ALMA MATER STUDIORUM – UNIVERSITY OF BOLOGNA

SCHOOL OF ENGINEERING

Department of
Electrical, Electronic and Information Engineering
"Guglielmo Marconi"
DEI

MASTER'S DEGREE IN TELECOMMUNICATION ENGINEERING

MASTER'S THESIS
in
Multimedia Services and Applications

Network Federated Learning for Real Time Traffic Predictions in Internet of Vehicles Scenarios

Candidate

Abdullah Abbasi

Supervisor

Prof. Daniele Tarchi

Co-Supervisor

Mr. Swapnil Sadashiv Shinde

Academic Year
2021/2022

Abstract

The project aims to present a theoretical and algorithmic framework for Network Federated Learning (NFL) in decentralized collections of local datasets, where the datasets are structurally related through some specific similarity measure. The similarity can be based on functional relationships, statistical dependencies, or spatiotemporal proximity. In our case, we considered the mobility model of vehicles to model the movement of vehicles within an area to better analyse the dynamics of traffic within the city. In order to formulate NFL, our methodology uses a Generalized Total Variation (GTV) minimization which is the extension of existing federated multi-task learning methods. The approach we presented is adaptable and works with different models.

Moreover, for local models that result in convex problems, we provide precise conditions on the local models and their network structure such that the algorithm learns nearly optimal local models. The analysis reveals an interesting interplay between the convex geometry of local models and the (cluster-) geometry of their network structure. Means, the shape of the local models which is determined by their convex geometry, and the structure of the network determined by their cluster geometry, can influence the quality of the learned local models.

We then compare the results of our algorithm with centralized federated learning (FL) model. The analysis shows our algorithm for NFL has quite better performance with respect to centralized FL.

Index

Introduction

One of the relatively few business areas that has had rapid growth for more than 30 years is the mobile communication industry. With three identified services/use cases, including eMBB (enhanced mobile broadband), URLLC (ultra-reliable and low-latency communications), and mMTC (massive machine-type communications) [1], the 5G mobile networks promise to further transform our contemporary society and vertical industries. As the commercial rollout of 5G mobile systems continues, research conversations on the 6G generation of mobile systems have begun [2], [3]. Extending beyond 5G capacity, 6G mobile network technologies are motivated by the demanding requirements of new mobile applications like extended reality, industry 5.0, and digital twins. Although 6G is still in the conceptual stage, some of the top vendors have released the first iterations of technology-driven key performance indicators (KPIs) for 6G. The potential enabling technologies for 6G are generally agreed upon, including THz communications, integrated spatial-terrestrial networks (ISTN), reconfigurable intelligent surfaces (RIS), and artificial intelligence (AI) [2], [4]. Existing technologies like enhanced channel coding and modulation, very large-scale antenna, spectrum sharing, full duplex, etc. will evolve because of these revolutionary technologies.

The global spread of 5G networks is anticipated to encourage the adoption of Internet of Things (IoT) technology and applications, which demand extremely high levels of connectivity, security, and low latency. However, it is well known that many IoT devices require more than just 5G to exchange different kinds of data in real time. These limitations encourage the development of 6G technologies, which can enable lower latency, better network capacity, and quicker data transmission rates. The sixth generation of mobile wireless networks (6G) is already conceptualized, and a few potentially disruptive technologies are already acting as catalysts for the birth of a variety of edge-cutting applications. As vehicular networks become more dynamic and complex with strict requirements on ultra-low latency, high reliability, and massive connections, 6G will be a key supporter for the development towards a truly Intelligent Transportation System and the realization of the Smart City concept by overcoming the limitations of 5G. More importantly, giving these vital systems security and privacy should be a key priority because weaknesses can have disastrous effects.

Figure 1. shows the architecture of 6G and Internet of Vehicle (IoV) system together [6]. 6G is responsible for providing high reliability, low latency, scalability, and high throughput. With all these KPI’s we can perform communications, computations, positioning and sensing functionalities for vehicular networks.

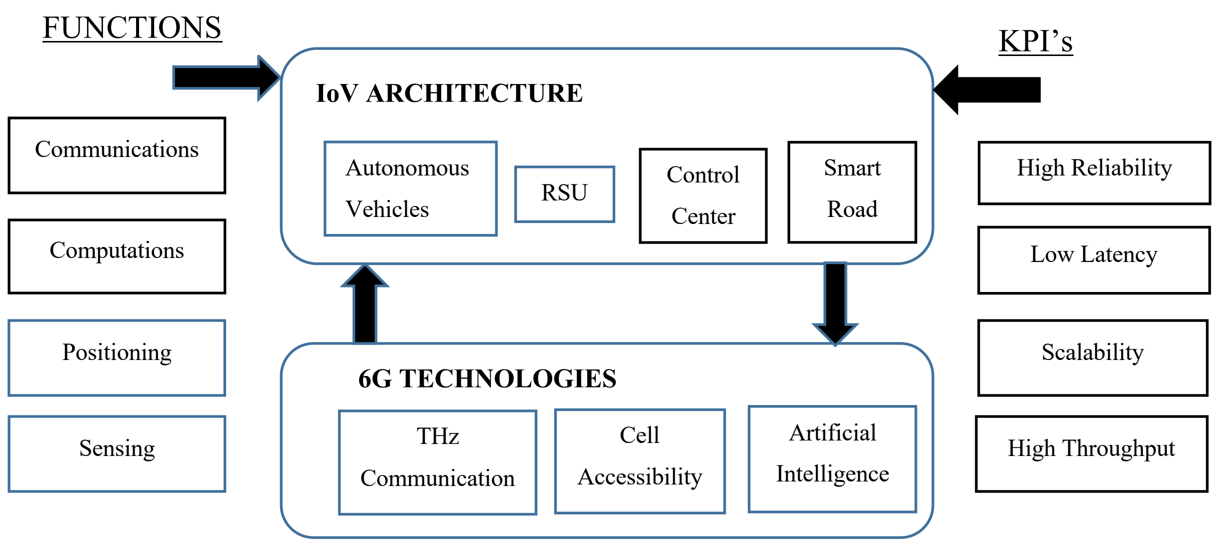


Figure 1. 6G & IoV Architecture [6]

Internet of Vehicles (IoVs) has been regarded as a crucial technology for establishing Intelligent Transportation Systems in smart cities and is one of the most promising applications in the future Internet of Things. The introduction of sixth generation (6G) communications technologies will result in large network infrastructures being densely distributed and an exponential rise in the number of network nodes. We have seen a booming industry and increased use of AI solutions in a variety of ICT applications in recent years. Intelligent personal assistants, video/audio surveillance, smart city operations, and autonomous cars are just a few of the ways that AI services are growing in popularity. In fact, whole sectors are changing. Industry 4.0, which aspires to digitize manufacturing, robotics, automation, and related industrial fields as part of digital transformation, is a prime example. In addition, the growing usage of computers and software necessitates new kinds of design trade-offs, such as those involving the energy requirements for computations, the timing of data transmissions, privacy, and security [6].

Two technologies that are anticipated to influence communication and transportation in the future are 6G and IoVs. In addition to enabling new and inventive apps and services, 6G's ultra-fast speed, low latency, and large bandwidth along with IoVs improved capabilities for vehicles will also increase safety and efficiency.

IoVs can be represented by a network structure, which can be visualized as a graph consisting of nodes and edges. Nodes in an IoV network can represent various components such as vehicles, roadside sensors, cloud servers, and mobile devices. These nodes are interconnected by edges that represent the communication links between them. The edges can be directed or undirected, depending on the type of communication between the nodes. Representing the IoV as a network structure provides a useful way to understand the complex interactions and communication patterns that occur in this emerging field.

In application domains that that generate local datasets through a network structure, there are many uses of federated learning (FL) [46]. Two examples of such domains are pandemic management and the Internet of Things (IoT) [47], where local datasets are generated by smartphones, wearables, or industrial IoT devices. FL refers to a collection of machine learning techniques that collaboratively train models on decentralized collections of local datasets [39]. Instead of collecting all data at a central location, FL methods carry out computations such as gradient descent steps during model training at the location of data generation. FL is appealing for applications involving sensitive data, such as healthcare, as it does not require the exchange of raw data but only model updates without leaking sensitive information in local datasets. Furthermore, FL methods can offer robustness against malicious data perturbation due to intrinsic averaging or aggregation over large collections of mostly benign datasets [48].

The challenges faced by FL applications that deal with local datasets is that many datasets have different statistical properties such as in case of IoVs the movement of vehicles from one area to the other are different at different time instants (see Chapter 3). Each local dataset creates a unique learning task that requires optimizing the parameters of a local model. We propose an optimization method that trains personalized local models related to the statistical properties of the corresponding local dataset. This method uses a regularized empirical risk minimization approach that incorporates a measure of local model parameter variation as a regularizer term, (see Chapter 5). The resulting optimization problem is solved by iteratively using Gradient Descent steps which implements message passing over the network structure of local datasets (see Chapter 5).

To address the heterogeneity among local datasets, we introduce Clustered FL, which uses clustering assumptions to group local datasets with similar statistical properties into disjoint subsets or clusters. The clustering assumption requires local datasets and their associated learning tasks to belong to statistically homogeneous clusters (see Chapter 4). The main contribution of this thesis is to provide a detailed characterization of the cluster structure and local model geometry for local datasets that allow their methods to pool local datasets that form statistically homogeneous clusters of datasets. The method proposed for NFL, does not simply combine clustering methods and model training, but instead interweaves the pooling of local datasets with model training using the connectivity of the empirical graph.

In Chapter 1, we will talk about the importance of AI in 6g for creating a native AI-based network. We will see the role of 6G and its importance for enabling Internet of Vehicles (IoVs). Then Chapter 2 will focus on the importance of using Federated Learning (FL) and Network Federated Learning (NFL) and how NFL outperforms FL. In Chapter 3 we will see how we use network graphical model and the mobility model of vehicles in order to model the movement of vehicles within the city. We will talk about how we build Manhattan Grid Mobility Model (MGMM) in our case for this project. Then Chapter 4 will focus on how we represented our data through an undirected empirical graph for our approach in IoVs scenario and we will discuss the modelling of the whole network for our project and we also talk about the clustering assumptions for Network Federated Learning. Chapter 5 will focus on the algorithmic framework for our project. Here we will discuss that how we performed Generalized Total Variation (GTV) Minimization to measure the variation of local model parameters. We then presented an algorithm for GTV minimization in our project. In Chapter 6 we will compare and talk about the results of our algorithm with the results of traditional centralised federated learning.

1. 6G & Internet of Vehicles (IoVs)

The fifth generation (5G) mobile networks brought in tremendous improvements over the previous generations of mobile communication technology. However, the development of the next generation of mobile communication technology, 6G, is already in progress, and it promises to be even more transformative.

One of the most significant features of 6G is expected to be its native support for Artificial Intelligence (AI). AI is already transforming industries ranging from healthcare to transportation, and it is poised to have an even more significant impact on the world in the coming years. By integrating AI into the 6G network, we can unlock even more potential for this transformative technology [7].

The inclusion of AI in 6G is essential to realizing the full potential of this transformative technology. By integrating AI into the very fabric of the network, we can enhance network management, improve the user experience, enhance security, enable intelligent connectivity, and unlock unprecedented data capabilities. With these benefits, the potential for innovation and progress is limitless.

1.1 The Role of 6G in Enabling the Internet of Vehicles (IoVs)

With the varying demands in service qualities, Internet of vehicles (IoVs) are one of the essential vertical businesses in 6G. For IoV, there are two levels of definition [5]:

1. In essence, Autonomous Vehicles (AV) that are connected to other cars and/or infrastructure are referred to as IoVs.
2. With little to no human control, AVs are capable of perceiving their driving surroundings and moving safely.

Autonomous and linked vehicles, which are seen as the two most promising technologies for future transportation systems, were first developed concurrently. They both, however, have intrinsic flaws. The development of IoVs, or linked and autonomous cars, has generated enormous impetus for addressing transportation issues. Numerous prospective IoVs applications, including cooperative platooning, intelligent crossings, and cooperative perception, have the potential to drastically increase traffic congestion, fuel efficiency, and road safety. The following stricter KPIs must be met by 6G in order to fully realize the potential of IoVs:

1. Extremely high reliability
2. Extremely low latency
3. Extremely massive instant access: anytime, anywhere.
4. Extremely high throughput to support high volume of data transactions.

These KPIs present substantial problems that necessitate innovative thinking and edge-cutting communication technologies, like Long Term Evolution (LTE) enabled Connected vehicle-to-everything (C-V2X) and 5G New Radio (NR) V2X communications. This paradigm known as the IoVs has emerged as a result of advancements in vehicle communications. Using vehicle-to-everything (V2X) connections, the IoVs concept enables the integration of smart vehicles with the Internet, transportation infrastructure, and other road users. Vehicles are outfitted with sensors, control and computer units, communication, storage, and learning capabilities. Dedicated short-range communication (DSRC), which is based on IEEE 802.11, was for a long time the only V2X solution. The 3GPP adopted an edge-cutting technology in 2017 known as cellular-enabled V2X, or C-V2X, which can offer significantly higher system performance, higher spectral efficiency, higher range, reliability, and security than alternative technologies, enabling higher levels of safety for more road users. C-V2X relies on the capabilities of 4G, 5G, and future 6G cellular networks [8]. Two complementary transmission modes are used by C-V2X to offer a wide range of driving safety measures. These modes are the long-range network communications (C-V2X Mobile Communications) and the short-range direct communications (C-V2X Direct). Short-range communication between vehicles (V2V), vehicles and infrastructure (V2I), and vehicles and pedestrians are all included in C-V2X Direct (V2P).
We will discuss two scenarios to see how 6G is important to Internet of Vehicles (IoVs):

CASE 1:

The implementation of 6G technology is expected to play a significant role in enabling the mission-critical services of IoVs. Unlike 5G, which primarily focuses on communication between humans or machines, 6G is expected to expand beyond this by utilizing radio frequency (RF) for object sensing and positioning. The use of higher frequency bands in 6G, such as THz, will result in a more precise sensing and positioning resolution and increased data throughput through enhanced beamforming directionality. There is also a growing trend to incorporate intelligence into communication networks to handle the increasing complexity of network management. The combination of Communications, Computation, Positioning, and Sensing, known as CCPS, will allow for the deployment of a 6G system that satisfies service and application requirements while being cost-effective and able to be deployed at a large scale. A joint design of a 6G system with multiple cross-functional components is believed to provide a more efficient solution that benefits all CCPS functions, which are crucial for a IoV system to achieve autonomous driving.

CASE 2:

The integration of IoVs will improve the delivery and operation of 6G services. Roads play a crucial role in contemporary cities, just as vehicles do for families and society. The combination of stationary roads and mobile vehicles constitutes a significant part of our foundational infrastructure. The IoVs system, consisting of smart roads, Roadside Units (RSUs), and vehicles, can offer ample resources, physical space, and services for communication, computing, and intelligence.

The exceptional features of IoVs, such as their controlled mobility and simple deployment, can effectively support the Communication and Computing Platform Services (CCPS) of 6G systems by extending the infrastructure, monitoring the networks, and maintaining 6G networks to achieve its goal of widespread wireless intelligence while minimizing network operation costs. With the predicted substantial investments in 6G and IoVs infrastructures, the 6G communication and IoVs systems can be jointly designed, planned, and operated, leading to a better reuse of system resources and services.

As previously mentioned, the KPIs for IoVs and the functions of the control center for pedestrian safety (CCPS) play a significant role in shaping the design of a IoVs system, which includes AV's, RSUs, intelligent roads, and a control center. These functions are made possible by the use of advanced technologies such as THz, cell-free technology, and AI that are part of the 6G revolution. In turn, the IoVs infrastructure will also aid in the practical application and deployment of 6G technology. It's worth noting that traditional 4G and developing 5G technologies will continue to support IoVs.

1.2 THE PROGRESSION AND GROWTH OF VEHICLE-TO-EVERYTHING (V2X) AND INTERNET OF VEHICLES (IOV)

1.2.1 The progression and growth of Vehicle-to-Everything (V2X)

The ability to connect wirelessly is essential for IoV applications. V2X (vehicle to everything) technology has been developed and standardized as a key component of connected vehicles. This encompasses various forms of communication, including V2V (vehicle to vehicle), V2I (vehicle to infrastructure), V2P (vehicle to pedestrian), and V2N (vehicle to network). The first V2X technology was dedicated short-range communication (DSRC) which utilizes a 5.9 GHz frequency band that has been designated by the US Federal Communications Commission and the European ETSI for intelligent transportation systems (ITS). DSRC supports a range of ITS applications such as toll collection, vehicle safety, and in-vehicle entertainment. The technology behind DSRC is based on IEEE 802.11p, which is a modification to the IEEE 802.11 standard that provides wireless access in vehicular environments, with a specific focus on the physical and media access layers. Throughout the years, various V2X configurations have been created, both in academic and industrial settings, including ITS-G5 in Europe, which were mainly based on the IEEE 802.11p standard.

However, the use of random channel access in IEEE 802.11p has resulted in various limitations, such as lack of quality-of-service guarantees, long delays, weak connectivity between vehicles and infrastructure, and the need for large-scale deployment of infrastructure. To overcome these limitations, the 3rd generation partnership project (3GPP) introduced C-V2X, a cellular-based V2X solution that uses existing cellular communication infrastructure. Unlike IEEE 802.11p, cellular technologies have built-in quality of service mechanisms and are renowned for their mobility management capabilities. CV2X started with LTE, which is widely used and commercially available in everyday life as 4G.

The support for V2X services by 3GPP was defined in the Release 14 standards and can be provided through side link transmissions over the PC5 interface. The V2X transmissions are either managed through centralized scheduling by eNodeBs or gNBs, or through distributed scheduling by vehicles. While centralized scheduling ensures high broadcast reliability, it also leads to high signaling overheads due to frequent updates and resource allocation. On the other hand, distributed scheduling has better scalability with autonomous resource selection. In 3GPP

Release 15, the sidelink transmission for V2X was improved with features such as transmission diversity, carrier aggregation, and higher quadrature amplitude modulation. The 3GPP Release 16 specifies 5G NR based V2X support with additional features including unicast and multicast support, enhanced channel sensing, improved resource selection and QoS management, and congestion control [8], [9]. The 3GPP Release 17, which is expected to be completed in 2024, is currently in its planning stage and is aimed at enhancing side link efficiency, URLLC, positioning, and the use of relay and frequency range 2 over 6 GHz.

As C-V2X technology advances quickly, IEEE is keeping pace by forming a new Study Group called IEEE 802.11 Next Generation V2X in March 2018, which resulted in the creation of IEEE Task Group 802.11bd (TGbd) in January 2019. Both 802.11bd and 5G NR-V2X aim to minimize latency, increase efficiency and offer a higher level of reliability (i.e. 99.999%). However, their approaches differ, as 802.11bd must maintain compatibility on the same physical channel, while NR-V2X can communicate with its predecessor (LTE-V2X) through a different radio channel. Although the 802.11bd methodology offers significant benefits, it also presents significant design and performance challenges [8].

1.2.2 The progression and growth of Internet of Vehicles (IoV)

An additional timeline in the progress of IoVs technology is the rise in automation in vehicles. Starting from basic systems like forward collision warning and automatic braking, significant advancements in IoVs have been made, such as the availability of autonomous taxi services in the US and China. These taxis are capable of self-driving under specific circumstances, which is classified as Level 4 on the Automated Driving standard set by the Society of Automotive Engineers. Many well-known car manufacturers and technology companies, like Audi, Mercedes Benz, and Google, are currently working towards achieving full automation (Level 5 of the SAE automated driving standard).

Mobileye, the leading company in the field of Autonomous Driving Systems (ADS), identifies three critical components of autonomous driving technology: sensing, driving, and mapping. The objective of sensing is to create a comprehensive and accurate environment model with a 360-degree view. This includes identifying obstacles and road signs. Autonomous vehicles utilize a variety of sensors including cameras, ultrasound, lidar, short- and long-range radars. Each sensor has its own advantages and disadvantages. For instance, cameras offer high resolution, long-range detection, and the ability to recognize road signs and traffic lights, but they struggle in poor lighting and weather conditions. On the other hand, radars are effective in detecting objects at a distance and determining their speed, but they have low resolution and cannot identify the shape or height of objects [7]. Lidar has a broad field of view and accurate distance measurement capabilities, but it has limited resolution and is impacted by adverse weather conditions.

To ensure safe and dependable driving in various difficult road, weather, and lighting conditions, a variety of techniques utilizing a mix of sensors have been selected for vehicles. For instance, the latest Waymo autonomous driving platform 1 uses radar, lidar, and cameras, while Tesla autonomous vehicles employ cameras, radar, and ultrasonic but not lidar. The combination of multiple local sensors can improve sensing and safety, but they are still limited by the line of sight detection and the performance decreases with increasing object distance [6]. Despite significant investments by autonomous driving companies in vehicles with more powerful sensors, complete driving automation may not be possible due to challenges such as limitations in machine learning algorithms and sensors, difficult driving conditions and road emergencies, and the absence of redundancy in sensor safety and infrastructure support. Connected vehicles are seen as a crucial complementary technology for autonomous vehicles and are regarded as an integral part of full automation [5]. The integration of autonomous vehicles and connected vehicles can address most of the issues faced by autonomous vehicles alone and offer new road efficiency applications such as cooperative platooning and remote driving [10]. IoVs has received a lot of support from the automotive, telecommunication, academic, and public sectors in recent years.

1.3 SUPPORT FOR IOV WITH 6G TECHNOLOGY

A typical connected and automated vehicle system, as illustrated in Figure 2. below, consists of various important components that include IoVs on roads, roadside units (RSUs) equipped with communication, computing, and traffic control devices, smart roads with intelligent materials and sensors, and a transportation control center [5]. The RSUs will play a crucial role in collaborative mobility and computing. Additionally, unmanned aerial vehicles (UAVs) are being utilized in various scenarios to complement on-road vehicles, and connected unmanned aerial vehicles (CUAVs) are considered a part of the Connected-Autonomous-vehicle (CAV) system. The new capabilities that 6G systems can offer, such as the cloud-based positioning system (CCPS), can bring significant benefits to IoVs from both connectivity and computing perspectives, and are assisted by key technologies present in 5G networks, such as millimeter wave, massive multiple-input multiple-output, network virtualization function, and software-defined networks.

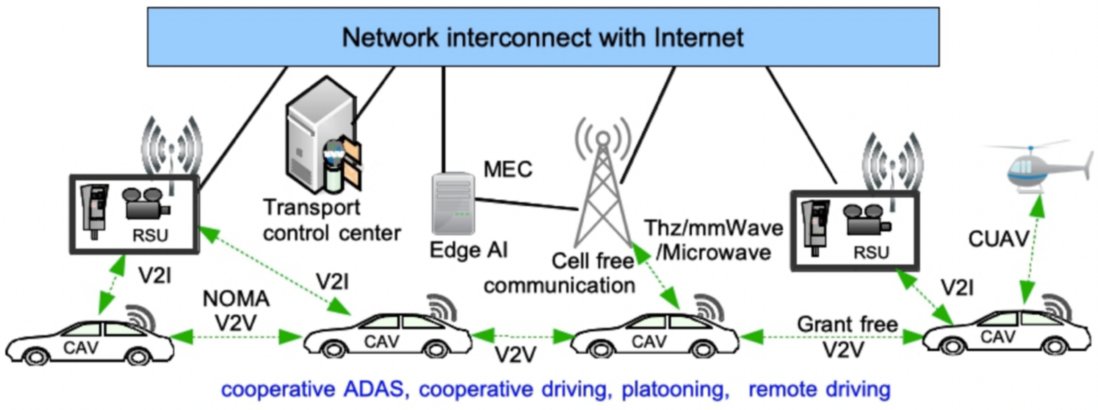


Figure 2. MEC: Mobile Edge Computing, RSU: Roadside Unit, CUAV: Connected Unmanned Ariel Vehicle, CAV: Connected & Autonomous Vehicle, V2V: Vehicle to Vehicle, V2I: Vehicle to Infrastructure [5]

1.3.1 Key Technologies for 6G Enabling.

The 6G technology is expected to bring about exceptional capacity, incredibly low latency, and extensive coverage [2]. It aims to deliver Tbps data rates and supports a variety of innovative applications that require agility, reliability, low latency, and energy efficiency. There are global research initiatives underway, exploring different technologies and roadmaps for 6G. The ITU formed a research group for 2030 network technologies in July 2018. In 2018, SK Telecom proposed a 6G technical roadmap with THz, cell-free networks, and airborne wireless platforms, and partnered with Samsung to work on joint 6G evolution technologies. China began its 6G research initiatives in 2018 with the aim of defining the Beyond 5G vision and requirements. Huawei also announced its 6G research study in 2019. The FCC sees 6G as featuring THz networks with multiple data transfer beams, which will require unprecedented network densification. In the following, some of the relevant technologies for IoVs are mentioned.

- THz for IoVs.
- Grant Free and Non-orthogonal Multiple Access (NOMA)

- Cell Free Communications
- Artificial Intelligence (AI) and Edge Intelligence (EI)

1.4 IMPROVEMENTS IN IOV TECHNOLOGY FOR 6G NETWORK

The implementation of 6G technology will greatly enhance the capabilities of IoVs, and in turn, IoVs will provide significant support for 6G in communication, networking, computing, and management services. This will allow 6G systems to achieve their goal of widespread wireless intelligence. The proposed 6G network architecture, as depicted in Figure 3. consists of a mobile core network, a space access network, edge clouds and servers, edge access networks, and end devices [5]. The mobile core network operates similarly to 5G networks' gNBs and manages mobility, network connections, and internet access. The space access network provides connectivity through satellites and UAVs. Networking and computing services are provided by central and edge servers, with features like network slicing and NVF. The edge access network connects mobile users and end devices to the mobile core network and edge servers. In addition to supporting mmWave and THz communications, the CAV system can also enhance 6G networks through edge and space access.

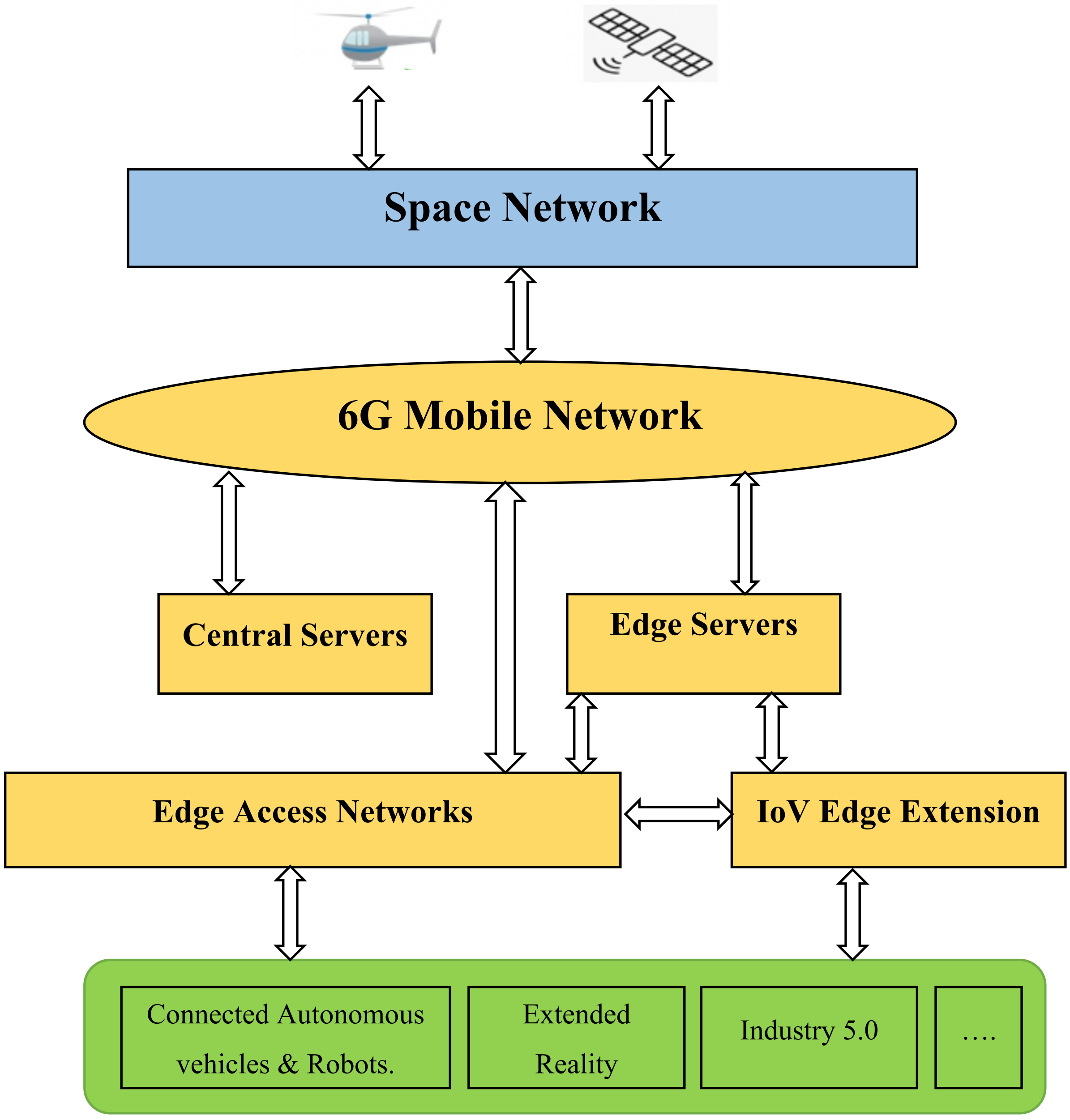


Figure 3. IoVs in 6G Network Architecture [5]

1.4.1 Extension of 6G Infrastructure

The deployment of 6G base stations requires a substantial amount of investment and is not easily adaptable to the constantly changing patterns of cellular network traffic. In emergency situations, such as natural disasters, the communication infrastructure may not be functional. To overcome these challenges, the use of surface connected and unmanned aerial vehicles (CAVs and CUAVs) as mobile platforms to extend the 6G communication infrastructure is proposed. These vehicles can be converted into mobile base stations (MBS) and deployed as needed to increase coverage. MBS provides a cost-effective and flexible solution to supplement the fixed 6G communication infrastructure. The MBSs can work together and adjust their services according to changing demands. While some MBSs may be dedicated to providing communication services, most are expected to primarily provide mobility services. The IoVs are intended to support the communication infrastructure by offloading some traffic and establishing connections with base stations during times when they are not needed for transportation. The patterns of road traffic and wireless network traffic make it possible for the IoVs to provide communication services when not needed for transportation, such as at night when there is less road traffic and more wireless network traffic. By utilizing the IoVs in this manner, the utilization of the IoVs can be improved and the operational costs of the 6G network can be reduced.

1.4.2 Mobile Vehicle Edge Computing

The main characteristic of 6G is the provision of intelligent services and applications through the use of a cloud computing continuum (3C) that includes remote clouds and edge clouds [5]. The 3C will handle edge computing applications that require various computing performance capabilities. However, due to the mobility of edge computing service users, computing and data migration will present a challenge within the 3C framework. With the growing computing power needed for high-level automation of internet of vehicles (IoVs), they can support 6G networks in providing mobile computing services. The abundance of storage space provides a chance for IoVs to be equipped with more resources for larger scale vehicle edge computing (VEC) services. These services can be deployed based on the needs of 6G network operators or individuals, serving other IoVs or other mobile users. VECs, with their autonomous mobility and ease of deployment, represent a flexible and valuable addition to 6G mobile edge computing (MEC).

1.4.3 Network Performance Monitoring and Sensing for 6G

The 6G network will have a complex design and will require sophisticated management and control. There is a growing trend towards automating networks and making them more resilient, both from the perspective of telecommunications equipment vendors and network operators. IoV technology can play a role in automating 6G networks. Accurate measurement and monitoring are crucial for achieving intelligent control and automation, which IoVs and smart roads equipped with intelligent sensors can help with by providing real-time updates on service quality and monitoring the network infrastructure. IoVs and smart roads can support the creation and operation of a 6G RIS-based smart communication environment. They can be deployed on demand to monitor network service quality, act as temporary base stations to enhance 6G communications, detect and even fix problematic outdoor 6G network components. With an increasing number of IoVs, they can provide a cost-effective alternative to manual network monitoring and improve 6G automation.

2. Federated Learning (FL) & Network Federated Learning (NFL)

Recently, data and computing resources are often distributed among end-user devices, various regions, or organizations. Due to legal restrictions, the data and computing resources cannot be combined or directly shared for machine learning (ML) purposes. To take advantage of these distributed resources, Network Federated Learning (NFL) & Federated learning (FL) has emerged as an effective solution for training ML models collaboratively, while still complying with laws and regulations and protecting data security and privacy [12].

ML involves automatically discovering patterns or models within data. These models are expressed as ML models, which consist of a model structure represented as a directed acyclic graph, processing units such as activation functions in deep neural networks (DNNs), and associated parameters or hyper-parameters. Input data is processed through the ML model to produce output, such as prediction or classification results, which is called the inference process. The ML model is created using training data through a training process, in which the parameters or structure of the model are adjusted using a training algorithm to improve performance, like accuracy or generalization capacity. This algorithm is also referred to as a ML algorithm and the duration of the training process is the training time.

ML training can be grouped into four categories based on the presence of labels in the training data [11]. The categories are supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. The focus of us is on supervised learning, where the training data includes both input features and corresponding labels. For example, each data point consists of (x, y), where x is the input feature and y is the desired output. In unsupervised learning, the training data only consists of input features and does not have output values. In semi-supervised learning, a portion of the training data has output values while the rest does not. Reinforcement learning involves considering observations from the previous iteration during each training iteration.

As the size of the training data increases, reaching terabytes or becoming too big to store on a single machine, the training process is carried out using distributed resources, referred to as distributed ML. This approach can significantly improve training speed and reduce the time needed for training. Distributed ML takes advantage of three forms of parallelism to parallelize the training process, including data parallelism, model parallelism, and pipeline parallelism. With data parallelism, the training data is divided into chunks and processed by different computing resources. Model parallelism involves each computing resource processing a copy of the data with different parts of the machine learning model. Pipeline parallelism combines the two, with each computing resource processing a portion of the training data and model, allowing for parallelized computation and communication at each node.

Figure 4. below shows machine learning consists of combination of three components [11]. So, we have.

a). Data: Data is the foundation of ML, and it refers to the information that is used to train a ML model. Data can come in various forms, such as images, text, audio, video, and numerical data. ML algorithms learn from data by extracting patterns and insights, which can then be used to make predictions or decisions. The quality and quantity of the data used to train a ML model directly impact its accuracy and reliability [14].

b). User Specific Model: A user-specific model in ML is a model that is trained on data that is specific to a particular user. These models are personalized to the individual user and can be used to make personalized recommendations, predictions, or decisions. User-specific models are often used in applications such as recommender systems, where the goal is to provide personalized recommendations to individual users based on their preferences and past behaviour [15].

c). Notion of loss function: A loss function is a mathematical function that is used to measure the difference between the predicted values of a ML model and the actual values. The goal of a ML model is to minimize the loss function, which indicates how well the model is performing on the given task. Different ML algorithms and models use different loss functions, depending on the specific problem being solved. For example, the mean squared error (MSE) loss function is commonly used in regression problems, while the cross-entropy loss function is used in classification problems [14].

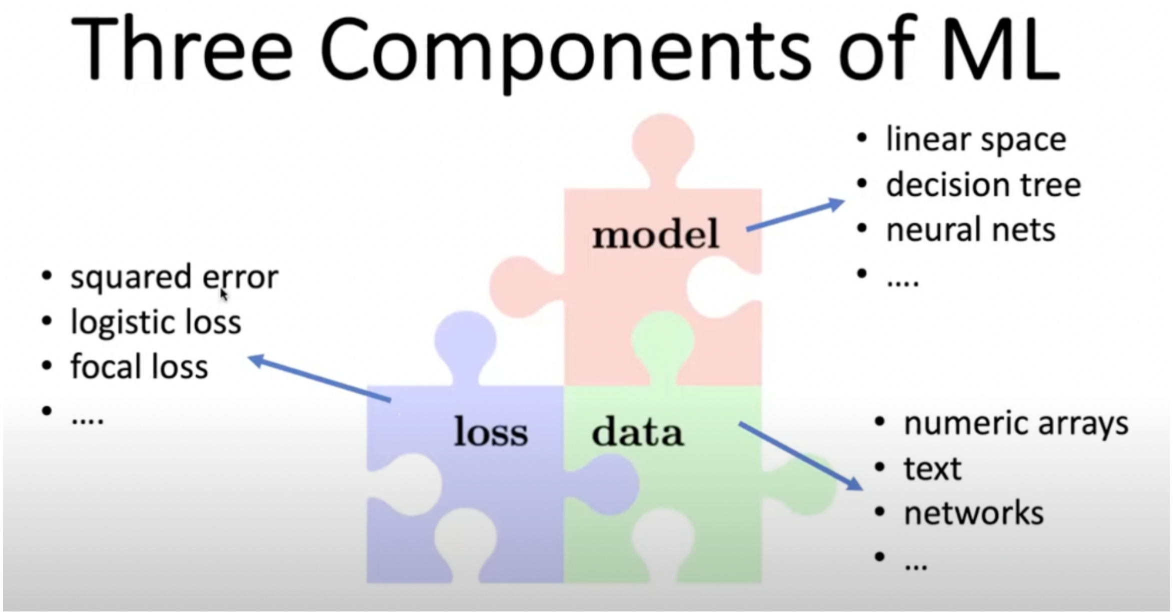


Figure 4. Components of Machine Learning [11]

2.1 FEDERATED LEARNING (FL)

FL is a collaborative ML method where multiple users work together to train a model without the raw data being centralized in a single location. The model used in FL is referred to as the FL model, which was initially designed to handle imbalanced and non-IID data found in mobile devices. Later, the idea of FL was expanded to cover distributed data from multiple sources, such as multiple organizations or regions [16]. FL systems are utilized throughout the FL model's life cycle and serve as a distributed system to manage the distributed training process with dispersed resources.

FL is a unique form of distributed machine learning that sets itself apart from other methods in three keyways. Firstly, FL prohibits direct communication of raw data, while other methods have no such restrictions. Secondly, FL utilizes the computing resources across multiple regions and organizations, while other methods only use a single server or cluster in one region owned by one organization. FL enables collaboration between different organizations. Thirdly, FL uses encryption and other security measures to protect data privacy and security [17], which is not a priority for other methods. This is important as the loss of information can result in financial and reputational damage.

2.1.1 Federated Learning model cycle

The FL model life cycle involves several stages [18], which are as follows:

Data Collection: The first stage of the FL model life cycle is data collection, where data from multiple clients (such as mobile devices, laptops, etc.) is collected. Each client has its own data that is stored locally and cannot be shared due to privacy and security concerns.

Model Initialization: In this stage, a global model is initialized, which will be trained on the federated data from the clients. The model parameters are randomly initialized, and the model architecture is defined.

Model Training: In this stage, the global model is trained on the federated data from the clients. The model parameters are updated based on the local data on each client, without sharing the raw data. This is done using techniques such as secure aggregation and differential privacy.

Model Evaluation: After the model is trained, it is evaluated to determine its accuracy and performance. This is done by comparing the model's predictions with the actual data on a holdout set or by using other evaluation metrics.

Model Deployment: Once the model is evaluated and found to be accurate, it can be deployed for use in the real world. The deployed model can be used for various applications, such as making predictions, classifications, or recommendations.

Continuous Monitoring: The deployed model is continuously monitored to detect any changes in the data distribution or to identify any drift in the model's performance. This is important to ensure that the model remains accurate and relevant over time.

Model Re-training: If the model's performance starts to deteriorate, it can be re-trained using updated data and the latest version of the algorithm. This process can be repeated as needed to keep the model up-to-date and accurate.

In conclusion, the FL model life cycle is an iterative process that involves data collection, model initialization, training, evaluation, deployment, continuous monitoring, and re-training. The goal is to maintain an accurate and up-to-date model that can be used for various applications while preserving the privacy and security of the data.

2.1.2 Architecture of Federated Learning System

A FL system is a distributed ML framework where multiple participating devices (e.g. smartphones, IoT devices) train a shared model simultaneously without exchanging raw data with a central server. The functional architecture of a federated learning system typically consists of the following components:

Client devices: These are the participating devices that hold the local data and perform local model training and updates.

Aggregation Server: This is a central node that manages the overall training process and coordinates the communication between the client devices.

Model: This is the shared model that is trained on the federated data from the client devices. The server holds the current version of the model and distributes it to the client devices for training.

Communication protocol: This defines the rules for exchanging messages between the client devices and the server. This includes sending model updates from the client devices to the server and sending the latest version of the model from the server to the client devices.

Aggregation function: This is the function used by the server to aggregate the updates received from the client devices and update the shared model.

The architecture is illustrated as follows:

$$Client\ devices\ (1, 2, \ldots, n)\ <--->\ Server\ <--->\ Model$$

In this approach, the client devices, represented by (1, 2, ..., n), communicate directly with the server to perform model training. The client devices perform local model training using their

local data, and send their model updates to the server. The server aggregates the updates from all the client devices and updates the shared model. The updated model is then distributed back to the client devices for the next round of training. This process continues until the shared model converges to an optimal solution, providing a highly scalable and privacy-preserving machine learning solution.
In Figure 5. A generalized FL architecture is presented with basic FL elements [12].

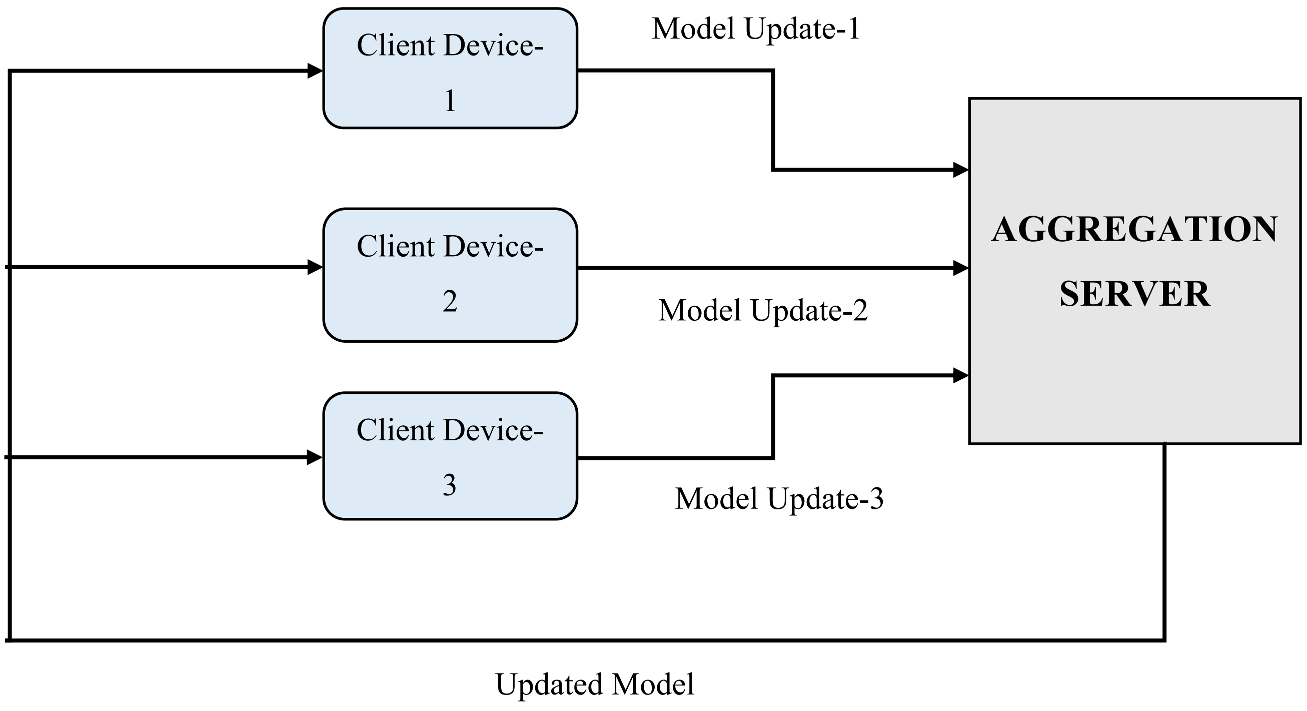


Figure 5. Federated Learning Architecture [12]

2.2 DISTRIBUTED TRAINING

Distributed training refers to a ML process in which the training of a model is divided and executed across multiple computers or machines, working together as a team. The goal of distributed training is to achieve a faster and more efficient training process by utilizing the processing power and memory of multiple machines.

In a distributed training process, a large dataset is divided into smaller parts and each machine is assigned a portion of the data to work with. Each machine trains a separate model on its portion of the data, and then the models are combined to form a single, more accurate model.

The result is a more robust and powerful model that can be used to make predictions with high accuracy.

Distributed training is particularly useful in cases where the dataset is too large to be processed by a single machine. It also helps in reducing the time required for training, as multiple machines can work in parallel. This can be particularly useful in deep learning and other complex ML algorithms that require a lot of computational resources.

Overall, distributed training offers many advantages, including increased speed and accuracy, as well as improved scalability and flexibility, making it an increasingly popular approach in modern ML [17].

2.2.1 Aggregation

Aggregation algorithms are utilized in combination with horizontal FL and data parallelism to aggregate the models or gradients that result from the forward and backward propagation processes carried out on each computing resource. These algorithms come in three forms: centralized, hierarchical, and decentralized. Centralized algorithms rely on a central server, known as a parameter server, to coordinate and control the distribution of computing resources. Hierarchical algorithms use multiple parameter servers to handle the aggregation of models. Decentralized algorithms, on the other hand, involve each computing resource conducting calculations according to a predefined protocol, without the need for a central server.

Centralized aggregation in federated learning refers to a method where the training data from multiple devices is collected and combined in a single location, usually a server, for the purpose of model training.

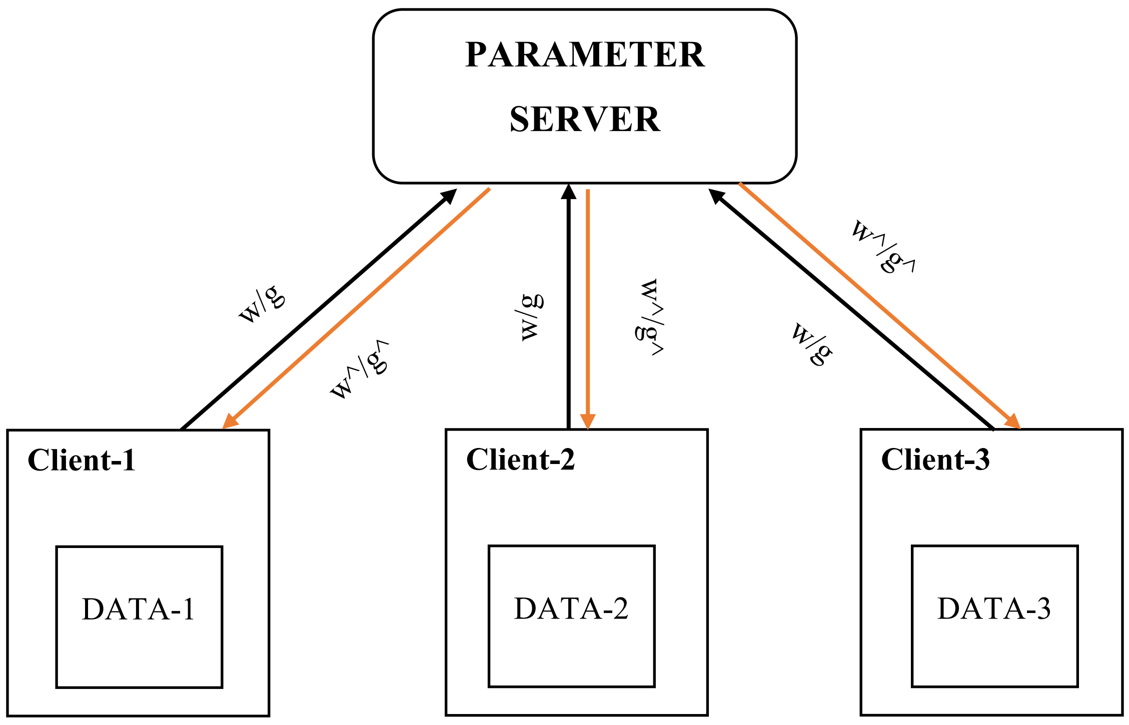


Figure 6. Centralized Aggregation [12]

Figure 6. shows how centralized aggregation works [12]. In a federated learning scenario, many devices, such as smartphones or IoT devices, have their own data that they can use to train a machine learning model. Rather than sending the raw data to a central server for model training, the devices send their locally trained model parameters (w/g) to the central server. The central server then aggregates the parameters from all devices and uses them to update the global model (w^/g^).

The central server is responsible for combining the parameters from all the devices and averaging them to obtain the updated model. This process is repeated multiple times until the global model converges to an optimal solution. The updated global model is then sent back to the devices for further fine-tuning on their local data.

Centralized aggregation is a simple and efficient method for federated learning, but it raises privacy concerns since all the data must be sent to a central server. To address this issue, alternative methods such as secure multi-party computation and homomorphic encryption have been proposed.

2.3 NETWORK FEDERATED LEARNING (NFL):

NFL refers to a distributed ML approach that utilizes a decentralized network of nodes to perform ML tasks. The main objective of this approach is to overcome the limitations of traditional ML, such as limited data access, low computational power, and security concerns. NFL enables multiple nodes to work together to build a model that can be trained and updated in a decentralized manner, thereby preserving privacy and security. The introduction of NFL presents a promising solution for overcoming the challenges of traditional ML and enabling advanced ML techniques for real-world applications.

The basic idea behind NFL is to leverage the data and computing resources available in the network, while preserving the privacy and security of the data. In this approach, each node in the network contributes its own data to train the model, without sharing it with the other nodes. The model parameters are then aggregated and updated using a consensus algorithm, such as Federated Averaging. NFL approach combines the computational power and data resources of multiple remote nodes within a network to train a shared machine learning model. It enables the development of models that are trained on a larger and more diverse dataset, resulting in higher accuracy and better generalization compared to traditional single-node training.

NFL has several advantages over traditional single-node training. Firstly, it enables the training of models on large and diverse datasets that would otherwise be infeasible to store and process on a single node. Secondly, it reduces the amount of data that needs to be transmitted over the network, as only the model parameters are shared, not the raw data [19]. This not only conserves bandwidth, but also ensures the privacy and security of the data.

Below are some advantages of NFL over traditional FL:

Improved performance: NFL enables the aggregation of a larger amount of data from different sources, which leads to improved model accuracy and generalization. A study conducted by Google Research showed that NFL outperformed traditional FL in terms of model accuracy and convergence speed on various datasets.

Better privacy: NFL enables the separation of data and model updates across different organizations, which enhances privacy and data confidentiality. This approach reduces the risk of sensitive information exposure or data leakage, which is a significant concern in many industries.

Enhanced scalability: NFL enables the training of models across a larger number of devices and organizations, which increases the scalability of the system. This approach also reduces the burden on individual devices and organizations, which can improve the training efficiency and reduce the training time.

More flexible architecture: NFL allows for a more flexible and adaptable architecture that can accommodate various training scenarios and requirements. This approach can be used to train models across multiple domains, such as healthcare, finance, and transportation, and can be customized to meet specific data privacy and security requirements.

One of the key challenges in Network Federated Learning is the ability to handle the variability and heterogeneity of the data and computing resources in the network. For example, different nodes may have different amounts of data, different data distributions, and different processing capabilities. To overcome these challenges, various algorithms and techniques have been developed, such as federated transfer learning, federated meta-learning, and federated domain adaptation [20].

Overall, NFL is a promising approach for developing ML models that are more accurate and generalize better, while preserving privacy and security. It has a wide range of applications in areas such as Internet of Things (IoT), Healthcare, and Finance. For example, in healthcare, doctors can use NFL to train models on patient data without revealing sensitive information. In finance, banks can use NFL to improve their fraud detection models without sharing customer transaction data.

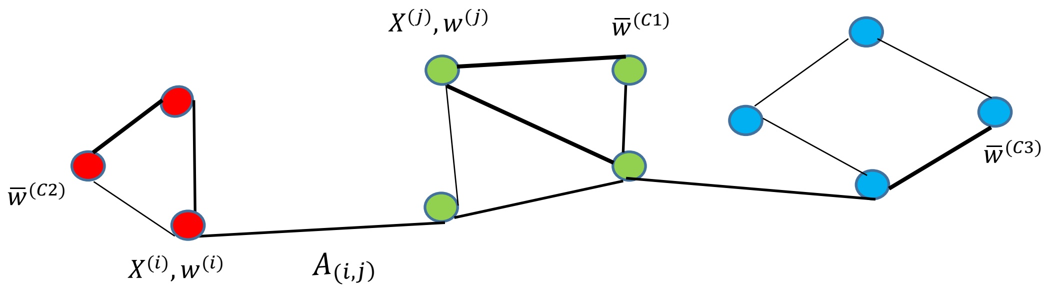


Figure 7. Empirical Graph for Network Federated Learning

Figure 7. shows an undirected empirical graph $G = (V, E)$ to represent networked data and corresponding models [21]. Each node $i \in V$ has a local dataset $X^{(i)}$ and model weights $w^{(i)}$ scored by a local loss function $L_i(w^{(i)})$ based on the local dataset. Nodes are connected by weighted edges $\{i,\ i'\}$ if they have datasets with similar statistical properties, with the weight $(A_{\mathrm{i,i}})$ indicating the degree of similarity. We assume that nodes in the same cluster $C(c) \subseteq V$ have nearly identical optimal parameter vectors. The empirical graph is divided into three disjoint clusters C(1), C(2), and C(3). NFL method can learn the partition based on the local datasets and network structure of G.

3. Network Graphical Model

We have created a framework for NFL, where data is collected in decentralized collections and has an inherent network structure based on similarities between local datasets, such as spatio-temporal proximity or statistical dependencies. Our approach uses a generalized total variation (GTV) minimization to unify and expand upon existing federated multi-task learning methods. We have developed an algorithm that is suitable for distributed computing environments like edge computing over wireless networks and can handle limited computational resources. We have identified conditions for local models and network structure that allow our algorithm to learn nearly optimal local models, and our analysis shows a relationship between the convex geometry of local models and the cluster geometry of the network structure.

Virtual connections that form a network structure and are significant in various application domains. Pandemic management and the Internet of Things are two such domains where such networked data is increasingly important [23]. Local datasets created by smartphones, wearables or industrial IoT devices contribute to the network, which is established through physical or virtual connections.

FL applications face a significant challenge due to the diversity of local datasets [24]. The statistical characteristics of each local dataset may differ greatly, making it difficult to model them as independent and identically distributed. Each local dataset requires a distinct learning task, where the parameters of a local model are learned or optimized. Here we explore an optimization technique that customizes local models to the statistical properties of their respective local datasets. This technique involves regularized empirical risk minimization, where a measure of the variation of local model parameters is used as a regularizer. The resulting optimization problem is solved using a Generalised Total Variation (GTV) Minimisation method, which can be implemented as message passing over the network structure of local datasets.

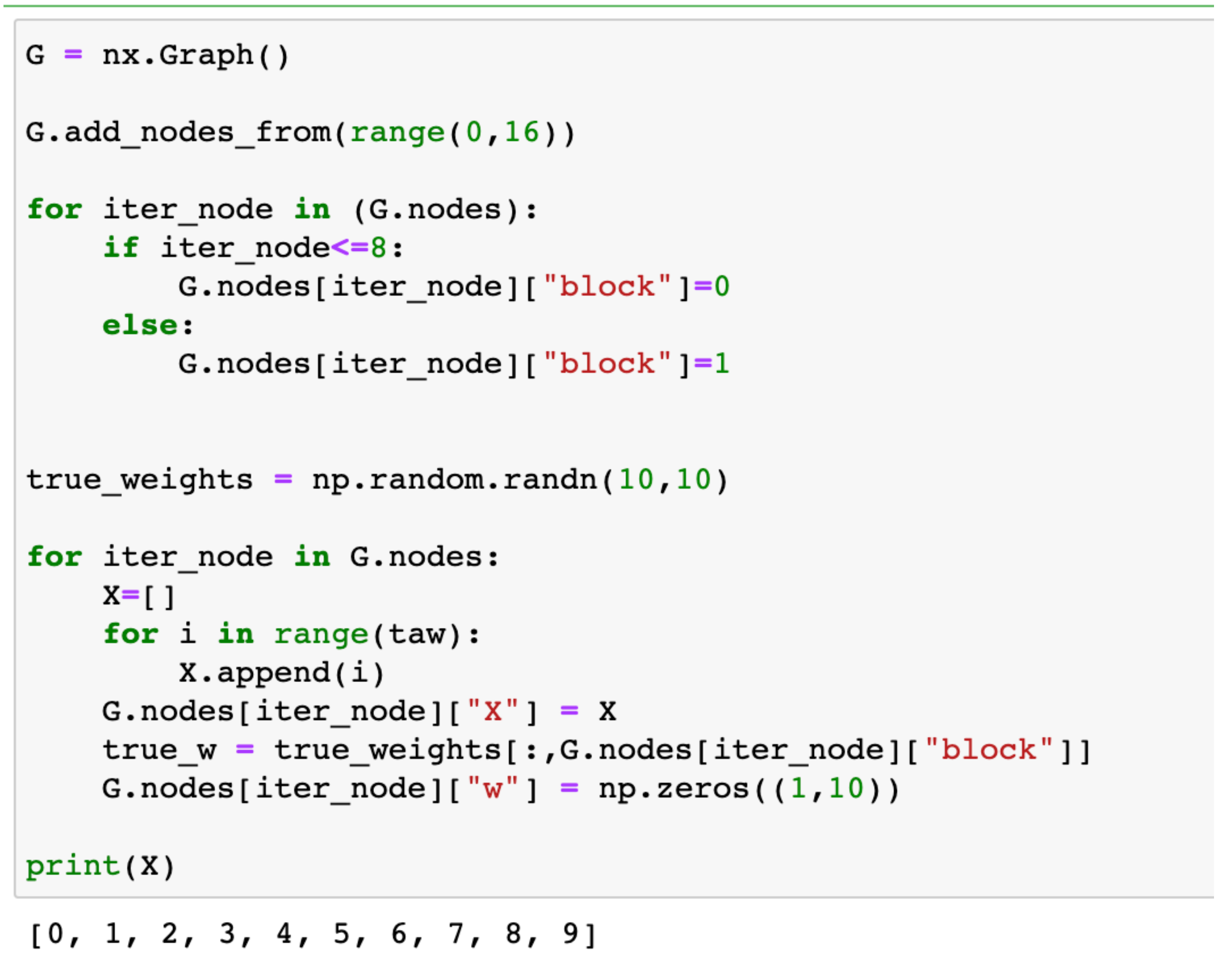
The concept of Clustered FL is focused on dealing with the diversity of local datasets through the application of different forms of a clustering assumption [25]. This assumption stipulates that local datasets and their associated learning tasks must be grouped into a small number of non-overlapping subsets or clusters. Those local datasets that belong to the same cluster will share similar statistical properties and will therefore have optimal parameter values for their corresponding local models that are also similar. The primary contribution of this thesis is an accurate characterization of the cluster structure and local model geometry of local datasets that enable the pooling of those datasets that are statistically homogeneous.

The approach we are using for clustered FL is distinct from previous methods. We make use of known similarities between local datasets, which are represented through weighted undirected edges in an empirical graph [26]. NFL applies this graph to cluster or group local datasets into similar training sets, with the goal of achieving a nearly homogenous ("i.i.d.") result. A model is then trained for each resulting cluster, which is used for all local datasets within the same cluster. The clustering of local datasets is determined by the connections within the empirical graph and is implemented through distributed optimization methods. The NFL methods can be understood as advanced methods for clustering graphs. This involves merging the structure of the graph with the statistical characteristics of the local datasets to group the nodes in the real-world graph. Additionally, the NFL can also be viewed as a distinct kind of multi-task learning. Each cluster of the local datasets creates a separate learning task, with the objective of identifying the optimal parameters for all nodes in the cluster [27]. Finally, NFL can also be interpreted as a type of network optimization.
It is important to note that our approach requires a practical selection for the empirical graph of networked data. This empirical graph should demonstrate the connectivity of nodes, which represent local datasets, and reflect the clustering of local datasets that share similar statistical properties. In some application domains, a suitable empirical graph is readily available through spatio-temporal proximity, functional relations, physical models, or the underlying distributed computing infrastructure [28]. However, if a natural choice for the empirical graph is not present, we can attempt to learn the network structure through methods that measure statistical similarity between two datasets [29].

3.1 DATA (THE MOBILITY MODEL)

To model the movement of vehicles within a city, a graph representation of the area was used. The graph was denoted as G = {E, V}, where the vertexes represented the cells within the city, and the edges represented the demand shift between adjacent cells. In other words, the edges represented the mobility model of the vehicles, showing how they move from one cell to another. In addition to this, certain hypotheses were taken into account to ensure the accuracy of the model. These hypotheses were considered to be important in providing an accurate and realistic representation of the movement of vehicles within the city. By taking into account these factors, the model was able to provide an effective representation of the mobility of vehicles, helping researchers to better understand and analyse the dynamics of traffic within the city.

The Figure 8. below shows the code that how we construct a network graph taking 16 nodes and 10-time instances for our project. So, in this case we will see further that we have at each node the behaviour of random change of traffic at different time instances. Here we see we have generated our Feature Matrix “*X*” i.e., the time-instants at each node. Now for label vectors “y” at each node we have to implement the mobility model which we will see further.



```
G = nx.Graph()

G.add_nodes_from(range(0,16))

for iter_node in (G.nodes):
    if iter_node<=8:
        G.nodes[iter_node]["block"]=0
    else:
        G.nodes[iter_node]["block"]=1


true_weights = np.random.randn(10,10)

for iter_node in G.nodes:
    X=[]
    for i in range(taw):
        X.append(i)
    G.nodes[iter_node]["X"] = X
    true_w = true_weights[:,G.nodes[iter_node]["block"]]
    G.nodes[iter_node]["w"] = np.zeros((1,10))

print(X)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Figure 8. Implementation of Network Graph

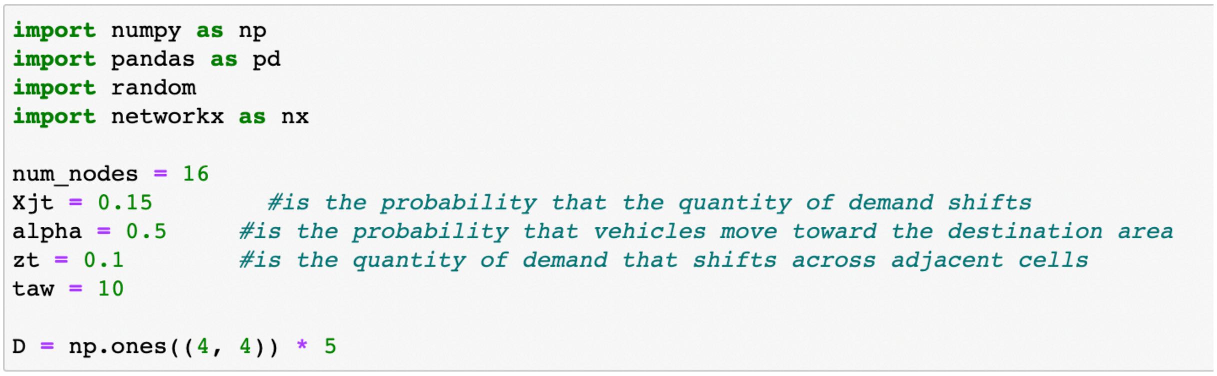
To implement the mobility model, we considered several hypotheses in our analysis. Firstly, we assumed that the demand vector would have the same number of cells as the topology. Additionally, we assumed that demands could only shift between adjacent cells, including those on the perimeter of the region being studied. Furthermore, we assumed that the total demand would remain constant over time. This means that if part of the demand leaves the system by crossing the perimeter, an equal amount of demand will enter from the opposite border. In mathematical terms, this is expressed in (1).

$$\sum_{C_j \in A} d_j^t = \|\boldsymbol{D}\|_1 \qquad \forall t \in T \qquad (1)$$

where $\|\boldsymbol{D}\|_1$ represents the sum of all the demands in all cells and is assumed to remain constant throughout the entire time period.

The model used in this study assumes that only a certain number of vehicles can move to adjacent cells from a starting point at a given time slot. This means that some vehicles, such as parked cars or vulnerable road users, are stationary or move slowly. The previous selection of a vehicle may or may not result in its movement to another cell, as there may be reasons for it to stop. Furthermore, only a portion of the selected vehicles are assumed to move towards their usual destination, while the rest may move back for other reasons. The model is based on three probabilistic parameters and is useful for mapping behaviour in urban scenarios. The study focuses on a high-density area, specifically the downtown area, which attracts vehicles during certain periods of the day and from which vehicles depart during other periods. The cell representing the downtown area is referred to as C_{DT} and is identified by its coordinates (x^{DT}, y^{DT}).
The mobility model in Algorithm 1 is presented as follows: At a certain timeslot 't', a number of demands or vehicles are given. By focusing on the jth cell, it is assumed that only a fraction of them, $\xi_j^t \in [0,1]$, are able to move to an adjacent cell. Thus, the vehicles that can potentially move from the jth cell to adjacent cells at time t are determined by $\lceil \xi_j^t \ . \ \mathrm{d}_j^t \rceil$, which is rounded up to an integer to ensure that there are always potential moving vehicles. This means that some road users may be stationary or slow-moving, such as parked cars or vulnerable road users, while others can move to a different cell. The chosen $\lceil \xi_j^t \ . \ \mathrm{d}_j^t \rceil$ vehicles are assumed to move with a certain probability, denoted as $\chi_j^t \in (0, 1]$, while the rest will remain in the same cell with probability $(1 - \chi_j^t)$.

The parameters that we set for demands shifting are represented below in a Figure 9.



```python
import numpy as np
import pandas as pd
import random
import networkx as nx

num_nodes = 16
Xjt = 0.15        #is the probability that the quantity of demand shifts
alpha = 0.5     #is the probability that vehicles move toward the destination area
zt = 0.1        #is the quantity of demand that shifts across adjacent cells
taw = 10

D = np.ones((4, 4)) * 5
```

Figure 9. Parameters for Mobility Model

Now below Algorithm.1 discusses the presence of a highly attractive cell, known as the downtown area, which vehicles tend to gravitate towards during certain hours of the day. To accurately represent this behavior, a probability factor, α_j^t is introduced to determine the likelihood of vehicles moving towards the downtown cell from their current location (cell j) at a specific time (t). The probability of vehicles moving in the opposite direction is represented by $(1 - \alpha_j^t)$.

To account for the shifting demands of each cell, two random values are utilized to determine the probability of vehicles moving towards the destination area. If the random variable aligns with the probability of moving towards the downtown area, either the SHIFTTOWARD() or SHIFTFARTHER() function is executed, based on the comparison of the two probabilities.

Algorithm_1 (Pseudo Algorithm for Mobility Model)	
1.	$\mathbf{D}^{(t)}$ is a list of demands at time *t*
2.	C_{DT} is the destination cell
3.	C_j is the *j*-th cell
4.	$\xi_j^t \in [0,1]$, is the quantity of demand that shifts across adjacent cells
5.	$\chi_j^t \in (0, 1]$, is the probability that the quantity of demand shifts
6.	$\alpha_j^t \in (0, 1]$, is the probability that vehicles move toward the destination area
7.	*randShift* is a random number between (0, 1]
8.	*randAlpha* is a random number between (0, 1]

```
9.  for all $d_j^t \in \mathbf{D}^{(t)}$ do
10.     if $randShift \leq \chi_j^t$ then
11.         if $randAlpha \leq \alpha_j^t$ then
12.             SHIFTTOWARD($d_j^t$ , $C_j$ , $C_{DT}$ , $\xi_j^t$ )
13.         else
14.             SHIFTFARTHER($d_j^t$ , $C_j$ , $C_{DT}$ , $\xi_j^t$ )
15.         end if
16.     end if
17. end for
```

The functions SHIFTTOWARD() and SHIFTFARTHER() are explained in Algorithms 2 and 3. It is assumed that the cell C_ψ is closer to the destination than the cell C_j, while the cell C_ω is further away. In the SHIFTTOWARD() function, the demands for the next time step are updated by increasing the demand for the cell closer to the destination and decreasing it for the originating cell C_j. Similarly, in the SHIFTFARTHER() function, the demands for the next time step are increased for the more distant cell(s) and decreased for the originating cell

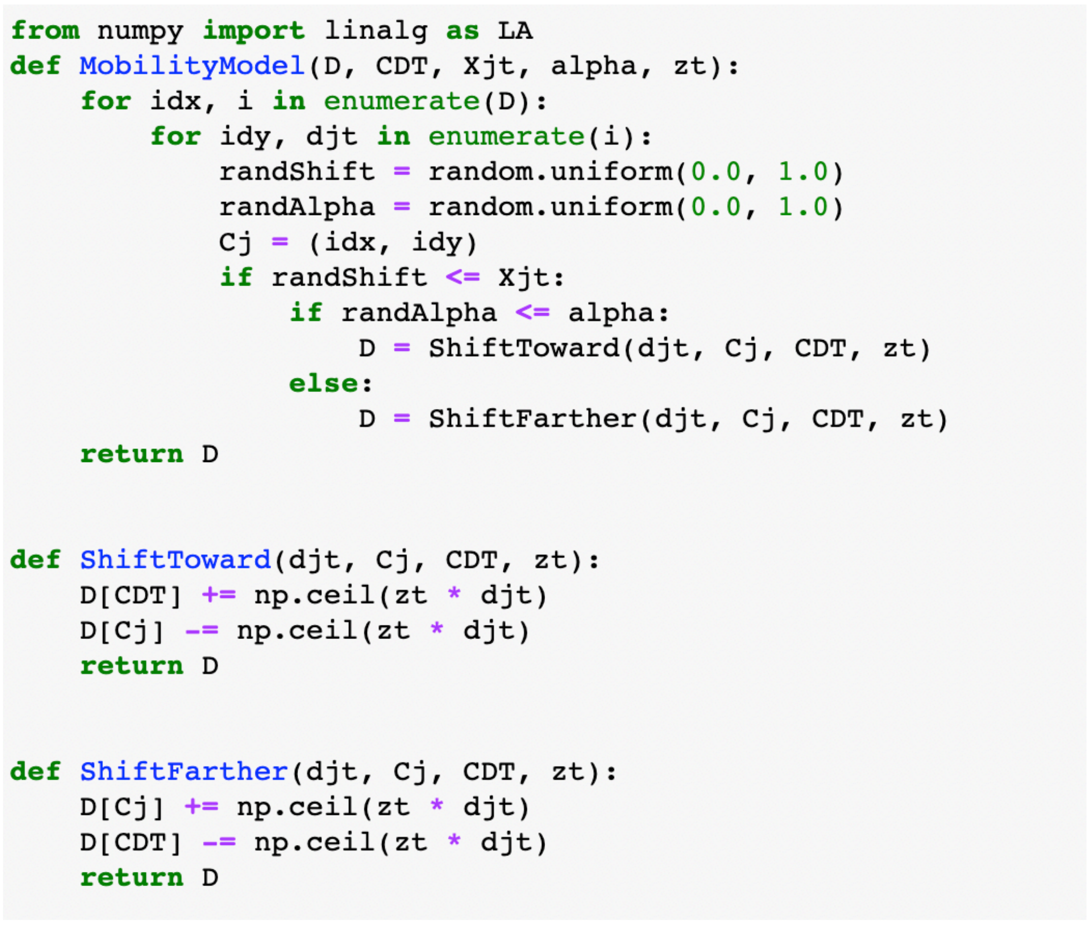
Algorithm_2 (Pseudo-code for Shift Toward Function)

```
1. functionSHIFTTOWARD($d_j^t$ , $C_j$ , $C_{DT}$ , $\xi_j^t$)
2.     $d_\psi^{t+1}$ += $\lceil \xi_j^t \cdot d_j^t \rceil$
3.     $d_j^{t+1}$ −= $\lceil \xi_j^t \cdot d_j^t \rceil$
4. end function
```

Algorithm_3 (Pseudo-code for Shift Farther Function)

```
1. functionSHIFTFARTHER($d_j^t$,$C_j$,$C_{DT}$,$\xi_j^t$)
2.     $d_\omega^{t+1}$ += $\lceil \xi_j^t \cdot d_j^t \rceil$
3.     $d_j^{t+1}$ −= $\lceil \xi_j^t \cdot d_j^t \rceil$
4. end function
```

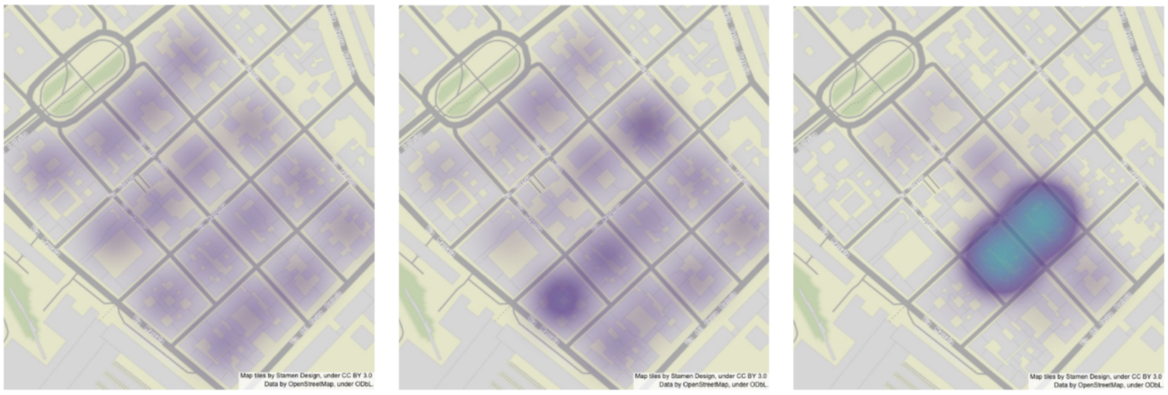
Figure 10. below shows the implementation of the above 3 algorithms in python code by considering all the parameters that we stated above.



```python
from numpy import linalg as LA
def MobilityModel(D, CDT, Xjt, alpha, zt):
    for idx, i in enumerate(D):
        for idy, djt in enumerate(i):
            randShift = random.uniform(0.0, 1.0)
            randAlpha = random.uniform(0.0, 1.0)
            Cj = (idx, idy)
            if randShift <= Xjt:
                if randAlpha <= alpha:
                    D = ShiftToward(djt, Cj, CDT, zt)
                else:
                    D = ShiftFarther(djt, Cj, CDT, zt)
    return D


def ShiftToward(djt, Cj, CDT, zt):
    D[CDT] += np.ceil(zt * djt)
    D[Cj] -= np.ceil(zt * djt)
    return D


def ShiftFarther(djt, Cj, CDT, zt):
    D[Cj] += np.ceil(zt * djt)
    D[CDT] -= np.ceil(zt * djt)
    return D
```

Figure 10. Mobility Model

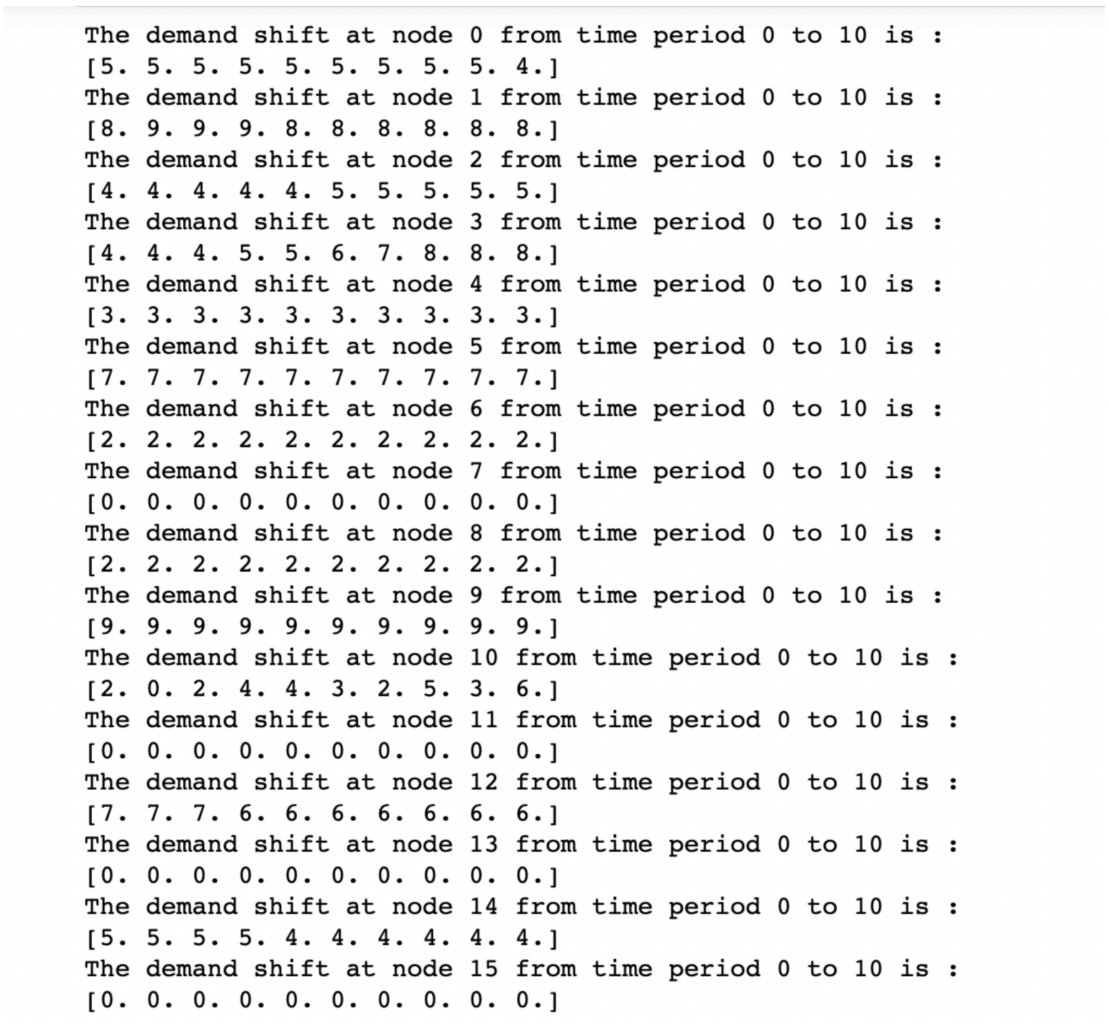
The given matrix in the Figure 11. below is used to explain a concept and help readers better understand it. The matrix displays a geographic heatmap of a neighborhood that is part of Rome, Italy. In the initial scenario, which is shown in Figure 11 a), the demands are spread evenly throughout the neighborhood [22]. However, as time passes, the demands begin to concentrate in the downtown area. The change in demand is reflected in the heatmap, which shows a greater concentration of demand in the downtown area.


Map tiles by Stamen Design, under CC BY 3.0
Data by OpenStreetMap, under ODbL.

a) Period (t=0) b) Period (t=1) c) Period (t=15)

Figure 11. Heat Maps of Traffic Density at different time periods [22]

Now we will see the results of mobility model implementation in the Figure 12. below. The figure shows that how using the mobility model the quantity of demand of vehicles changes at each node/area from time instant 0 to 10 for our project. So, our feature matrix “X” will be the time instant (0-10) and the label vector “y” will be the data of demand shift of vehicles at each node for time instant (0-10).



```
The demand shift at node 0 from time period 0 to 10 is :
[5. 5. 5. 5. 5. 5. 5. 5. 5. 4.]
The demand shift at node 1 from time period 0 to 10 is :
[8. 9. 9. 9. 8. 8. 8. 8. 8. 8.]
The demand shift at node 2 from time period 0 to 10 is :
[4. 4. 4. 4. 4. 5. 5. 5. 5. 5.]
The demand shift at node 3 from time period 0 to 10 is :
[4. 4. 4. 5. 5. 6. 7. 8. 8. 8.]
The demand shift at node 4 from time period 0 to 10 is :
[3. 3. 3. 3. 3. 3. 3. 3. 3. 3.]
The demand shift at node 5 from time period 0 to 10 is :
[7. 7. 7. 7. 7. 7. 7. 7. 7. 7.]
The demand shift at node 6 from time period 0 to 10 is :
[2. 2. 2. 2. 2. 2. 2. 2. 2. 2.]
The demand shift at node 7 from time period 0 to 10 is :
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
The demand shift at node 8 from time period 0 to 10 is :
[2. 2. 2. 2. 2. 2. 2. 2. 2. 2.]
The demand shift at node 9 from time period 0 to 10 is :
[9. 9. 9. 9. 9. 9. 9. 9. 9. 9.]
The demand shift at node 10 from time period 0 to 10 is :
[2. 0. 2. 4. 4. 3. 2. 5. 3. 6.]
The demand shift at node 11 from time period 0 to 10 is :
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
The demand shift at node 12 from time period 0 to 10 is :
[7. 7. 7. 6. 6. 6. 6. 6. 6. 6.]
The demand shift at node 13 from time period 0 to 10 is :
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
The demand shift at node 14 from time period 0 to 10 is :
[5. 5. 5. 5. 4. 4. 4. 4. 4. 4.]
The demand shift at node 15 from time period 0 to 10 is :
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Figure 12. The demand shift of vehicles at each node for different time instances

So, we have the data set for our model. A data set is basically a collection of data, usually organized in a tabular format with rows and columns. It is a set of information that is collected and stored for analysis, interpretation, and processing. Data sets can be used for a variety of purposes, including research, analysis, statistics, ML, and more. They can be either structured or unstructured and can be collected from a variety of sources, such as surveys, sensors, social media, and databases. A data set can contain any type of data, including numerical, categorical, and textual data [14].

Now the last part of Network Graphical Model will be the choice of Mobility Model. In our case we have considered Manhattan Grid Mobility Model.

3.2 MANHATTAN GRID MOBILITY MODEL

The Manhattan Grid Mobility Model (MGMM) is a widely used mobility model in the field of wireless communication and networking. It is named after the street grid system of Manhattan, New York, which is characterized by a regular, rectangular street pattern.

The MGMM is a realistic model that captures the movement of vehicles, pedestrians, and other mobile entities in an urban environment. It is commonly used to simulate and evaluate the performance of wireless communication protocols, such as routing, medium access control, and congestion control, in urban environments[30].

The basic assumption of the MGMM is that mobile entities move along the streets in a predetermined direction at a constant speed. The movement of each entity is modeled as a sequence of straight-line segments, each of which represents the entity's movement along a street segment [31]. The length of each segment is determined by the speed of the entity and the duration of the segment.

The MGMM uses a set of parameters to control the movement of entities. These parameters include the density of entities, the size of the simulation area, the average speed of entities, and the duration of the simulation. By adjusting these parameters, the MGMM can be used to simulate different scenarios, such as rush hour traffic, pedestrian congestion, and emergency vehicle response times.

One of the key advantages of the MGMM is its simplicity. The model is easy to implement and can be used to generate large amounts of mobility data in a short amount of time. This makes it an ideal choice for large-scale simulations and performance evaluations.

However, the MGMM has several limitations [32]. First, it assumes that all entities move at a constant speed and in a predetermined direction, which may not accurately capture the behavior of real-world entities. Second, the model does not consider the effects of traffic lights, stop signs, and other traffic control measures, which can significantly impact the movement of entities. Finally, the model does not capture the dynamic nature of urban environments, such as changes in traffic patterns and road closures.

In summary, the Manhattan Grid Mobility Model is a popular and widely used model for simulating the movement of mobile entities in urban environments. While it has some limitations, it remains a valuable tool for evaluating the performance of wireless communication protocols in realistic scenarios.

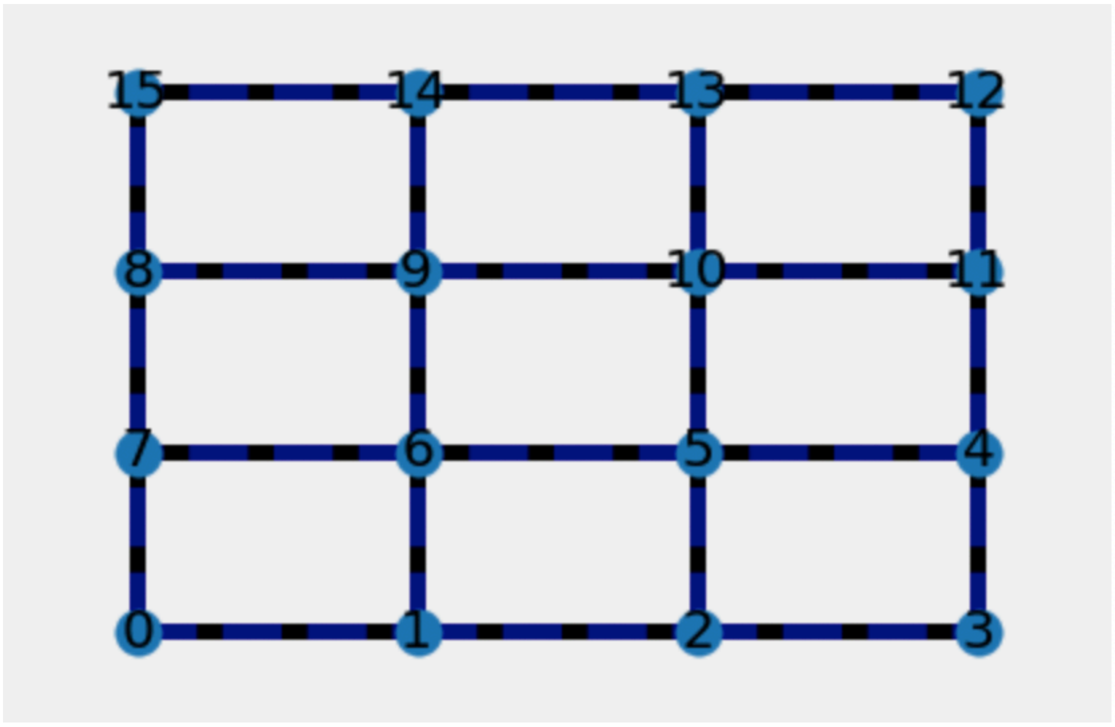
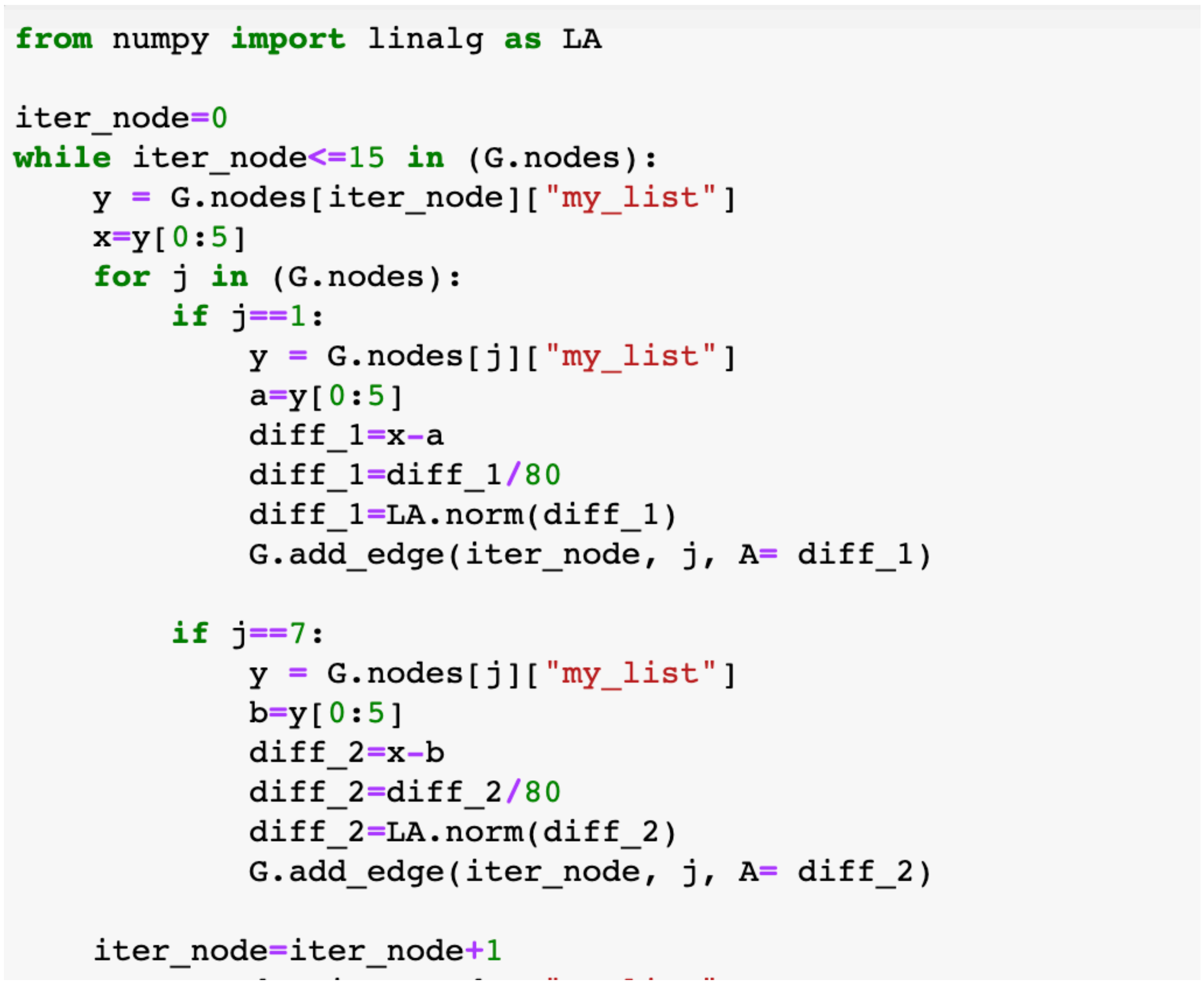


Figure 13. Manhattan Grid Mobility Model for our project

Here in Figure 13. We see that we have 16 nodes, and each node is connected to its neighboring nodes with an edge through some edge weights. Edge weights in a graph are numerical values assigned to the edges that indicate the cost or distance associated with that edge. For example, in a transportation network, edge weights could represent the distance between two locations, or the time required to travel between them. In a social network, edge weights could represent the strength of the relationship between two people, such as the number of mutual friends they have or the frequency of their communication. Edge weights are used in various graph algorithms such as shortest path algorithms, minimum spanning tree algorithms, and network flow algorithms.

So here in our case edge weights represent the demand shift between two nodes. So, we have calculated edge weights by taking the difference between the quantity of demands on one particular node and the quantity of demand on its neighboring nodes. The Figure 14. Below shows the code snippet of how we calculated edge weights for the first node i.e., Node_0;



```
from numpy import linalg as LA

iter_node=0
while iter_node<=15 in (G.nodes):
    y = G.nodes[iter_node]["my_list"]
    x=y[0:5]
    for j in (G.nodes):
        if j==1:
            y = G.nodes[j]["my_list"]
            a=y[0:5]
            diff_1=x-a
            diff_1=diff_1/80
            diff_1=LA.norm(diff_1)
            G.add_edge(iter_node, j, A= diff_1)

        if j==7:
            y = G.nodes[j]["my_list"]
            b=y[0:5]
            diff_2=x-b
            diff_2=diff_2/80
            diff_2=LA.norm(diff_2)
            G.add_edge(iter_node, j, A= diff_2)

    iter_node=iter_node+1
```

Figure 14. Calculation of Edge Weights at Node-0

Until now we have generated our data. We have feature matrix ‘X’ i.e., time instants and we also have our label vectors ‘y’ i.e., quantity of demand shifts from one node to the other one. We also have generated our Manhattan based Mobility Model by setting all the required parameters i.e., Edge Weights, No of Nodes, how the demand shift.

So, in summary we have generated all the required data for our Network Federated Learning model. Now next task is to perform learning i.e., Federated Learning (FL) and Network Federated Learning (NFL).

In following chapters, we will see how we have performed learning based on Generalized Total Variation (GTV) minimization problem for NFL

4. Problem Formulation

We have represented networked data through an undirected weighted empirical graph G = (V, E) which is a useful approach. For ease of notation, the nodes of the graph are identified by natural numbers, $V = \{1,...,|V|\}$. Each node $i \in V$ has its own local dataset X. It can be helpful to view X as a dataset with labels 'y' represented by (2) below.

$$X^{i} = \{(x^{(i,1)}, y^{(i,1)}),...,(x^{(i,m_i)}, x^{(i,m_i)})\}, \qquad i \in [0,15] \qquad (2)$$

In this scenario, $x^{(r)}$ and $y^{(r)}$ represent the characteristics and correct classification of the r-th piece of data in the localized dataset $X^{(\mathrm{i})}$. It's important to note that the number of data points in each local dataset, represented by m_i, may differ between nodes i.e., $i \in V$. Figure 7. provides a visual representation of an observed graph consisting of n nodes, each possessing their own localized dataset.

It is important to note that the NFL method described is not limited to local datasets that follow the format described in the above equation (2). The algorithm and its analysis, only require access to X indirectly, through the evaluation of a local loss function $L_i\ (v)$. This function determines how well a model with parameters v fits the local dataset $X^{(i)}$.

There are two advantages of our method of accessing data, which involves evaluating local loss functions. Firstly, our approach is privacy conscious. This is because it produces NFL techniques that do not share raw data, but instead only reveal information about the local loss functions. Specifically, this includes their gradients at the present selections for local model parameters. This information is generally derived from the averages of data points, thereby disclosing minimal data about individual points (as long as the sample size is sufficiently large).

The data access model we designed can also be used to accommodate scenarios where local datasets can only be accessed by a limited number of nodes, typically denoted as a small subset $M = \{i_1,...,i_m\} \subseteq V$. This is particularly applicable to wireless sensor networks, where the devices are powered by batteries and require wireless communication for data exchange [33]. In such cases, nodes that are unable to access their local datasets can be handled by incorporating a simple loss function, denoted as L_i(v) = 0.

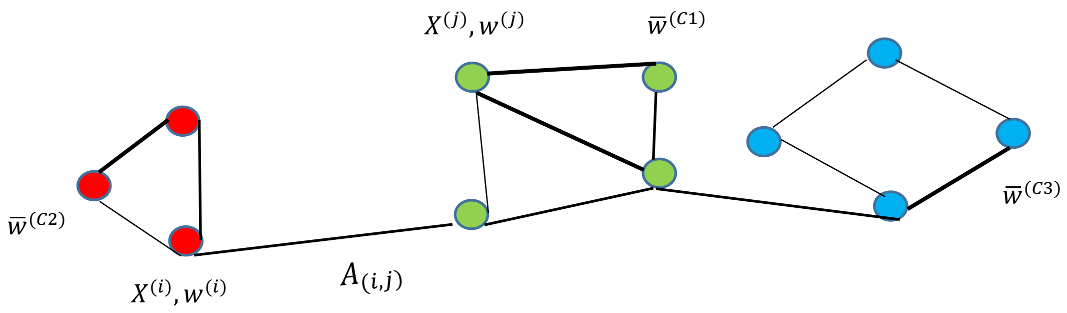


Figure 15. Undirected Empirical Graph for Network Federated Learning

The information presented in Figure 15. shows an example how networked data and corresponding models are represented using an undirected empirical graph G [21]. This graph consists of nodes (i.e., individual data sets) that carry local datasets $X^{(i)}$ and model weights $w^{(i)}$. Each node is scored using a local loss function L_i, which encapsulates the local dataset. Nodes are connected by weighted edges if they carry datasets with similar statistical properties. The edge weight $A_{i,i'}$ indicates the degree of similarity between the two nodes. The method relies on a clustering assumption where optimal parameter vectors for nodes in the same cluster C must be almost identical. The empirical graph is partitioned into three disjoint clusters, $C^{(1)}$, $C^{(2)}$, and $C^{(3)}$, based on the network structure and local datasets. It's important to note that the FL method used does not require knowledge of the partition, but instead learns it based on the data and graph structure.

Here we describe the use of undirected edges to represent the similarity between local datasets in a graph. The presence of an edge {i, i'} in the graph indicates that the datasets at nodes i and i' have similar statistical properties. The strength of this similarity is quantified by the edge weight $A_{i,i'}$, which is a positive value. To simplify notation, the weight of an edge can be denoted as $A_e = A_{i,i'}$. If there is no edge between nodes i and i', then the weight is zero, meaning that the datasets are not similar.

The undirected edges in the graph represent a symmetric relationship between local datasets. This means that if the dataset at node i is similar to the dataset at node i', then the similarity is also true in the reverse direction. The symmetric nature of the relationships between the datasets is also reflected in the edge weights, which are symmetric for each edge. So, the use of undirected edges in this graph provides a clear representation of the similarity between local datasets and their symmetric relationships.

$$A_{i,\, i'} \;=\; A_{i',i} \; for\; any\; two\; nodes\; i, i' \;\in\; V$$

Although the empirical graph G represents similarity in a symmetric way, the proposed NFL algorithm will be easier to formulate and analyze if the edges are oriented. To do this, the head and tail of an undirected edge $e \;=\; \{i, i\}$ are defined as $e+ \;=\; min\{i, i'\}$ and $e- \;=\; max\{i,\, i'\}$, respectively. By applying this definition to all edges in G, a set of directed edges for the empirical graph is obtained (3).

$$\overrightarrow{\mathcal{E}} \;= \{(i, i'): i, i' \in V, i < i' \; and \; \{i, i'\} \mathcal{E}\} \qquad (3)$$

We sometimes use the symbol $\mathcal{E}$ to represent both the set of undirected edges and the set of directed edges in the empirical graph G, even though they are different sets.

An empirical graph G has two vector spaces that are commonly linked to it.

a) One is called the "node space" w and is comprised of mappings or functions.

$$w \,:\, V \;\rightarrow\; \mathbb{R}^n \,:\, i \mapsto\; w^{(i)}$$

which assign node $i \in V$ a vector node $w^{(i)} \in \mathbb{R}^n$

b) Similarly, each edge e in the graph is assigned a vector $u^{(e)} \; in \; \mathbb{R}^n$ through a set of mappings $u \,:\, E \;\rightarrow\; R^n$. These two sets of vectors are connected through the block-incidence matrix (4).

$$D \,:\, W \;\rightarrow\; U \,:\, w \;\mapsto\; u \, with \, u^{(e)} \;=\; w^{(e+)} \;-\; w^{(e-)} \qquad (4)$$

4.1 NETWORKED MODELS

A networked model is composed of individual local models, one for each local dataset $X^{(i)}$. Our approach to NFL provides flexibility in designing these local models, as long as they are parametrized by a common finite-dimensional Euclidean space $\mathbb{R}^d$. This includes popular ML models such as (regularized) generalized linear models or linear time series models [34], [35]. The networked models are parametrized by a map w ∈ W that assigns each node $i \;\in\; V$ in the empirical graph G a local model parameter vector $w^{(i)} \in \mathbb{R}^d$. This can be expressed as a function:

$$w: V \rightarrow \mathbb{R}^n, i \mapsto w^{(i)}$$

where for each node i, $w^{(\mathrm{i})}$ is the local model parameter vector.

We evaluate the effectiveness of a specific selection of the local model parameters, $w^{(\mathrm{i})}$, using a local loss function, $L_i(w^{(i)})$. Typically, we assume that the local loss functions are both convex and differentiable. The NFL method presented, permits various options for the local loss functions.

The primary requirement for the choice of the local loss function $L_i(w^{(i)})$, from a computational standpoint, is that it enables efficient resolution of the regularized problem expressed in (5).

$$min_{w' \in \mathbb{R}^d} \ L_i(w') + \lambda \, \|w' - w''\|_2^2 \qquad (5)$$

The efficiency of our approach is determined by how quickly we can solve the above equation for any positive value of λ and any vector w'' in d-dimensional space. Solving above equation (5) is the same as evaluating the proximity operator $prox_{\mathrm{L}_i(\cdot),2\lambda}(w'')$. The proximity operator is a mathematical function that maps a vector to its nearest vector in a set, and it is used to find the solution to certain optimization problems. Therefore, the speed of our approach depends on our ability to efficiently evaluate this proximity operator for different values of λ and w''.

The NFL method described is specifically designed to be used with parametric models that can be trained by minimizing a loss function, denoted as $L_i(\cdot)$, where the proximity operator can be evaluated efficiently [36]. It is important to note that the shape of the loss function is typically determined by both the choice of the local model used and the metric used to measure prediction errors. In other words, the choice of model and metric used to evaluate the model's performance can have a significant impact on the shape of the loss function. This means that different local models and metrics may require different loss functions, and that it is important to carefully choose these components in order to achieve the best possible results.

The main focus of the project is on ML applications where the local loss functions $L_i\ (w^{(\mathrm{i})})$ do not have enough statistical power to guide the learning of model parameters $w^{(\mathrm{i})}$. This is particularly true in cases where the local dataset consists of feature vectors with a large number of dimensions, but the number of training examples is relatively small. The goal in such cases is to learn the parameter vector of a linear hypothesis, where the hypothesis is defined as the dot product of the feature vector and the parameter vector, $h(x) = (x^T . w^{(i)})$.

The common method used for learning the parameter vector is linear regression, which involves minimizing the average squared error loss function (6).

$$L_i(w^{(i)}) = (1/m_i) \sum_{r=1}^{m_i} (y^{(r)} - (w^{(i)})^T x^{(r)})^2 \qquad (6)$$

However, in the high-dimensional regime where the number of features is much larger than the number of training examples, the minimum is not unique, and a poor hypothesis may be generated, leading to large prediction errors on data points outside of the training set.

To overcome this issue, regularization methods such as ridge regression or Lasso can be used to add a penalty term to the loss function [37], which helps to constrain the values of the parameter vector and reduce overfitting. By doing so, the learning algorithm can produce a more robust and accurate hypothesis that can better generalize to new data points outside of the training set.

The central idea of the project is to use the empirical graph G to regulate the learning of local model parameters. This regulation involves limiting the variation of local model parameters over edges with large weights. We introduce the concept of (GTV) minimization, which measures the variation of local parameter vectors, and proposes that a small GTV can be achieved by minimizing the smoothness assumption used in semi-supervised learning [38].

GTV in more detail described as a quantitative measure for the variation of local parameter vectors. The smoothness assumption used in semi-supervised learning requires that GTV be small. The analysis relates the smoothness assumption underlying GTV minimization to a clustering assumption. This clustering assumption requires that local model parameters be constant over subsets or clusters of nodes in the empirical graph.

The clustering of local model parameters is essentially an adaptive pooling of local datasets, where the pooling is driven jointly by the connectivity of the empirical graph G and the shape of local loss functions.

4.2 Clustering Assumption

We talk about the scenario where there is a networked data system, represented by a graph G with nodes and edges (V, E). Each node in the graph, denoted by i, has a local dataset $X^{(\mathrm{i})}$ and a local model with parameters $w^{(\mathrm{i})}$. The objective is to learn the local model parameters $w^{(\mathrm{i})}$ for every node i in the graph.

The crucial assumption of clustered Federated Learning is that the local datasets in the graph form clusters, where the datasets in the same cluster have similar statistical properties [27].

Now, if there is a cluster C of nodes, it is natural to combine their local datasets or add their local functions to learn a cluster-specific parameter vector. In other words, instead of training individual models for each node, the models' parameters for the nodes in the same cluster are combined to form a single parameter vector that represents the cluster (7).

$$\overline{w}^{(C)} = \underbrace{argmin}_{v\in\mathbb{R}^n} f^C(v) \; with \; f^C(v) := \sum_{i\in C} L_i\,(v) \qquad (7)$$

Overall, this describes the concept of clustered Federated Learning, where the local datasets in a networked data system are assumed to form clusters, and the local models' parameters in each cluster are combined to form a single parameter vector representing the cluster. This approach enables efficient training of models on decentralized data systems.

Equation (7) cannot be applied in real-life scenarios as we often lack knowledge about the cluster C. The primary analytical contribution is providing an upper bound that measures the difference between solutions obtained from GTV minimization and cluster-based learning problem defined by the equation (7) above, which is not practical to implement. This bound helps to identify the statistical properties of FL algorithms that are derived by using optimization techniques for solving GTV minimization. The equation (7) has a solution called $w^{(\mathrm{C})}$ which has the property of minimizing the sum of all the local loss functions that belong to the same cluster C, which is a subset of V. This means that $w^{(C)}$ is the best model parameter for a training set created by pooling all the local datasets that belong to cluster C. It is assumed that the solution to this equation is unique, as indicated by the notation. Assumption below ensures the uniqueness of the solution in above equation (7).

To clarify our assumption that datasets within the same cluster share similar statistical properties, we need to establish a specific requirement. This requirement states that nodes $i \in C$ within the same cluster C must have nearby minimizers for their respective local loss functions. This means that there should be a small deviation $\left\| v^{(i)} - \overline{w}^{(C)} \right\|$ between the minimizer $v^{(\mathrm{i})}$ of $L_{\mathrm{i}}\ (\cdot)$ and the corresponding optimal parameter vector for the entire cluster. For loss functions that are differentiable and convex, this requirement is equivalent to having a small gradient of the local loss functions at the cluster-wise minimizer.

To express this requirement in a more convenient way for analysis, we will set an upper bound for the dual norm $\left\|\nabla L_{\mathrm{i}}(\ w^{(C)})\right\|_*$ of the local loss gradient.

4.2.1 Assumption 1 (Clustering)

Consider a networked data represented by an empirical graph G with nodes that carry local loss functions, denoted as $L_i(v)$, where i is an element of the set of nodes V. The nodes are partitioned into disjoint clusters, denoted as $P = \{C^{(1)}, \ldots, C^{(k)}\}$, with each cluster denoted as $C^{(\mathrm{c})}$ and satisfying the condition that $C^{(\mathrm{c})} \cup C^{(\mathrm{c}\prime)}$ is empty for $c \neq c'$ and V is equal to the union of all clusters i.e., ($V\ =\ C^{(1)} \cup \ldots \cup C^{(k)}$).
Additionally, for each cluster $C^{(\mathrm{c})}$, there exists a solution $w^{(\mathrm{c})}$ in $\mathbb{R}^{\mathrm{d}}$ obtained through cluster-wise minimization, which satisfies the inequality that the norm of the gradient of the local loss function ∇L_i with respect to $w^{(\mathrm{c})}$ is less than or equal to a constant value $\delta^{(\mathrm{i})}$ for all nodes i in $C^{(\mathrm{c})}$ (8).

$$\left\| \nabla L_i\, \overline{w}^{(c)} \right\|_* \leq\ \delta^{(i)}\ for\ all\ i \in C^{(c)} \qquad (8)$$

This clustering assumption ensures that the norm of the gradient of the local loss function with respect to $w^{(\mathrm{c})}$ is bounded by a constant value $\delta^{(\mathrm{i})}$ for each node i in the empirical graph. This norm represents the difference between the cluster-wise minimizer $w^{(\mathrm{c})}$ and the minimizers of the local loss functions $\nabla L_i\, \overline{w}^{(c)}$ for each node i in $C^{(\mathrm{c})}$.

We use cluster-wise minimization as a tool to analyze the NFL methods. However, note that this approach is not practical as it assumes knowledge of the clusters in the partition, which is unrealistic under Assumption 1. Instead, we formulate NFL as a GTV minimization, which enforces the "clusteredness" of local model parameters by requiring a small variation across edges in the empirical graph. This regularization strategy works best when many edges connect nodes in the same cluster, but only a few connect nodes in different clusters.

We present a precise condition on the network structure for GTV minimization to capture the true underlying cluster structure of the local loss functions. However, the analysis of the NFL method requires the local loss functions to be convex and smooth, and their (partial) sums in the cluster-wise objective function to be strongly convex. This is an important requirement for the success of GTV minimization in capturing the true underlying cluster structure of the local loss functions.

4.2.2 Assumption 2 (Convexity and Smoothness)

In Assumption 2, which is a standard assumption in Federated Learning (FL) literature, states that for each node i in the set of nodes V ($i \in V$), the local loss function $L_i(w^{(i)})$ is convex and differentiable, with a gradient that satisfies a certain condition expressed in (9).

$$\|\nabla L_i\,(v\,') \,-\, \nabla L_i\,(v)\|_* \,\leq\, \beta^{(i)}\,\|v\,' - v\| \qquad (9)$$

The condition is given by equation (9) which bounds the norm of the difference between the gradients of L_i at two different points, v and v'. We introduce the objective function $f^{(c)}$ for each cluster $C^{(c)}$ in a partition P of the nodes. The objective function is strongly convex.

$$f^{(c)}(v\,') \,\geq\, f^{\,(c)}(v) + (v' - v)^T \partial f^{(c)}(v) \,+\, \left(\alpha^{(c)}/_2\right) \|v\,' - v\,\|^2 \; for\ any\ v\,', v \in\, \mathbb{R}^d \qquad (10)$$

as defined in above Equation (10), which states that the value of $f^{(c)}$ at any point v' is greater than or equal to its value at point v plus a quadratic term that depends on the difference between v' and v. Here $\alpha^{(c)}$ is a positive constant that can vary across different clusters.

Assumption 2 is satisfied by many important ML models, but not by some deep learning models that result in non-convex loss functions [39]. However, we expect their theoretical analysis to provide insight into settings where Assumption 2 is violated. Assumption 2 only requires the cluster-wise sums of local loss functions to be strongly convex and allows for trivial local loss functions that may be constant due to privacy or computational constraints. The NFL method described can handle non-informative local loss functions by leveraging the similarities between local datasets as represented by the edges in the empirical graph G.

5. Generalized Total Variation (GTV) Minimization

We will describe a way to measure the variation of local model parameters, represented by w, using a quantitative measure called the GTV. The Assumption 1 in clustering suggests that we should learn the model parameters, $w^{(\mathrm{i})}$ by optimizing them using a cluster-wise approach. Specifically, for each cluster $C^{(\mathrm{c})}$ we use the solution of the optimization (7) as the local model parameters for all the nodes i that belong to $C^{(\mathrm{c})}$. However, implementing this approach is not practical since we don't usually know the partition of clusters directly. Therefore, we use the empirical graph G to penalize variations in the local model parameters, $w^{(\mathrm{i})}$, over nodes that are well-connected.

We explain a method of penalizing variation in a graph, in order to encourage local model parameters to remain constant within clusters of nodes. The idea is that densely connected nodes within a cluster should have similar parameters. The success of this method depends on the clustering of the nodes in the empirical graph. The assumption is that nodes within the same cluster should have many connections between them, while nodes in different clusters should have only a few connections. Using this method, called GTV, as a regularizer will help in learning the local model parameters.

We discuss the relationship between the cluster structure of a graph G and the variation of local parameter vectors $w^{(\mathrm{i})}$. It suggests that if G has a high density of edges within clusters and few boundary edges between them, then it is reasonable to expect a small variation of local parameter vectors across edges. To measure this variation, we introduce a function u that maps each edge e in the graph to the difference between the local parameter vector $w^{(\mathrm{e}+)}$ at the endpoint where the edge points to and the local parameter vector $w^{(\mathrm{e}-)}$ at the endpoint where the edge comes from i.e., $u : e \in \mathcal{E} \longmapsto u^{(e)} := w^{(e+)} - w^{(e-)}$
Using the block-incidence matrix, the variation of w can be expressed more compactly as
$u = Dw.$

The GTV is calculated by summing up the values of a convex penalty function φ that measures the difference between the values of w at neighboring points (i and i'). The sum is taken over all pairs of neighboring points in a set of edges E (11).

$$\|w\|_{GTV} := \sum_{\{i,i'\} \in \mathcal{E}} A_{i,i'}\, \varphi\left(w^{(i')} - w^{(i)}\right) \qquad (11)$$

In addition to the overall GTV measure, we also introduce a subset measure, where the sum is only taken over a particular subset of edges S. This subset measure is denoted by $\|w\|_S$ and it provides a more targeted measure of the variation of w within a specific region of the model (12).

$$\|w\|_{S} := \sum_{\{i,i'\} \in S} A_{i,i'}\, \varphi\left(w^{(i')} - w^{(i)}\right) \qquad (12)$$

Overall, we present a mathematical framework for measuring the variation of local model parameters w, which can be useful in various applications, such as image processing and machine learning.

To provide further clarification, the GTV (11) refers to a technique that measures the variation in a set of data. The technique relies on a penalty function $\varphi(v) \in \mathbb{R}$ that determines the various measures of variation. The choice of penalty function is a crucial decision as it affects both the computational and statistical properties of the GTV minimization problem. The penalty function is assumed to be convex, and there are different options to choose from, including:

A) $\varphi(v) = \|v\|_2$, which is used in nLasso,
B) $\varphi(v) = (1/2)\|v\|_2^2$ used in MOCHA,
C) $\varphi(v) = \|v\|_1$, used by another recent FL method for networked data.

Each penalty function choice offers a different balance between computational complexity and statistical properties of the resulting FL algorithms. For instance, the penalty $\varphi(v) = \|v\|_2$ used in nLasso is more computationally challenging compared to the penalty $\varphi(v) = (1/2)\|v\|_2^2$ used in MOCHA.

However, MOCHA is more accurate in learning models for data with specific network structures such as chains that are challenging for GTV minimization methods using the smooth penalty [40].

$$\varphi(v) = (1/2)\|v\|_2^2 \qquad (13)$$

Overall, the choice of the penalty function in GTV (14) affects the computational and statistical properties of FL algorithms, and different penalty function choices offer different trade-offs between these properties.

The process of GTV minimization involves finding the local model parameters, $w^{(i)}$, by balancing the sum of local loss functions and the GTV. This is achieved by solving a minimization problem, where the empirical risk incurred by the local model parameters is measured by the sum of the local loss functions (14). The goal is to learn parameter vectors, $w^{(i)}$, with a small GTV while incurring a small local loss. The regularization parameter, λ, controls the preference for learning parameter vectors with small GTV versus incurring small local loss. Cross-validation or analysis of the solutions can be used to determine the optimal value for λ. Increasing the value of λ leads to increasingly clustered solutions, where the local model parameters become constant over larger subsets of nodes. If λ is larger than a critical value, $\widehat{w}^{(i)}$ becomes constant over all nodes $i \in V$. The solutions of GTV minimization capture the underlying partition of the network if certain conditions on the local loss functions and the empirical graph are met. GTV minimization unifies and extends some well-known methods for distributed optimization and learning.

$$\widehat{w} \in \underbrace{arg\ min}_{w \in W} \textstyle\sum_{i \in V} L_i\left(w^{(i)}\right) + \lambda \|w\|_{GTV} \qquad (14)$$

In our project, we consider the case of Network MOCHA, because it is more accurate in learning models for data with specific network structures such as chains that are challenging for GTV minimization methods using the smooth penalty. Here objective function is smooth and convex, so it can be solved iteratively using Gradient Descent steps.

We could learn local weight vector by solving local linear regression for each node (15),

$$\underbrace{min}_{w^{(i)} \in \mathbb{R}^n} \left({}^{1}/_{m_i}\right)\left\|y^{(i)} - X^{(i)}w^{(i)}\right\|_2^2 \qquad (15)$$

Where X being the feature matrix and y is the label vector and m_i is the sample size.

In any learning problem, the feature matrix, label vectors, and sample size are important concepts.

A) Feature matrix:

The feature matrix, also known as the design matrix, is a two-dimensional matrix that contains the input features of the data. Each row of the matrix represents a sample or observation, and each column represents a feature [41].

B) Label vector:

The label vector, also known as the target vector or response variable, is a one-dimensional vector that contains the output values we are trying to predict. Each element of the vector corresponds to a particular sample or observation in the feature matrix [42].

C) Sample size:

The sample size is the number of observations in the dataset. It represents the number of samples or instances we have available to train our model. A larger sample size generally leads to more accurate and reliable models [43].

The feature matrix, label vector, and sample size are important components of any machine learning problem, and they play a crucial role in training and evaluating ML models.

Then we have the total variation term which enforces weights to be nearly constant over clusters. i.e.,

$$\sum_{\{i,j\}} A_{i,j} \left\| w^{(i)} - w^{(i)} \right\|^2$$

Now we have the objective function which is represented by (16) below:

$$\underbrace{min}_{w^0, w^1, \ldots\ldots, w^{m-1}} \quad \Sigma_i \left({}^1/_{m_i} \right) \left\| y^{(i)} - X^{(i)} w^{(i)} \right\|_2^2 + \lambda_{GTV} \, \Sigma_{\{i,j\}} A_{i,j} \left\| w^{(i)} - w^{(i)} \right\|^2 \qquad (16)$$

Here the first term represents the local training error, and the second term is responsible for clusterdness. The more we increase the value of λ_{GTV} the more we are introducing the clusterdness.

Now since the objective function is smooth and convex, it can be solved iteratively using Gradient Descent steps represented by (17) below.

$$w^{(i,k+1)} = w^{(i,k)} - \propto \left(-\left(2/m_i\right)\left(X^{(i)}\right)^T\left(y^{(i)} - X^{(i)}w^{(i,k)}\right) + 2\lambda_{GTV}\sum_{i'\neq i} A_{i,i'}\left(w^{(i,k)} - w^{(i',k)}\right)\right) \quad (17)$$

The above formula represents the weight update rule for a network graph structure in a NFL algorithm. The detailed explanation of each component of the formula:

$w^{(i,k+1)}$:

This represents the updated weight value of node i in the k+1 iteration. It is calculated based on the previous weight value ($w^{(i,k)}$) and the gradient descent term.

$w^{(i,k)}$:

This represents the weight value of node i in the kth iteration. It is the starting point for the weight update rule.

$\propto$:

This is the learning rate or step size, which controls the magnitude of the weight update. A small learning rate leads to slow convergence but avoids overshooting, while a large learning rate leads to faster convergence but may overshoot the minimum.

$-\propto \left(-\left(2/m_i\right)\left(X^{(i)}\right)^T\left(y^{(i)} - X^{(i)}w^{(i,k)}\right)\right.$:

This is the gradient descent term that computes the negative gradient of the cost function with respect to the weight vector w. Here, $X^{(i)}$ is the feature matrix for node i, $y^{(i)}$ is the corresponding output vector, m_i is the number of training samples for node i, and $\left(X^{(i)}\right)^T$ denotes the transpose of $X^{(i)}$. The gradient descent term represents the direction of steepest descent in the cost function, which is used to update the weight values.

$2\lambda_{GTV}\sum_{i'\neq i} A_{i,i'}\left(w^{(i,k)} - w^{(i',k)}\right)$:

This is the regularization term that encourages smoothness in the weight vector across adjacent nodes in the graph structure. λ_{GTV} is the regularization parameter, $A_{i,i'}$ is the adjacency matrix element between nodes i and i', and $\sum_{i'\neq i}(.)$ represents the sum over all neighboring nodes of

i. The regularization term penalizes large changes in the weight vector between neighboring nodes, which helps to prevent overfitting.

The weight update rule combines the gradient descent term and the regularization term to update the weight values for each node in the network graph structure. The gradient descent term drives the weight values towards the minimum of the cost function, while the regularization term encourages smoothness in the weight vector across adjacent nodes. The learning rate controls the speed of convergence, and the regularization parameter controls the strength of the regularization term.

Algorithm_4 (GTV Minimization for Empirical Graph)

```
1.  Define a network graph.
2.  Init weight to zero at all nodes.
3.  Implement mobility model to model the movement of vehicles
4.  Determine the edge weights ('A')
5.  Plot the Manhattan Grid Mobility Model
6.  samplesize is the number of observations in the dataset
7.  N_GD is the number of GD steps
8.  lambda_TV is the regularization parameter
9.  learning_rate controls the magnitude of the weight update.
10. X is the feature matrix
11. y is the label vector
12. for iter_node in G.nodes do
13.     G.nodes[iter_node]["w"] = np.zeros(X.shape[1])
14. end for
15. for i in N_GD do
16.     for node_i ∈ G do
17.         tmp = -(2/samplesize)*(X.T.dot(y - X.dot(G.nodes[node_i]["w"])))
18.         for node_j in G
19.             edge_weight = float(G.edges[(node_i,node_j)]["A"])
20.             tmp = tmp + 2*lambda_TV*edge_weight*(G.nodes[node_i]["w"]-
                G.nodes[node_j]["w"] )
21.             G.nodes[node_i]["g"] = tmp
```

```
22.         end for
23.     end for
24.     for node_i ∈ G do
25.         G.nodes[node_i]["w"]=G.nodes[node_i]["w"]-
            (learning_rate*G.nodes[node_i]["g"] )
26.     end for
27. end for
```

The Algorithm_4 above defines the implementation of GTV which shows the overall scenario of the thesis. The algorithm is a distributed implementation of a gradient descent optimization algorithm for a linear regression model with total variation (TV) regularization. We use a network graph to represent the distributed system and applies the mobility model to model the movement of vehicles in the system. The algorithm starts by initializing the weights at all nodes to zero.

Next, we use a mobility model to determine the edge weights 'A' between the nodes in the network graph. The edge weights represent the demand shift of vehicles between two nodes and are used to determine the amount of regularization applied in the optimization algorithm.

The we plot the Manhattan Grid Mobility Model, which is a particular type of mobility model that simulates vehicles moving in a Manhattan grid-like pattern.

We then set the parameters for the optimization algorithm, including the number of observations in the dataset (samplesize), the number of gradient descent steps (N_GD), the regularization parameter (lambda_TV), the learning rate for weight updates, the feature matrix (X), and the label vector (y).

The main optimization loop starts by initializing the weight vector at each node to a zero vector of the same length as the feature matrix X. Then, for each iteration of the gradient descent algorithm (i.e., for each iteration in the range of N_GD), the algorithm updates the weight vector at each node by computing the gradient of the loss function and applying a weight update using the learning rate.

To compute the gradient, the algorithm first computes the local gradient for each node by computing the gradient of the loss function using the current weight vector at that node. This is done using the standard gradient descent update rule, which involves taking the derivative of the loss function with respect to the weight vector and multiplying it by the learning rate. The local gradient is stored in a temporary variable called 'tmp'.

Next, for each neighboring node in the network graph, the algorithm computes an additional term for the gradient that incorporates the TV regularization term. This term encourages the weight vectors at neighboring nodes to be similar and is scaled by the edge weight between the two nodes. The updated gradient for the current node is the sum of the local gradient and the regularization term for all neighboring nodes.

Finally, the weight vector at each node is updated using the gradient computed in the previous step, and the learning rate. The updated weight vector is then used as the starting point for the next iteration of the gradient descent algorithm.

Overall, the algorithm uses a distributed approach to solve a linear regression problem with GTV regularization by partitioning the data and the model weights across a network of nodes and using a mobility model to simulate the movement of vehicles in the system. By incorporating TV regularization and using a distributed approach, the algorithm is able to find a solution that is both accurate and robust to noise and outliers in the data.

6. Validation and Results

In this project, we have worked on implementation of Network Federated Learning (NFL) and Federated Learning (FL) for real-time traffic prediction in Internet of Vehicles (IoVs) scenario. We took advantage of Graphical Model where each node represents a certain area, and each node is connected to its neighboring nodes through an edge with some edge weights. We considered the Manhattan grid-based environment for this project.

At first, we designed a mobility model for IoVs system which shows on how the vehicles are changing randomly from one area/node to the other area/node in a certain time instant featuring the downtown area/node (hotspot region). This will be our generated data set for training purpose. So, we have the model which resembles the real time random traffic variation from one area to the other area.

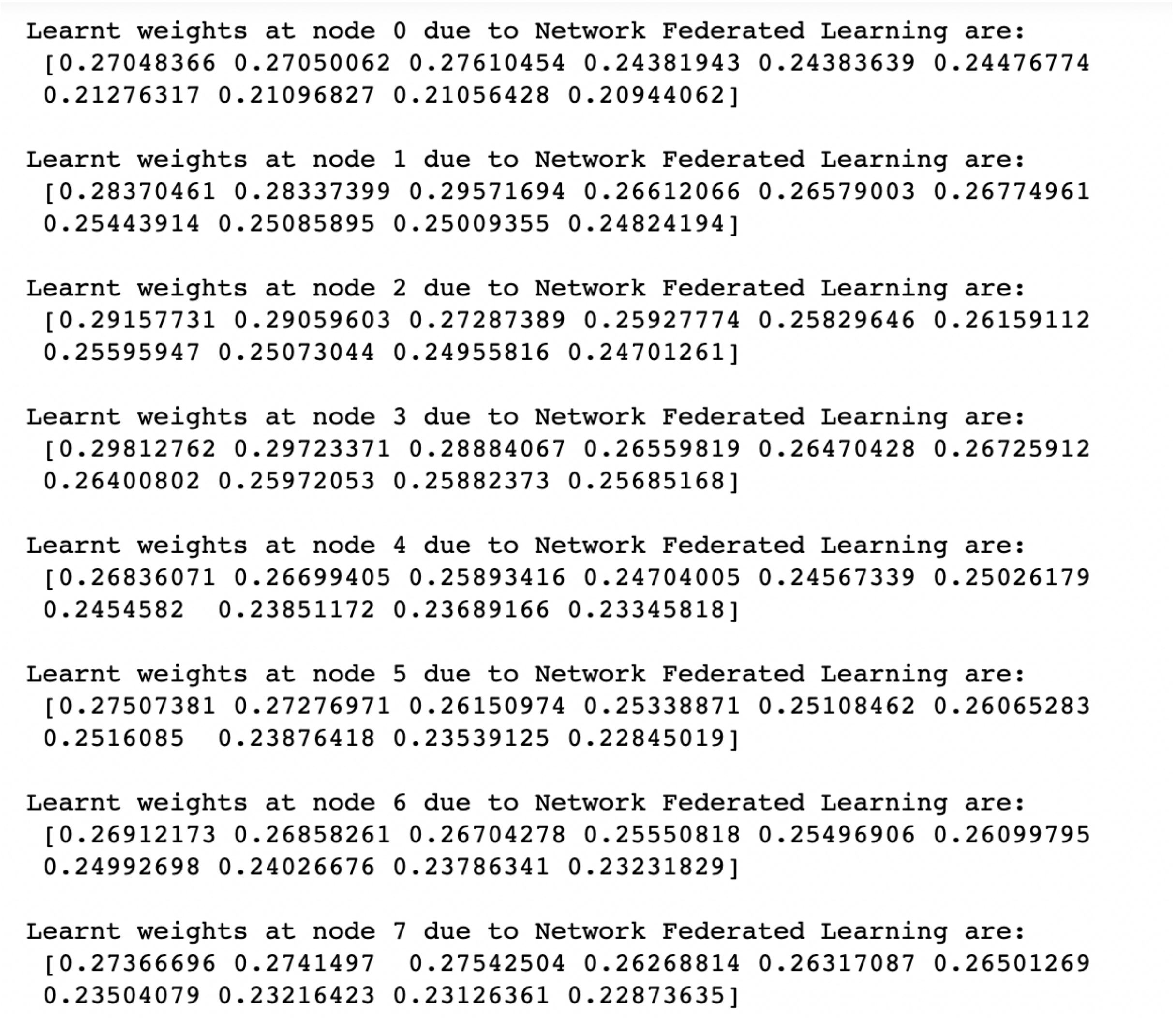
Then the implementation of NFL comes into play. Through Generalized Total Variation (GTV) Minimization method, we measure the clusterdness of weights, means the nodes close to each other doesn't vary too much so they can be considered into one cluster. The GTV parameter controls the clusterdness of the learnt hypothesis. Using small value of GTV parameter, we get the personalized hypothesis (Personalized Model) for each node. Then by increasing the value of GTV parameter results in Clustered Federated Learning which pools the dataset into well connected clusters of nodes. When the value of GTV parameter is sufficiently large enough, we get the Global Model/Fed Averaging, means we have only one hypothesis for the whole Network. So, here we learnt the weights for each node and by minimizing the loss through GTV method which acts as a regulariser. At the end what we get is the learnt weights at each node through GTV variation, means by setting the GTV parameter to introduce the clusterdness for NFL.

So, this was the training phase. Now to test the learnt data, we have created another mobility model that will be our dataset for testing (test data). Then we tested the predictions of NFL by calculating the Mean Absolute Error of trained weights with respect to the test data, (see Figure 19).

Then we performed Centralized Federated Learning to compare the results of NFL with respect to FL, see Figure 18(a, b, c, d). Here in Centralized FL training was performed at each node separately, then after training these nodes send the data to server side where server aggregate this data and send the aggregated data back to the nodes and this iteration continues. So, at the end what I get is the Global Model for the whole network.

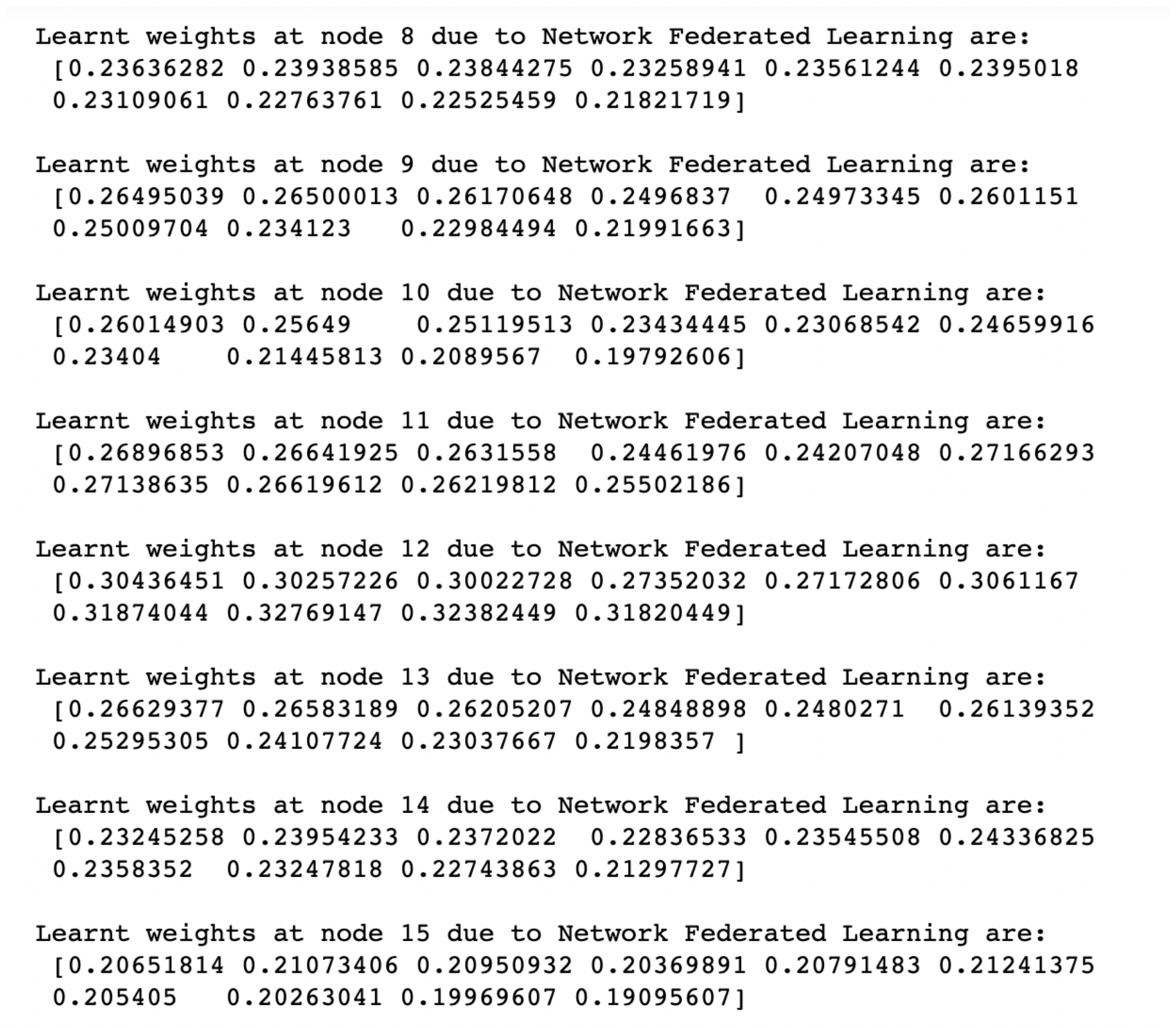
Then we have tested the predictions of FL by calculating the Mean Absolute Error (MAE) of trained weights with respect to the test data, (see Figure 19).

After comparing the MAE of NFL and FL, we observe that MAE due to NFL at each node is too low with respect to the MAE calculated due to FL. So, this means by introducing clusterdness we obtain better efficiency, plus we have better storage capacity at server side, plus we make our model more secure.



```
Learnt weights at node 0 due to Network Federated Learning are:
 [0.27048366 0.27050062 0.27610454 0.24381943 0.24383639 0.24476774
 0.21276317 0.21096827 0.21056428 0.20944062]

Learnt weights at node 1 due to Network Federated Learning are:
 [0.28370461 0.28337399 0.29571694 0.26612066 0.26579003 0.26774961
 0.25443914 0.25085895 0.25009355 0.24824194]

Learnt weights at node 2 due to Network Federated Learning are:
 [0.29157731 0.29059603 0.27287389 0.25927774 0.25829646 0.26159112
 0.25595947 0.25073044 0.24955816 0.24701261]

Learnt weights at node 3 due to Network Federated Learning are:
 [0.29812762 0.29723371 0.28884067 0.26559819 0.26470428 0.26725912
 0.26400802 0.25972053 0.25882373 0.25685168]

Learnt weights at node 4 due to Network Federated Learning are:
 [0.26836071 0.26699405 0.25893416 0.24704005 0.24567339 0.25026179
 0.2454582  0.23851172 0.23689166 0.23345818]

Learnt weights at node 5 due to Network Federated Learning are:
 [0.27507381 0.27276971 0.26150974 0.25338871 0.25108462 0.26065283
 0.2516085  0.23876418 0.23539125 0.22845019]

Learnt weights at node 6 due to Network Federated Learning are:
 [0.26912173 0.26858261 0.26704278 0.25550818 0.25496906 0.26099795
 0.24992698 0.24026676 0.23786341 0.23231829]

Learnt weights at node 7 due to Network Federated Learning are:
 [0.27366696 0.2741497  0.27542504 0.26268814 0.26317087 0.26501269
 0.23504079 0.23216423 0.23126361 0.22873635]
```

Figure 16a. Learnt Weights due to NFL

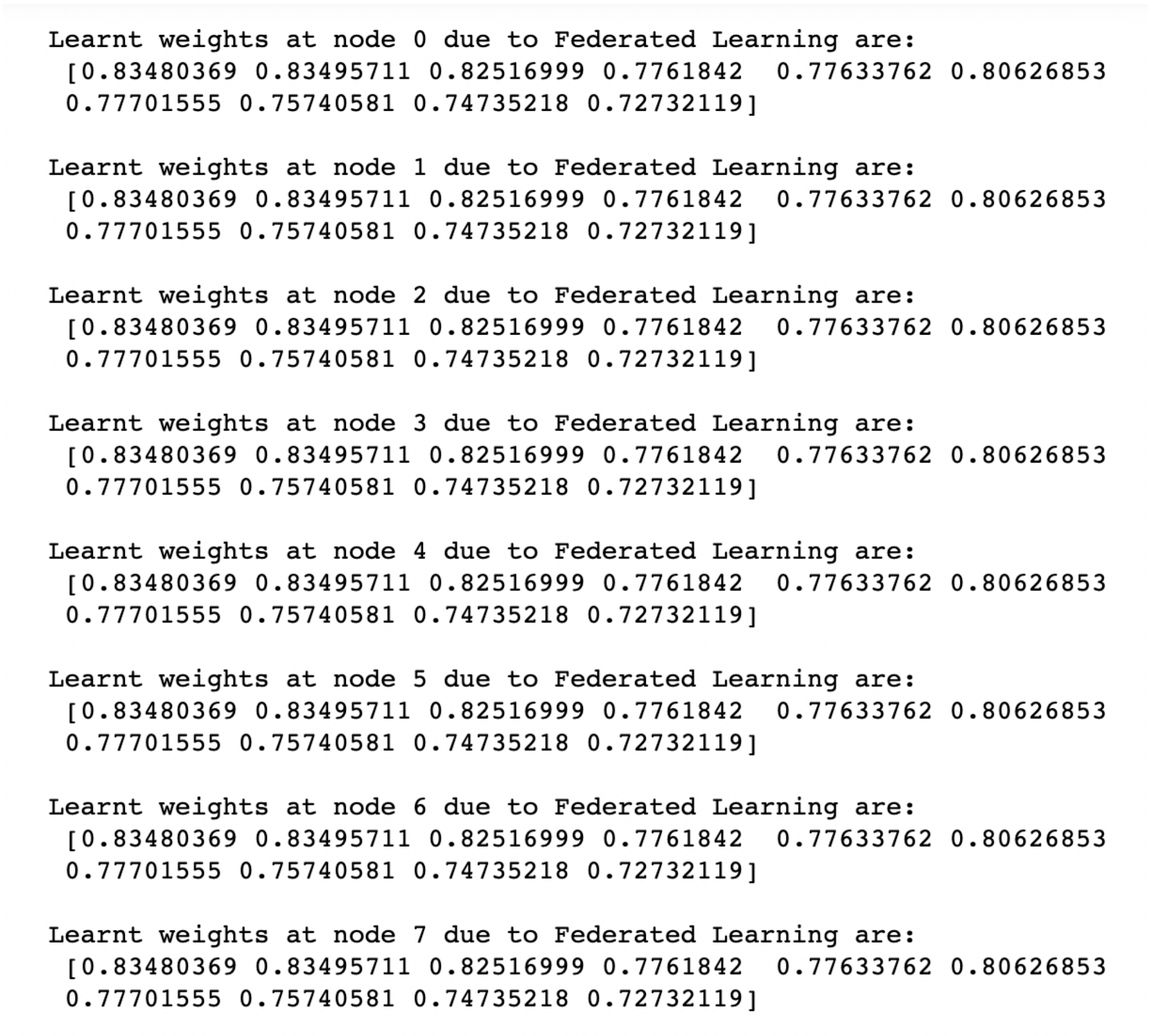


```
Learnt weights at node 8 due to Network Federated Learning are:
 [0.23636282 0.23938585 0.23844275 0.23258941 0.23561244 0.2395018
 0.23109061 0.22763761 0.22525459 0.21821719]

Learnt weights at node 9 due to Network Federated Learning are:
 [0.26495039 0.26500013 0.26170648 0.2496837  0.24973345 0.2601151
 0.25009704 0.234123   0.22984494 0.21991663]

Learnt weights at node 10 due to Network Federated Learning are:
 [0.26014903 0.25649    0.25119513 0.23434445 0.23068542 0.24659916
 0.23404    0.21445813 0.2089567  0.19792606]

Learnt weights at node 11 due to Network Federated Learning are:
 [0.26896853 0.26641925 0.2631558  0.24461976 0.24207048 0.27166293
 0.27138635 0.26619612 0.26219812 0.25502186]

Learnt weights at node 12 due to Network Federated Learning are:
 [0.30436451 0.30257226 0.30022728 0.27352032 0.27172806 0.3061167
 0.31874044 0.32769147 0.32382449 0.31820449]

Learnt weights at node 13 due to Network Federated Learning are:
 [0.26629377 0.26583189 0.26205207 0.24848898 0.2480271  0.26139352
 0.25295305 0.24107724 0.23037667 0.2198357 ]

Learnt weights at node 14 due to Network Federated Learning are:
 [0.23245258 0.23954233 0.2372022  0.22836533 0.23545508 0.24336825
 0.2358352  0.23247818 0.22743863 0.21297727]

Learnt weights at node 15 due to Network Federated Learning are:
 [0.20651814 0.21073406 0.20950932 0.20369891 0.20791483 0.21241375
 0.205405   0.20263041 0.19969607 0.19095607]
```

Figure 16b. Learnt Weights due to NFL

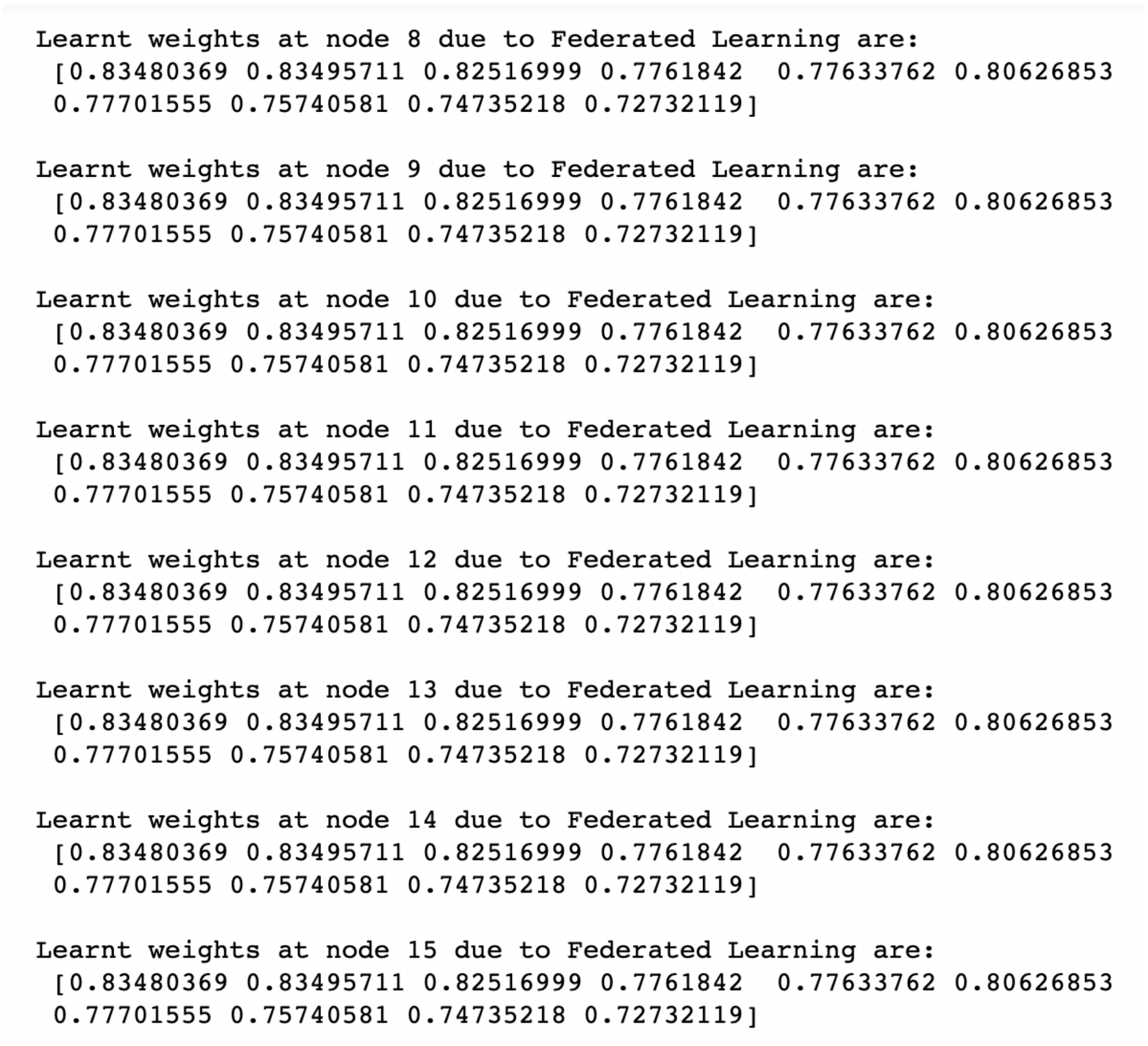
Figure 16a and 16b shows the learnt weight we obtained at each node for each time instant (in our case we take 10-time instants) through GTV method for NFL. Here we see that the learnt weights do not vary so much among the adjacent nodes, hence we introduce the concept of clusterdness. We have personalized model at each node, and we do not have one single global model for the whole network as in the case of FL we will see further.

To compare these results, we performed centralized FL at each node and we find the global model at for the whole network.



```
Learnt weights at node 0 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 1 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 2 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 3 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 4 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 5 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 6 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 7 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]
```

Figure 17a. Learnt Weights due to FL



```
Learnt weights at node 8 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 9 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 10 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 11 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 12 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 13 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 14 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]

Learnt weights at node 15 due to Federated Learning are:
 [0.83480369 0.83495711 0.82516999 0.7761842  0.77633762 0.80626853
 0.77701555 0.75740581 0.74735218 0.72732119]
```

Figure 17b. Learnt Weights due to FL

Figure 17a and 17b shows the learnt weight we obtained at each node for each time instant for FL case. Here we see that the learnt weights have single global model for each node at each time instant. Means we have one single global model for the whole network.

Now if we compare the learnt weights of NFL and the learnt weights of FL. We will see that the regression line of NFL converges more towards the original data set see Figure 18 (a, b, c, d). This means NFL is predicting the data at each node more efficiently with respect to FL.

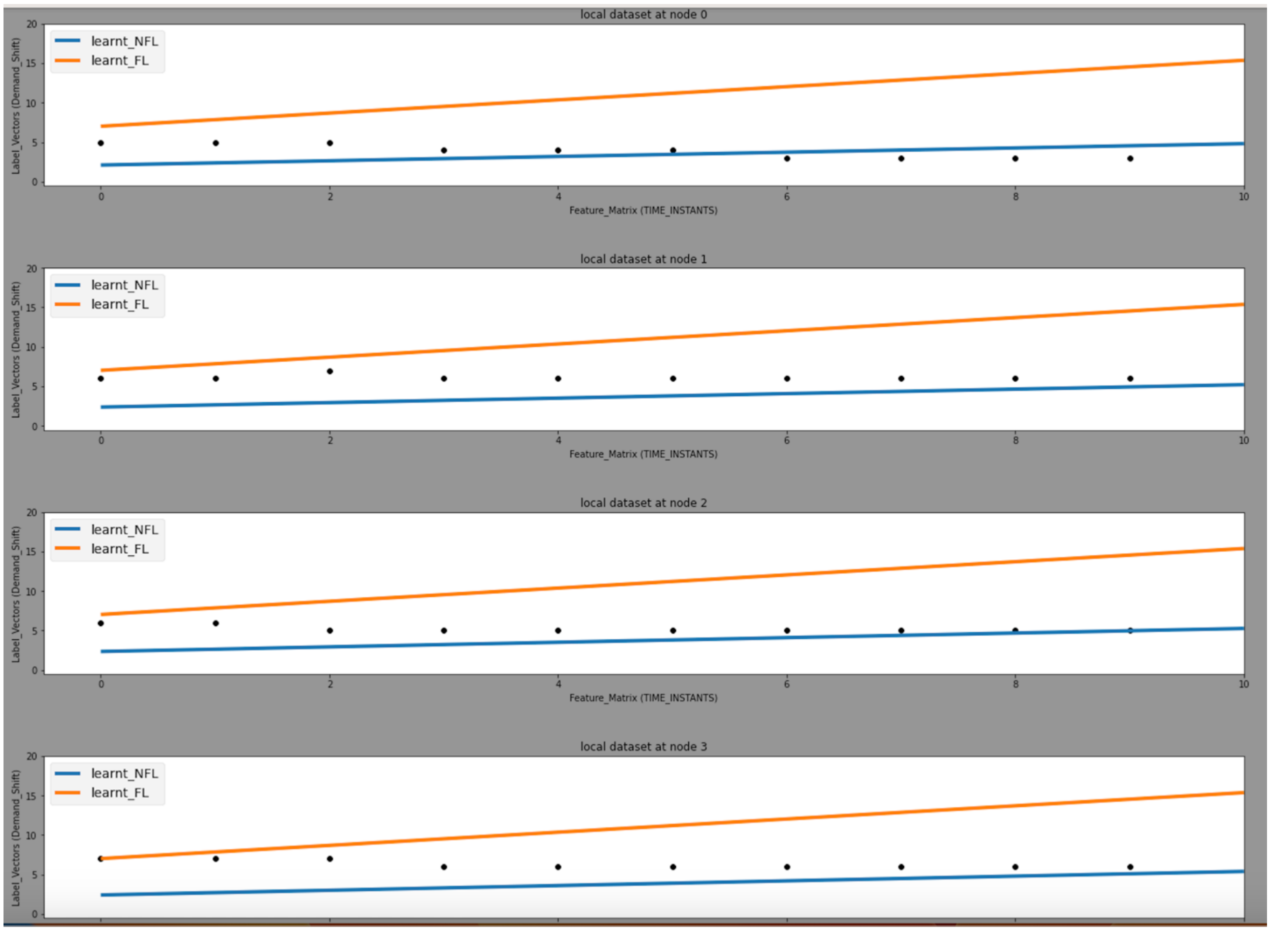


Figure 18a. Learnt weight of NFL vs FL

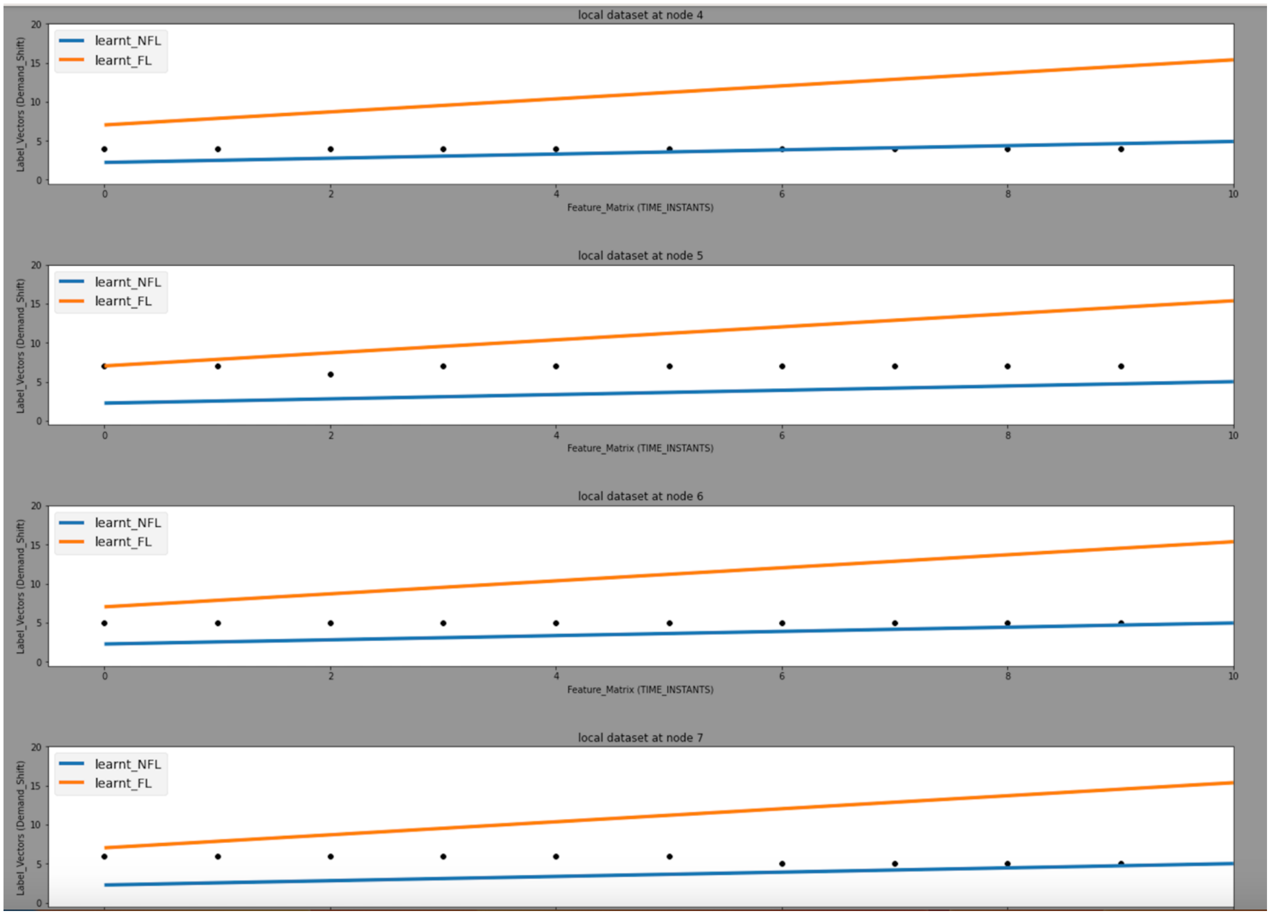


Figure 18b. Learnt weight of NFL vs FL

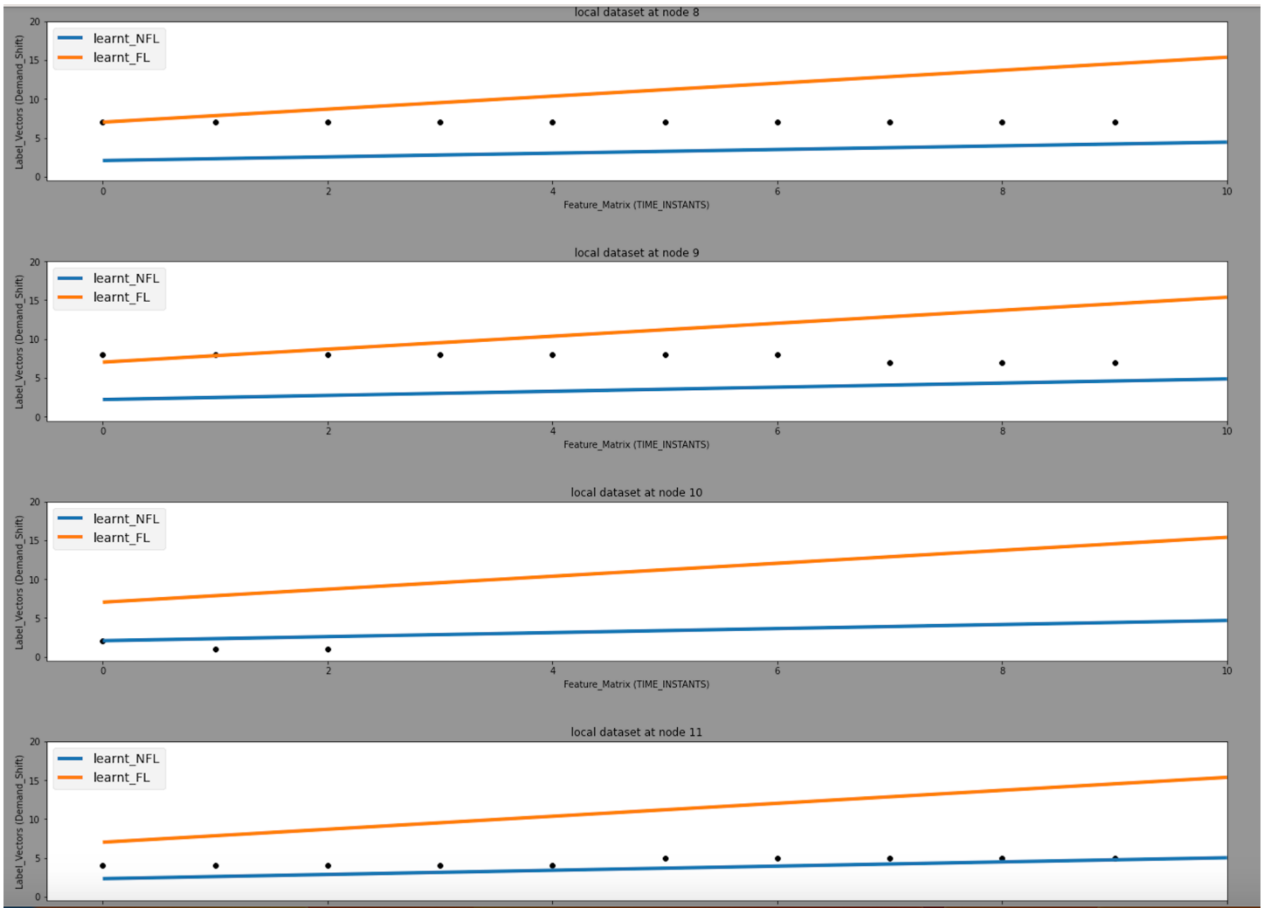


Figure 18c. Learnt weight of NFL vs FL

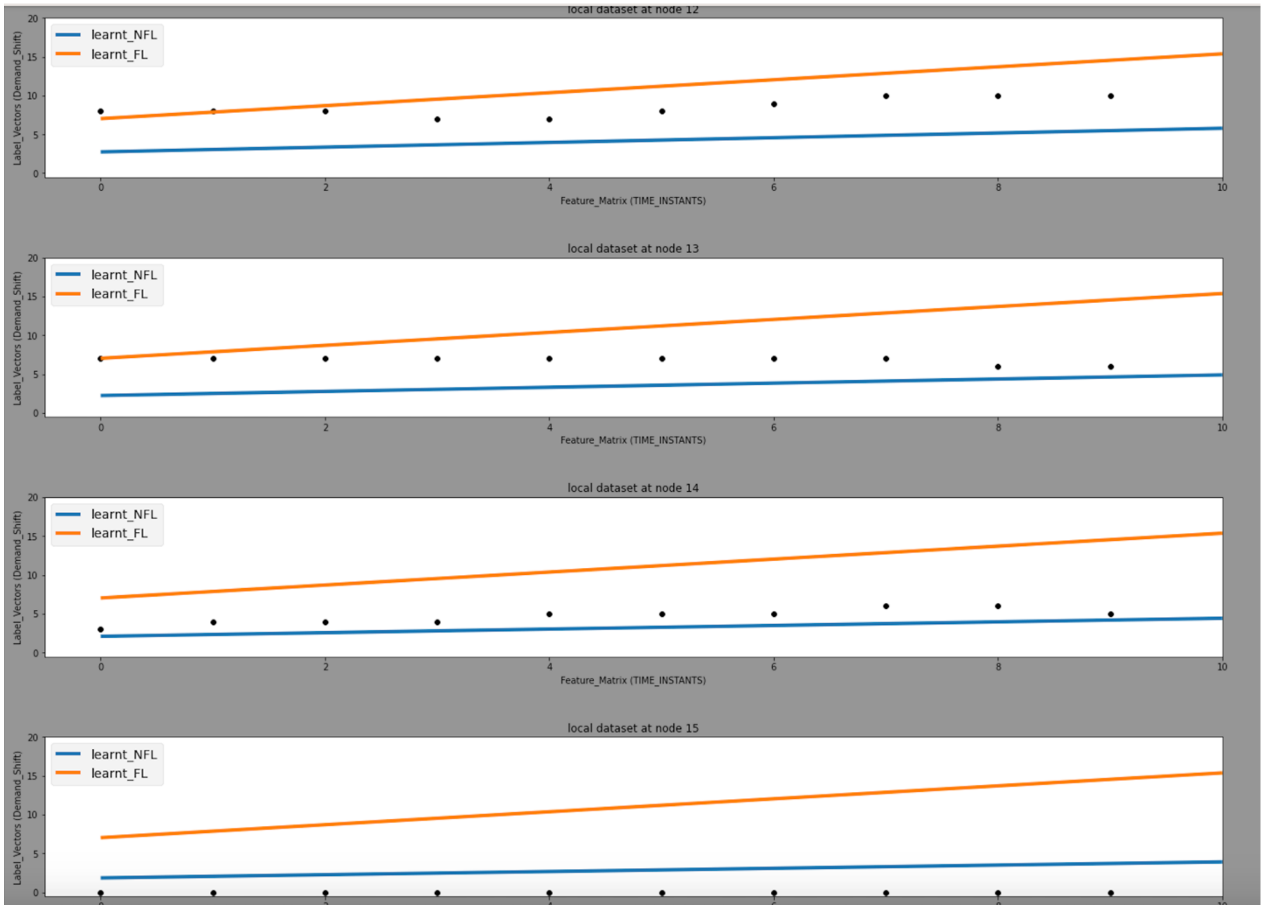


Figure 18d. Learnt weight of NFL vs FL

Figure 18 (a, b, c, d). shows the comparison of learnt weight of NFL and FL at each node, and here we see that the behavior of regression line of NFL and FL. On x-axis we have feature matrix (time instants) and on y-axis we have label matrix (quantity of vehicles shift from one node to the other). The scattered plot shows the number of vehicles changing in each time instant. Now here we see that the regression line of NFL converges more towards the original data and for each node and we have personalized model. But if we see the behavior of FL, we see that here we have only one global model and validation error in case of FL is quite large.

Furthermore, to see the comparison of NFL and FL we find the MAE due to NFL and FL. To do it we test the learnt data. We have created another mobility model that will be our dataset for testing (test data). Then we tested the predictions of NFL and FL by calculating the Mean Absolute Error of trained weights with respect to the test data.

Figure 19. below shows the result of MAE of NFL and FL.

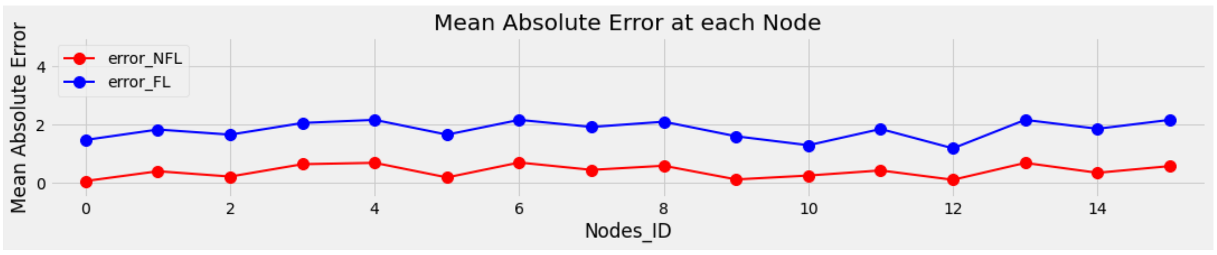


Figure 19. MAE of NFL and FL

If we see Figure 19. The red plot indicates the MAE due to NFL and the blue plot shows the MAE due to FL. On x-axis we have nodes_id i.e., in our case (Node_0 to Node_15) on y-axis we have MAE for each node and here we clearly see that MAE of NFL is quite less than MAE of FL. Which means the performance of MAE of NFL is quite better than the MAE of FL.

To further check the performance, we also have changed some parameters like Gradient descent (GD) steps, number of time instants and the number of nodes to see results of our NFL algorithm.

To check the MAE with respect to different GD steps, we have calculated the average of MAE for all nodes at each GD step size. So now in this case we have one value of MAE at each GD step. In our case we have considered four GD step sizes.

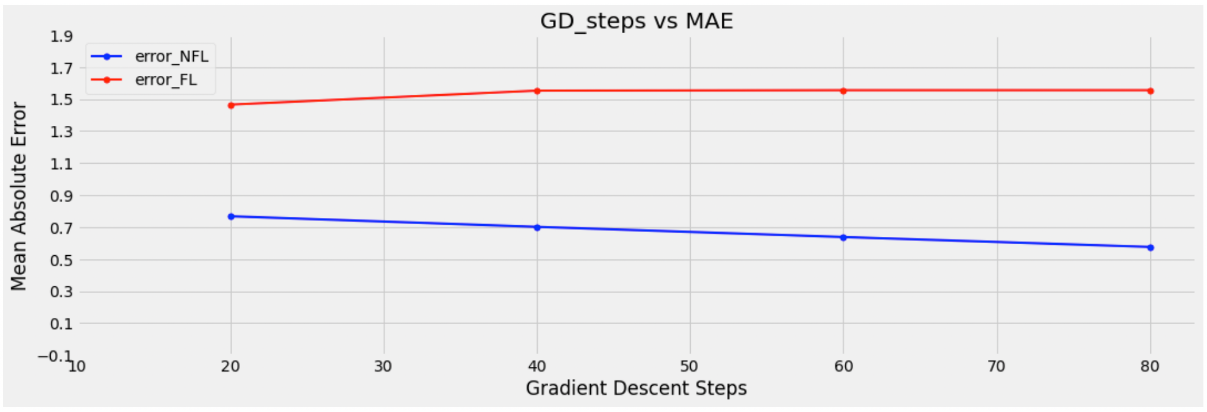


Figure 20. GD steps vs MAE

Here in Figure 20. We see the results of MAE with respect to different GD steps size for NFL and FL. On x-axis we have GD step size and on y-axis we have the MAE. We considered four different GD step size i.e., (20, 40, 60, 80). So, at first, we will see the behavior of MAE when GD iteration is small i.e., 20 and then we keep on increasing the value of GD iterations to see the behavior of our NFL algorithm. Here we see that if we are increasing the step size, we are reducing the MAE for NFL until a point where we have overfitting. If we compare the MAE of NFL with the MAE of FL, we see that we have lower values of error in NFL even by changing the gradient steps. Furthermore, if we see the MAE of FL, there is almost no change in performance. The reason is we are considering limited amount of data for training and even with this limited data we are having better performance with NFL.

Now we see the behavior of our algorithm if we change the feature matrix i.e., time instants. In our case now instead of 10 time instants we will have 20 time instances and we will see how the regression line of NFL and FL converges to original data by changing the feature matrix.

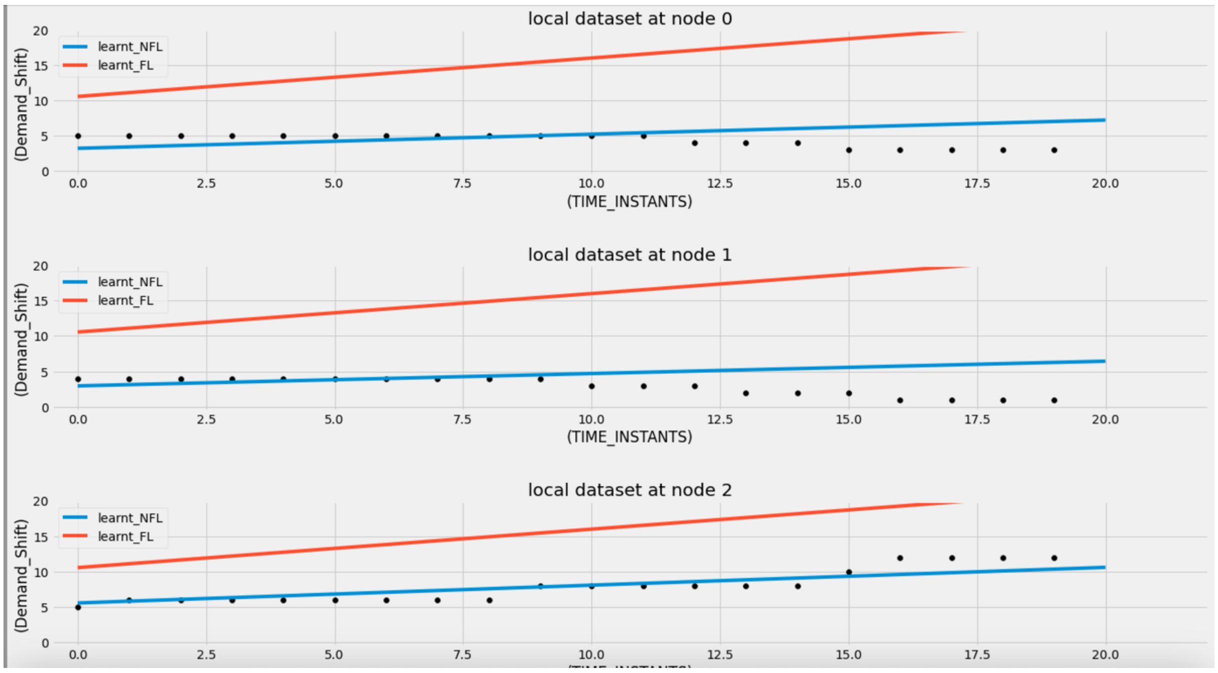


Figure 21a. Behavior of learnt weights with different time instance.

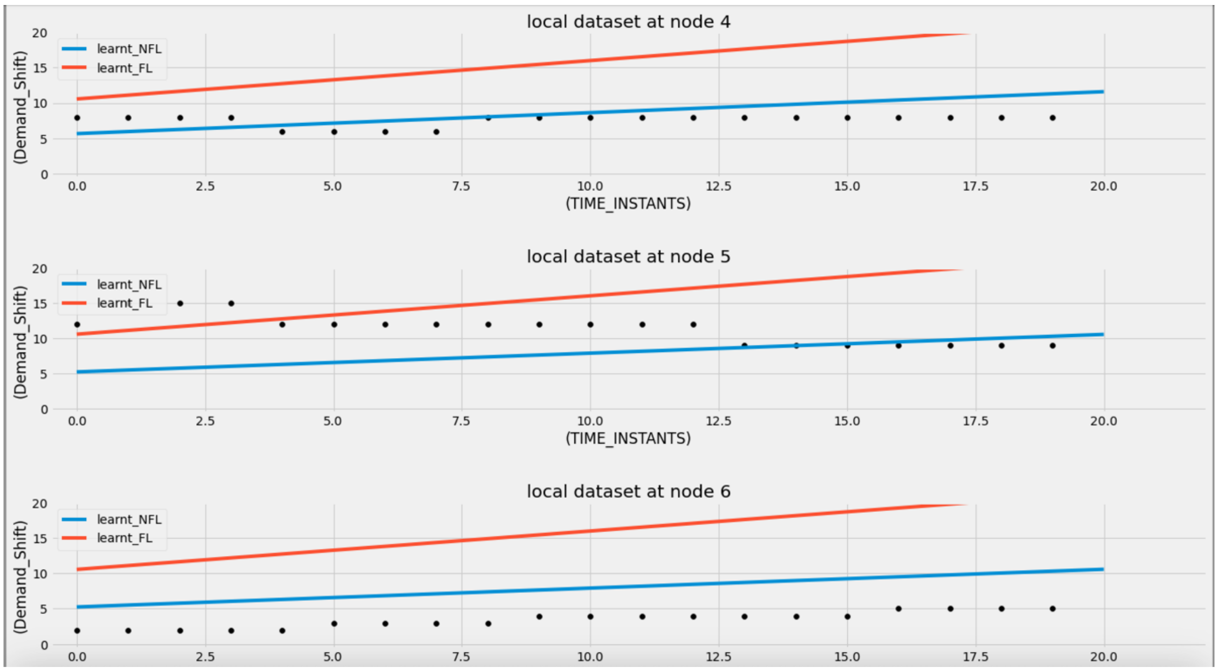


Figure 21b. Behavior of learnt weights with different time instance.

Now if we see the above Figure 21(a, b), in the regression line with 20-time instances we see that the regression line of NFL converges more towards the original data with respect to the FL. So, this proves that our algorithm works better even if we have a feature matrix which is quite large.

Now we will see the behavior of NFL algorithm with a greater number of label vectors i.e., if we increase the number of nodes and hence so the quantity of vehicle shifts from one node to the other increases. In this case we have 25 nodes instead of 16 and we will compare the learnt weights obtained from NFL algorithm with respect to the FL algorithms to see which one is performing better.

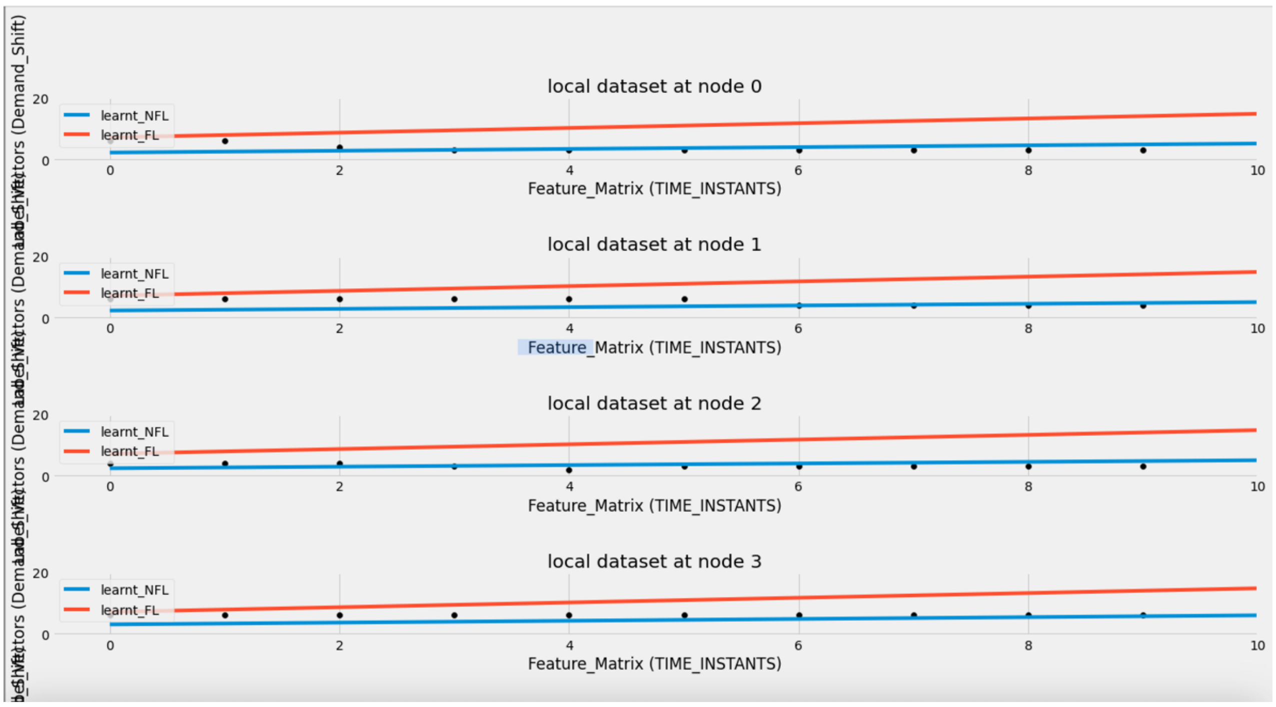


Figure 22a. Behavior of learnt weights with increasing No of nodes.

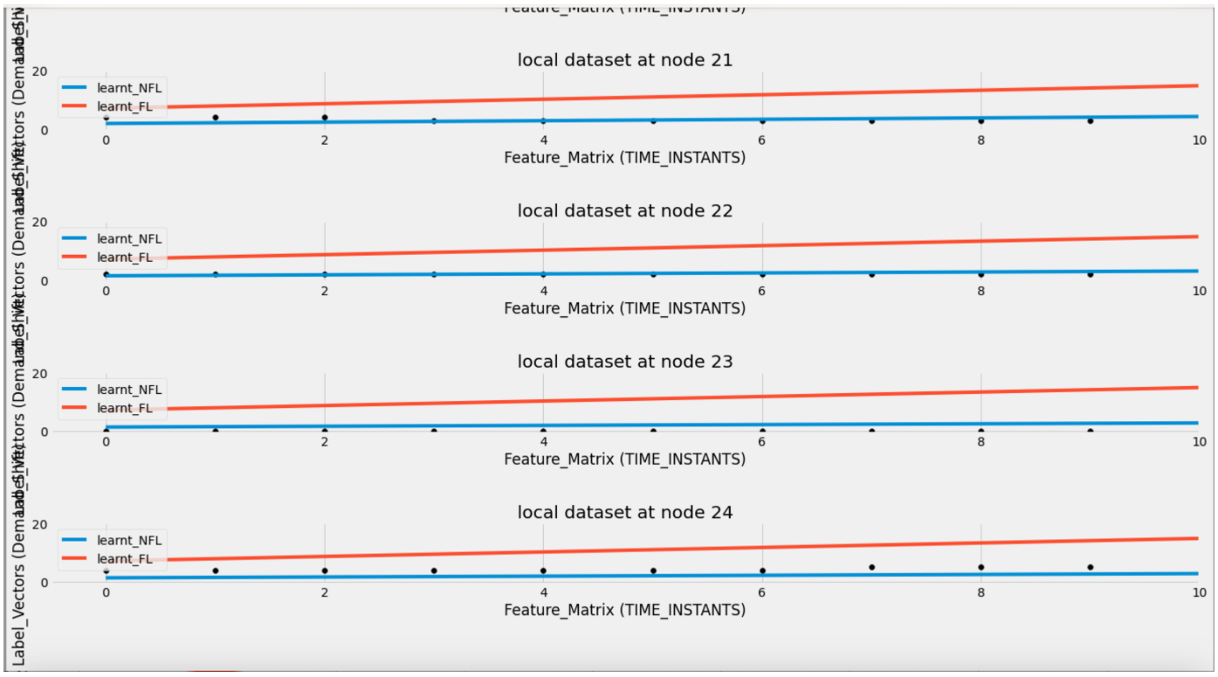


Figure 22b. Behavior of learnt weights with increasing No of nodes.

Figure 22 (a, b) shows the behavior of regression line for NFL and FL of the learnt weights at each node from Node_0 to Node_24. So, we see that regression line of NFL is converging more towards the original data set with respect to the regression line of FL.

This concludes that even with changing different parameters our algorithm for NFL has quite better performance with respect to traditional centralized FL. We get the better value of MAE, the regression line with respect to NFL is performing better as compared to that of FL and more importantly we introduce the concept of clusterdness in which we do not have a global model for the whole network instead the nodes which are close to one another and having similar characteristics can be considered into a single cluster. We can also have a personalized model for each node.

Conclusion

We discuss the use of GTV (Generalized Total Variation) Minimization methods for distributed learning of personalized models in networked collections of local datasets. We considered the mobility model of vehicles to model the movement of vehicles within an area to better analyse the change of traffic within the city at different time instances. We then considered a Manhattan Grid based Model for our project to evaluate the performance of our proposed algorithm. Using a grid-based structure of (4*4) matrix for our mobility model, where we have 16 nodes/areas within a city with different data of vehicles movement at different time instants.

The process of training involves using the GTV of local model parameters as a regularization term, which assumes that the statistical properties or similarities between local datasets are reflected by a known network structure. The resulting scalable NFL (Network federated learning) methods are achieved by solving GTV minimization using distributed optimization methods that are obtained from jointly solving GTV minimization with a network flow optimization problem. The message-passing algorithm that results from this approach leverages the known network structure of data to adaptively pool local datasets into clusters, allowing for the learning of personalized models for local datasets.

We then compare our results obtained by NFL with traditional centralized Federated Learning (FL). We have seen that NFL outperforms FL in every aspect. We have seen that our proposed algorithm obtains better results compare to FL in terms of MAE and learnt weights at each node. We have also seen that even by changing different parameters like data size, no of data points and GD step size we always obtain better results as compared to traditional FL.

Furthermore, we obtained personalized model for each node and we have also introduced the concept of clustering which means the nodes close to each other doesn't vary too much so they can be considered into one cluster. Using our algorithm for GTV we can control the clusterdness of the learnt hypothesis. While in the case of traditional FL we only obtain one single model for the whole network which might not be useful sometimes when our data varies drastically and hence using a global model result in much greater error.

Acknowledgement

I would like to express my sincere gratitude to all those who have supported me throughout my thesis journey. First and foremost, I would like to thank my supervisor **Prof. Daniele Tarchi**, and co-supervisor **Mr. Swapnil Sadashiv Shinde** for their valuable guidance. Their insightful comments and instructions have been instrumental in shaping my research.

I am also grateful to the faculty members and staff of "Department of Electrical, Electronic and Information Engineering / University of Bologna", who have provided me with a stimulating and supportive academic environment. I would like to extend my heartfelt thanks to my colleagues and friends who have shared their knowledge, experiences, and encouragement during this journey.

Furthermore, I would like to thank my family for their constant love and support. Their unwavering faith in me has been a source of motivation and inspiration.

References

[1] J. G. Andrews et al., “What Will 5G Be?” IEEE Journal on Selected Areas in Communications, vol. 32, no. 6, pp. 1065-1082, June 2014.

[2] M. Latva-aho, K. Leppnen (eds.), ”Key Drivers and Research Challenges for 6G Ubiquitous Wireless Intelligence,” 6G Flagship, September 2019.

[3] W. Saad, M. Bennis and M. Chen, ”A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems,” IEEE Network, 2020.

[4] Z. Zhang et al., ”6G Wireless Networks: Vision, Requirements, Architecture, and Key Technologies,” IEEE Vehicular Technology Magazine, vol. 14, no. 3, pp. 28-41, September 2019.

[5] J. He, K. Yang and H. -H. Chen, "6G Cellular Networks and Connected Autonomous Vehicles," in IEEE Network, vol. 35, no. 4, pp. 255-261, July/August 2021, doi: 10.1109/MNET.011.2000541.

[6] Li, Y., Zhang, Y., & Zhang, X. (2021). 6G-Enabled Intelligent Transportation Systems: Architecture, Challenges, and Future Directions. IEEE Network, 35(2), 44-51.

[7] Zhang, J., et al. (2020). Towards 6G Networks: Use Cases and Technologies. China Communications, 17(11), 1-14.

[8] G. Naik, B. Choudhury, J. Park, ”IEEE 802.11bd & 5G NR V2X: Evolution of Radio Access Technologies for V2X Communications,” IEEE Access, vol. 7, June 2019.

[9] S. Lien, et al., ”3GPP NR Sidelink Transmissions Toward 5G V2X,” IEEE Access, vol. 8, pp. 35368-35372, Feb. 2020.

[10] H. Abou-zeid, F. Pervez, A. Adinoyi, M. Aljlayl and H. Yanikomeroglu, ”Cellular V2X Transmission for Connected and Autonomous Vehicles Standardization, Applications, and Enabling Technologies,” IEEE Consumer Electronics Magazine, vol. 8, no. 6, pp. 91-98, November 2019.

[11] https://alexjungaalto.github.io/MLBasicsBook.pdf

[12] Liu, J., Huang, J., Zhou, Y. *et al.* From distributed machine learning to federated learning: a survey. *Knowl Inf Syst* **64**, 885–917 (2022). https://doi.org/10.1007/s10115-022-01664-x

[13] X. Zhou, W. Liang, J. She, Z. Yan and K. I. -K. Wang, "Two-Layer Federated Learning With Heterogeneous Model Aggregation for 6G Supported Internet of Vehicles," in IEEE Transactions on Vehicular Technology, vol. 70, no. 6, pp. 5308-5317, June 2021, doi: 10.1109/TVT.2021.3077893.

[14] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[15] Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. Springer.

[16] Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, D’Oliveira RG, et al. (2021) Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1)

[17] Zhu H, Zhang H, Jin Y (2021) From federated learning to federated neural architecture search: a survey. Complex Intell Syst

[18] Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, D'Oliveira RG (2019) Advances and open problems in federated learning. ArXiv preprint arXiv:1912.04977
[19] Google Research Blog. "Advancing federated learning with differential privacy and beyond.
" https://ai.googleblog.com/2019/06/advancing-federated-learning-with.html
[20] S. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. "Federated learning: Strategies for improving communication efficiency." arXiv preprint arXiv:1610.05492 (2016).
[21] SarcheshmehPour, Yasmin; Tian, Yu; Zhang, Linli; Jung, Alexander (2021): Networked Federated Multi-Task Learning. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.14685696.v1
[22] https://www.researchgate.net/publication/356486860_A_Real-Time_Energy-Saving_Mechanism_in_Internet_of_Vehicles_Systems
[23] H. Ates, A. Yetisen, F. Güder, and C. Dincer, "Wearable devices for the detection of covid-19," Nature Electronics, vol. 4, no. 1, pp. 13–14, 2021.
[24] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in 34th Conference on Neural Informatio
[25] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada, 2020.
[26] O. Chapelle, B. Schölkopf, and A. Zien, Eds., Semi-Supervised Learning. Cambridge, Massachusetts: The MIT Press, 2006.
[27] L. Jacob, J.-P. Vert, and F. Bach, "Clustered multi-task learning: A convex formulation," in Advances in Neural Information Processing Systems, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21. Curran Associates, Inc., 2009.
[28] M. E. J. Newman, Networks: An Introduction. Oxford Univ. Press, 2010.
[29] A. Jung, "Learning the conditional independence structure of stationary time series: A multitask learning approach," IEEE Trans. Signal Processing, vol. 63, no. 21, Nov. 2015.
[30] J. Yoon, M. Liu, and B. Noble, "Manhattan gridlock: An evaluation framework for urban vehicular wireless networks," in Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, Los Angeles, CA, USA, Sept. 2004, pp. 69-78. DOI: 10.1145/1023875.1023885.
[31] Camp, T., Boleng, J., & Davies, V. (2002). A survey of mobility models for ad hoc network research. Wireless Communications and Mobile Computing, 2(5), 483-502. Saha, S., & Johnson, D. B. (2007). Modeling and simulation of wireless and mobile systems. John Wiley & Sons.
[32] Bai, F., & Helmy, A. (2005). A survey of mobility models in wireless ad hoc networks. In Wireless Ad Hoc and Sensor Networks (pp. 483-527). Springer.
[33] A. Mahmoudi, H. Shokri-Ghadikolaei, and C. Fischione, "Cost-efficient distributed optimization in machine learning over wireless networks." organization, 2020
[34] A. Jung and N. Tran, "Localized linear regression in networked data," IEEE Sig. Proc. Lett., vol. 26, no. 7, Jul. 2019.

[35] P. J. Brockwell and R. A. Davis, Time Series: Theory and Methods. Springer New York, 1991.
[36] L. Condat, “A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms,” Journal of Opt. Th. and App., vol. 158, no. 2, pp. 460–479, Aug. 2013.
[37] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, ser. Springer Series in Statistics. New York, NY, USA: Springer, 2001.
[38] O. Chapelle, B. Schölkopf, and A. Zien, Eds., Semi-Supervised Learning. Cambridge, Massachusetts: The MIT Press, 2006.
[39] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 1273–1282.
[40] A. Jung, N. Tran, and A. Mara, “When is Network Lasso Accurate?” Front. Appl. Math. Stat., vol. 3, Jan. 2018.
[41] https://machinelearningmastery.com/what-is-a-feature-matrix-in-machine-learning/
[42] https://machinelearningmastery.com/what-is-a-label-vector-in-machine-learning/
[43] https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/sample-size/
[44] Sarcheshmehpour, Y., Leinonen, M., & Jung, A. (2021). Federated learning from big data over networks. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (Vol. 2021-June, pp. 3055-3059). (ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings). IEEE. https://doi.org/10.1109/ICASSP39728.2021.9414903
[45] X. Zhou, W. Liang, J. She, Z. Yan and K. I. -K. Wang, "Two-Layer Federated Learning With Heterogeneous Model Aggregation for 6G Supported Internet of Vehicles," in IEEE Transactions on Vehicular Technology, vol. 70, no. 6, pp. 5308-5317, June 2021, doi: 10.1109/TVT.2021.3077893.
[46] RENCES [1] S. Cui, A. Hero, Z.-Q. Luo, and J. Moura, Eds., Big Data over Networks. Cambridge Univ. P
[47] M. Wollschlaeger, T. Sauter, and J. Jasperneite, “The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0,” IEEE Industrial Electronics Magazine, vol. 11, no. 1, pp. 17–27, 2017.
[48] F. Sattler, K. Müller, T. Wiegand, and W. Samek, “On the byzantine robustness of clustered federated learning,” in 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 8861–8865.

List of figures