international cooperation on and access to science, technology, and innovation, and enhance knowledge-sharing on mutually agreed terms.

Federated learning often involves collaboration and knowledge-sharing among different entities, fostering partnerships for the development and deployment of advanced traffic monitoring systems.