Federated Learning With Distribution Models

A Project Report

submitted by

SRAVANI REDDY GUNJAPALLI

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING
INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

NEW DELHI- 110020

14-06-2024

THESIS CERTIFICATE

This is to certify that the thesis titled **"Federated Learning With Distribution Models"** submitted by **Sravani Reddy Gunjapalli** for the partial fulfillment of the requirements of Master of Technology in Computer Science and Engineering is a record of the bonafide work carried about by her under my guidance and supervision at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere for the reward of any other degree.

Dr. Bapi Chatterjee
Thesis Supervisor
Assistant Professor
Department Of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

Date: June 14, 2024

ACKNOWLEDGEMENTS

It is with great pleasure that I express my deepest gratitude to my mentors and friends, who provided immense support throughout my thesis journey. I am particularly thankful to my supervisor, Dr. Bapi Chatterjee, for his invaluable technical advice and motivation, which were crucial to my work. His insightful suggestions and inspiring guidance significantly shaped my thesis, bringing it to a fulfilling conclusion. I am also profoundly grateful to Mehak Bansal (M.Tech) for her outstanding collaboration during this study. Additionally, I thank the IIITD administration for supplying all the necessary facilities that facilitated my research. Finally, I want to thank my family for their unwavering support during my thesis.

ABSTRACT

KEYWORDS: MDN ; Federated Learning ; ASR ; Privacy-focused ; complex probability distributions ; centralized data; uncertainity

This thesis investigates the integration of Mixture Density Networks (MDNs) within Federated Learning (FL) frameworks to tackle challenges in data privacy, security, and model robustness, focusing on Automatic Speech Recognition (ASR) systems. Traditional ASR systems and other machine learning applications face privacy concerns and difficulties with data variability. Integrating MDNs in a federated learning context offers a novel solution by predicting parameters of probability distributions instead of fixed point estimates. Additionally, this research implements a Federated Gaussian Mixture Model (GMM) to showcase federated learning's flexibility in various tasks. Federated learning enables training MDNs and GMMs across decentralized devices, keeping data localized and enhancing privacy. The outcomes contribute to a deeper understanding of federated machine learning models, especially in privacy-sensitive healthcare and mobile communications applications. The findings lay the groundwork for future applications where MDNs can handle data uncertainties within a federated learning framework.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

iiitd	Indraprasta Institute of Technology,Delhi
FL	Federated Learning
MDN	Mixture Density Network
ASR	Automatic Speech Recognition
IID	Identically and Independently Distributed
MFCC	Mel-Frequency Cepstral Coefficient
WER	Word Error Rate
OpenAI	Open Source Artificial Intelligence
GPU	Graphics Processing Unit

NOTATION

r	Radius, m
α	Learning Rate
μ	Hyperparameter that controls strength of the regularization
ω	Local model parameters
ω_t	Global model parameters
π	Mixing coefficients

CHAPTER 1

INTRODUCTION

Machine learning focuses on creating mathematical models that analyze data to predict outcomes or make informed decisions. Traditionally, these models depended on centralized approaches for collecting and processing data. Initial machine learning systems were created in controlled settings where significant amounts of data were gathered, stored and analyzed on centralized servers or within single, typically isolated, organizational databases.

Centralized Learning Paradigm in conventional centralized machine learning setups, data from multiple sources is compiled into a single central repository. This centralized data aggregation is crucial during the model training phase, allowing algorithms to adjust their parameters across the complete dataset to enhance the accuracy of predictions or classifications. Such centralized models depend extensively on solid infrastructure that can handle large data storage volumes and deliver the necessary computational capacity for processing this data.

Data Silos within Organizations Traditionally, organizations have managed their data independently, resulting in the formation of data silos. These silos are controlled by specific departments and are isolated from other parts of the organization, effectively keeping data fixed within confined boundaries. This method is beneficial for managing proprietary information and safeguarding departmental data privacy. However, it presents substantial obstacles for machine learning endeavors, which thrive on varied and extensive datasets to enhance accuracy and diminish biases. These challenges are particularly pronounced during data collection and consolidation phases, where the need for comprehensive data access conflicts with the compartmentalized nature of data storage. Collecting and consolidating data in a centralized model involves multiple challenges:

Challenges as data volumes increase, the necessary infrastructure must also expand, often resulting in higher costs and greater complexity in managing data. Centralizing sensitive information heightens the risk of significant data breaches, as all data concentrated in one location creates a single point of failure. Moreover, sharing data across departments or with external parties involves navigating through a maze of regulatory and logistical hurdles, with strict laws like the EU's General Data Protection Regulation (GDPR) complicating data access and transfers. Additionally, in sectors requiring real-time operations, such as financial services or online retail, the latency associated with centralized data processing can lead to missed opportunities and delayed responses.

Evolution Towards Distributed Learning the challenges of centralized machine learning, especially regarding privacy, scalability, and operational efficiency, have prompted a shift towards distributed learning methodologies. Federated learning emerges as a prominent solution, addressing these challenges by decentralizing the data handling process mentioned in [5] and [6]. This approach keeps data localized at its source, significantly enhancing privacy and security. By reducing reliance on vast central data repositories, federated learning supports privacy by design and minimizes the infrastructural demands typically associated with traditional machine learning methods. This model allows for scalable and efficient processing, as it distributes the computational workload across multiple devices or nodes, each handling its subset of data.

1.1 Context and Background

Distributed Machine Learning systems play a crucial role in improving performance, increasing accuracy, and processing larger datasets effectively. These systems are designed to minimize errors and enable precise decision-making from extensive data sources. They are particularly adept at managing large datasets' scalability and computational challenges. Distributed machine learning becomes indispensable when traditional algorithms fail due to memory constraints, distributing the computational tasks across various workstations to expedite processing. In this setup, data is partitioned and allocated to different nodes or machines where each independently trains a model on its local dataset. The training results, precisely the model parameters or learned updates, are collected and integrated at a central server or node to form a cohesive global model. How federated learning emerged from distributed learning is discussed in [8].

Federated Learning is a distributed machine learning approach where multiple devices or servers, known as clients, collaborate to train a standard predictive model. In this setup, each client keeps its training data private and does not share it with a central server or other clients. Rather than transferring data, clients compute updates (such as gradients) and send these to a central server. The server aggregates these updates to enhance the global model, which is redistributed to the clients for additional training rounds.

Federated learning stands out as a specialized form of distributed learning that enhances privacy and manages data directly on local devices. This approach is especially beneficial for processing uneven, non-independent, and Identically Distributed (non-IID) data across mobile devices. With the enormous data produced by smartphones, using this data locally for on-device inference optimizes operational efficiency, conserves battery life, and boosts data privacy. For example, as discussed by [3], Google's Gboard keyboard enhances user privacy by utilizing a local cache for interactions that support Federated Learning processes, reducing the dependency on continuous server connections and enabling faster and more private user experiences. Federated learning is increasingly recognized as a vital technology in addressing key challenges in our data-centric world with many applications, as mentioned in [7]. This helps in Privacy

Preservation By allowing data to remain on local devices, federated learning minimizes the risk of personal data exposure, making it highly beneficial in sectors where privacy is paramount, such as healthcare and finance. Since only relevant model updates are transmitted, rather than large datasets, federated learning reduces the required bandwidth, alleviating network congestion. Since there is no single point where all data is stored, the risks associated with data breaches are significantly reduced.

Mixture Density Networks(MDNs) it is a unique neural network architecture tailored for probabilistic regression, offering a way to estimate uncertainty alongside predictions. This characteristic differentiates them from conventional regression models, which typically deliver point estimates without indicating the associated uncertainty. MDNs enhance the prediction process by determining parameters of probability distributions, such as means and standard deviations, rather than just outputting single-point values. This feature allows them to encompass the uncertainty inherent in predictions. Typically, MDNs employ Gaussian distributions, where the network predicts the mean and standard deviation through distinct neurons in its output layer. This adaptability makes MDNs suitable for both distributed learning and federated learning frameworks.

Mixture Density Networks (MDNs) are particularly effective in scenarios where a single input may correspond to multiple potential outputs. For instance, in image processing, the same object could be labeled differently based on varying perspectives, or in voice recognition, a particular sound could be transcribed into various words based on the context. In federated learning environments, which often involve non-IID (non-Independently and Identically Distributed) data across numerous devices, MDNs are highly capable of capturing the varied data distributions characteristic of such decentralized datasets. This adaptability makes them highly suitable for complex and diverse data environments in federated learning setups.

1.2 Literature Survey

From existing papers Mixture Density Networks (MDNs) represent an innovative approach for expanding the capabilities of conventional neural networks, allowing them to model arbitrary conditional probability distributions. This feature addresses the limitations of traditional networks, which primarily provide point estimates or conditional averages that often fail to capture the total variability and uncertainty inherent in complex datasets. MDNs combine a typical neural network with a mixture density model, usually Gaussian, to predict parameters of this distribution (means, variances, and mixture coefficients) based on input data. Unlike standard neural networks that output a single value, MDNs output parameters defining an entire probability distribution. This allows for a richer representation of data uncertainties and relationships. The main area to explore is federated learning according to paper [18].

Various studies have explored the application of federated learning to improve the privacy and effectiveness of ASR systems. These studies typically focus on adapting the centralized training processes of deep neural networks used in ASR to a decentralized environment. Challenges in federated learning for ASR include maintaining performance while data remains distributed and adapting to data's non-IID (independently and identically distributed) nature across clients. Techniques like model aggregation and client weighting are explored to mitigate these issues.

The existing papers on federated learning in ASR, which I explored, mainly used audio frameworks like espnet and kaldi for automatic speech recognition(ASR) and primarily used federated averaging and fedprox algorithms for federated learning approaches. These researches mainly focused on improving the privacy and personalization in ASR using a federated learning approach.

1.3 Problem

This thesis Explores the incorporation of Mixture Density Networks (MDNs) into federated learning environments, with an initial broad application before specifically targeting improvements in Automatic Speech Recognition (ASR) systems. Traditional ASR systems often grapple with privacy concerns and data variability, issues that federated learning can substantially mitigate. Typically, these systems generate deterministic outputs, neglecting the inherent uncertainties in speech data. By integrating MDNs, which adeptly model the probability distributions of potential outcomes, this research aims to capture the uncertainty and multi-modality inherent in ASR predictions. The thesis will investigate the adaptation of MDNs to federated learning frameworks, evaluate their role in enhancing privacy and data security, and examine their effectiveness in modeling ASR systems under federated learning paradigms.

1.4 Outline

To effectively address the integration of Mixture Density Networks (MDNs) into federated learning frameworks with an application to Automatic Speech Recognition (ASR). Several critical areas are structured around foundational theory, practical implementation, and empirical evaluation to explore. Beginning by establishing the theoretical basis of MDNs and their importance in modeling probability distributions, particularly in environments where data privacy and decentralized processing are prioritized, such as federated learning. The development and adaptation of an MDN architecture suited for federated settings were initially tested on the MNIST dataset to validate the approach before applying it to ASR. This involves customizing the MDN for speech data, ensuring it effectively handles the variability and uncertainty inherent in speech recognition tasks. Implement these models using PyTorch, as outlined in the provided code, where the model training, evaluation, and optimization are conducted across distributed datasets without compromising data privacy. The effectiveness of this integrated approach will be assessed through rigorous testing on speech datasets, analyzing the model's accuracy and robustness against traditional and federated learning benchmarks.

CHAPTER 2

Federated Learnig

2.1 Introduction

Federated Learning emerged as a revolutionary approach in machine learning (ML) technologies. It represents a shift from the traditional centralized model of data processing, where all data must be aggregated on a single server, to a distributed model where the data remains on local devices. This approach enhances privacy and facilitates real-time learning capabilities across various devices.

Federated Learning is a variant of distributed machine learning that facilitates large-scale model training across numerous endpoints while maintaining data privacy. In this approach, the model is distributed to various devices rather than sending data to a central server. Each device trains the model locally using its data, processing updates independently. The devices then send these updates—often gradients or model parameters—back to a central server. At the server, these contributions are aggregated to enhance the model, which is then sent back to the devices for further training iterations. This process repeats until the model reaches optimal performance. This approach is particularly beneficial for privacy preservation, as it eliminates the need to transfer or expose raw data outside of its original environment.

The decentralized nature of Federated Learning makes it ideal for scenarios where data security and privacy are paramount and where data is inherently decentralized across many users or locations. This has implications for consumer devices and industries like finance and telecommunications, where data security is a significant concern.

2.2 Need for FL

The need for Federated Learning (FL) arises from several critical challenges and limitations inherent in traditional centralized machine learning frameworks, especially as data generation and consumption patterns evolve in the age of ubiquitous computing. Here's a detailed look at why Federated Learning is becoming increasingly important:

Privacy and Data Security Concerns conventional machine learning methods require consolidating data from multiple sources into a central server, often necessitating the movement of sensitive or personal information from individual devices to centralized data processing centers, which raises substantial privacy concerns. Risks such as data breaches, unauthorized access, and potential data misuse are prevalent issues. This mainly helps privacy data like hospital data, as mentioned in [16]. Federated Learning offers a solution by keeping the data localized on user devices. Only the model updates are sent to the server, not the raw data. This approach dramatically diminishes the likelihood of sensitive information exposure, making it compliant with stringent privacy laws like the GDPR (General Data Protection Regulation) in Europe.

Bandwidth and Latency sending massive datasets from countless devices to a central server consumes extensive bandwidth and can cause considerable delays in processing data and updating models. This issue is especially critical for applications that depend on real-time or nearly real-time predictions from vast datasets. Federated Learning improves efficiency by spreading computational tasks across numerous devices, including smartphones and IoT devices. This method minimizes the data transmitted over the network, conserving bandwidth and reducing delays.

Data Silos and Accessibility data silos arise when different departments or geographic locations within an organization collect and keep data separately. This separation can obstruct data consolidation for centralized training due to logistical, technical, or regulatory barriers. Federated Learning allows cooperative model training across various organizational and jurisdictional boundaries without centralizing data. Participants train models on their local datasets and share only the updates to the model, effectively dismantling the barriers created by data silos.

Scaling Challenges as data volumes increase and machine learning models become more complex, central servers can become overwhelmed, leading to processing bottlenecks. Enhancing these servers to cope with growing demands can be expensive and inefficient. Federated Learning addresses this problem by distributing the training workload across many devices, each harnessing its computational resources. This decentralized training method reduces the pressure on central servers and effortlessly expands as more devices join the network.

Handling Non-IID Data data across different devices is not uniformly distributed (Non-IID), which can hinder the performance of traditional machine learning models trained on centrally aggregated data. This issue is prevalent in applications like mobile keyboard suggestions or personalized health services. Federated Learning effectively manages this challenge by enabling devices to train models on their unique data independently. This strategy promotes the creation of more tailored and efficient models for users, enhancing their overall experience.

2.3 Data Heterogeneity

Data Heterogenity and Distribution in federated learning describes the variations in data distribution among different client devices involved in training a model. This variance presents substantial challenges within federated learning settings. Effectively tackling these issues is essential for adequately implementing federated learning systems, particularly considering the diverse data types across multiple devices.

Feature Distribution skew different clients may have different distributions of input features. For example, a model trained on smartphone data might receive inputs from high-end and low-end devices with various sensors and data quality and characteristics.

Label Distribution Skew the distribution of labels across clients may vary significantly. In a healthcare application, for instance, some hospitals (clients) might specialize in certain diseases and thus have more samples of related conditions than others.

Quantity Skew often, the data available across different clients must be more balanced. Some clients might generate more data than others due to higher user interaction or longer deployment times.

Challenges from [9] Data heterogeneity among client devices presents numerous challenges in federated learning, including:

1. Model Convergence: Data that is not identically and independently distributed (Non-IID) can adversely impact the global model's rate of convergence. Training on such diverse data might result in slower convergence or failure to achieve optimal global convergence, potentially diminishing model performance.

2. Model Fairness and Bias: Models trained within heterogeneous environments may exhibit good average performance but could underperform on specific clients or data segments. This scenario can give rise to fairness concerns, where the model disproportionately favors the characteristics or labels more prevalent in most of the data.

3. Client Participation and Influence: In federated learning, clients contributing larger or more representative datasets of the overall demographic can disproportionately impact the model. Ensuring all clients contribute equally to the learning process, thus maintaining balance and equity, represents a significant challenge.

Handling data heterogenity several approaches can help address data heterogeneity in federated learning effectively:

1. Client Clustering: By grouping clients based on the similarity of their data distributions, heterogeneity can be better managed. This approach involves either training separate models within each cluster or applying different weighting schemes to model updates according to the characteristics of each cluster.

2. Robust Aggregation Algorithms: Enhancing the federated averaging algorithm to handle outliers and skewed distributions better is crucial. Aggregation methods such as the trimmed mean or median of means can provide more resilient model updates by reducing the influence of extreme values.

3. Personalized Federated Learning: Developing personalized models tailored to the specific data characteristics of each client can improve model relevance and effectiveness. This can be achieved through meta-learning or multi-task learning, which allows customization of the learning process to individual client needs.

2.4 Types of FL

In federated learning various forms are designed to suit distinct data distribution scenarios and privacy needs. Horizontal Federated Learning, often called sample-based federated learning, is practiced when participants (like mobile devices or local data centers) possess datasets with identical features but different individual entries. An example can be multiple hospitals across various regions, each hospital collects the same types of data about different sets of patients.

Horizontal Federated Learning where each participant independently trains a model on their data and then sends updates like gradients or model parameters to a centralized server. These updates are combined, generally via a method like federated averaging, to enhance a collective global model. This model is then redistributed to all participants for further enhancements in subsequent training cycles. A significant benefit of this approach is that it integrates diverse data insights from across all participants, which can enhance the model's generalizability and robustness.

Vertical Federated Learning Also known as feature-based federated learning, it is employed when various organizations gather distinct feature sets about the same individuals. This arrangement is prevalent in sectors like finance and retail, where different companies may collect unique data types about the same customers. In Vertical Federated Learning (VFL), the participants collaborate to develop a model by aligning their datasets using a common identifier, such as a customer ID, while keeping the data private. Each participant calculates updates for the model based on their unique set of features. These partial updates are combined with other participants to create a comprehensive update reflecting the entire data feature space, enhancing the model's accuracy.

Federated Transfer Learning is utilized when a robust model trained on a large and diverse dataset can be adapted to work with data on local devices that may not have enough label diversity or data volume (the target domain). This approach is useful in scenarios where local devices generate specialized data not needing to be presented in large-scale models' training stages. The global model is first trained on the central

server's extensive dataset in FTL. The model's knowledge is then transferred to local models. These local models fine-tune the pre-trained global model using their local data, enabling personalized adjustments while benefiting from the learned features of the global model.

Cross-silo Federated Learning is applicable in environments where data is stored across different organizational silos or geographical locations and cannot be pooled due to privacy, policy, or competitive reasons. Examples include multinational corporations that operate in multiple regulatory environments or competing businesses that wish to collaborate without revealing sensitive information. Like horizontal federated learning, cross-silo FL involves training local models on each silo's data. However, unlike conventional FL, where edge devices (like smartphones) are used, cross-silo FL typically involves secure, server-grade machines within each organization. Model updates are exchanged between silos via a central coordinator or through a decentralized protocol, ensuring that each participant can benefit from the aggregated insights without compromising their data sovereignty.

2.5 Algorithms

Federated Learning (FL) algorithms enable distributed devices to collaboratively learn a shared prediction model while keeping the training data localized. Among these algorithms, Federated Averaging (FedAvg) and FedProx are particularly notable for their foundational and advanced contributions to the field. Here's an in-depth look at both, including their mathematical underpinnings and algorithms. Different algorithm's performance is discussed in the paper by [13]

Algorithm 1 Federated Averaging (FedAvg). Total number of clients is K, N is the number of clients randomly selected in each round, E is number of local epochs performed by each client, B is mini-batch size, η is the learning rate.

Inputs: K, N, E, B, η
Initialize w_0
for each round $t = 1, 2, \ldots, T$ **do**
    $S_t \leftarrow$ (randomly select N clients from K)
    for each client $k \in S_t$ in parallel **do**
        $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    end for
    $w_{t+1} \leftarrow \sum(\frac{n_k}{n}) \cdot w_{t+1}^k$ ▷ Aggregate updates
end for
procedure CLIENTUPDATE(k, w)
    $w_k \leftarrow w$
    for each local epoch i from 1 to E **do**
        for mini-batch $b \in$ Randomly partition local data into batches of size B **do**
            $w_k \leftarrow w_k - \eta \cdot \text{Gradient}(\text{Loss}(w_k; b))$
        end for
    end for
    return w_k to server
end procedure

2.5.1 Federated Averaging

Federated Learning (FL) is a machine learning approach designed to train algorithms across decentralized devices or servers holding local data samples without exchanging them. This technique is beneficial when data privacy is paramount or cannot be centralized due to privacy concerns or logistical issues. One of the foundational algorithms in federated learning is Federated Averaging (FedAvg), proposed by [14] [4] in their seminal paper Communication-Efficient Learning of Deep Networks from Decentralized Data [10].

1. Initialization : A global model is initialized on a central server. The server initializes and distributes the global model parameters to the participating clients. The initial model can be a randomly initialized machine learning, depending on the specific problem and the neural network architecture used.

2. Client Selection : At each training round, a subset of available clients is randomly selected to participate in Training.

3. Local Training : Each selected client receives and improves the current global model using its local data. Each client receives the global model and performs Training on their local dataset. This Training is typically conducted for several epochs or until a specific local convergence criterion is met. The local training process involves calculating the gradients of the model's loss function (which measures prediction errors) concerning the model parameters, followed by updating the model parameters using SGD or other optimization algorithms like Adam.

4. Local Update Submission : After Training, each client computes the difference between the updated model parameters and the parameters initially received. This difference, often in the form of gradients or parameter updates, is sent back to the server. These updates can be compressed or encrypted to reduce communication costs and enhance privacy.

5. Aggregation and Update : The server aggregates these updates by computing their mean and updates the global model. Once the server receives updates from the selected clients, it averages these updates to adjust the global model. Mathematically, if there are K clients, each with parameters θ_k and number of data points N_k the update can be expressed as:

$$\theta_{global}^{new} = \theta_{global}^{old} + \alpha \sum_{k=1}^{K} \frac{n_k}{N} (\theta_k - \theta_{global}^{old})$$

Where α is the learning rate, N is the total number of data points across all clients, and $\theta_{global}^{old}, \theta_{global}^{new}$ are the global parameters before and after the update, respectively.

This process is repeated for several rounds, with the server sending the updated global model back to clients for further training. This iterative process continues until the worldwide model achieves satisfactory performance or converges.

Advantages Network Efficiency: By transmitting only model updates rather than raw data, FedAvg can significantly reduce the amount of data that needs to be sent over the network. Since raw data never leaves the client, Federated Averaging provides a way to build machine learning models without compromising data privacy. Federated Averaging can handle many decentralized clients, making it suitable for scenarios like mobile devices where data is generated by millions of users, thus enabling scalability.

Challenges In real-world scenarios, data distribution across clients can be highly skewed (non-IID, non-independent, and identically distributed). This can lead to models that perform well on some clients but poorly on others. Despite reductions in data transmission, the need to communicate frequently between clients and the server poses bandwidth demands, a large number of clients(Communication Overhead)

2.5.2 FedProx

FedProx [7] is a sophisticated adaptation of the conventional Federated Averaging (FedAvg) method, specifically tailored to more effectively manage the diverse and heterogeneous environments typical in federated learning systems. FedProx is designed to handle discrepancies in computational and communication capacities across the network's devices, called system heterogeneity. It effectively addresses the challenges posed by non-IID data distributions, prevalent in federated settings where each device's data may significantly differ, called statistical heterogeneity. FedProx introduces a proximal term to the local objective function, which penalizes significant deviations of local model updates from the global model. This term is defined as:

$$\frac{\mu}{2}||\omega - \omega_t||^2$$

where μ is a hyper-parameter that controls the strength of the regularization, ω represents the local model parameters, and ω_t indicates the global parameters distributed by

the server. During local training phases, each device optimizes its local loss and this proximal term, thereby ensuring that local updates do not stray too far from the global model parameters, which promotes greater uniformity and stability across the network's updates.. The algorithm works as follows:

Algorithm 2 Federated Proximal (FedProx). Total number of clients is K, N is number of clients randomly selected per round. E is number of local training epochs per client, B is Batch size for mini-batch SGD, is Learning rate for model updates, Proximal term coefficient for regularization

Inputs: K, N, E, B, η, μ
Initialize w_0
for each round $t = 1, 2, \ldots, T$ **do**
    $S_t \leftarrow$ (randomly select N clients from K)
    for each client $k \in S_t$ in parallel **do**
        $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    end for
    $w_{t+1} \leftarrow \sum(\frac{n_k}{n}) \cdot w_{t+1}^k$ ▷ Aggregate updates
end for
procedure CLIENTUPDATE(k, w)
    $w_k \leftarrow w$
    for each local epoch i from 1 to E **do**
        for mini-batch $b \in$ Randomly partition local data into batches of size B **do**
            $w_k \leftarrow w_k - \eta \cdot (\text{Gradient}(\text{Loss}(w_k; b)) + \mu \cdot (w_k - w))$
        end for
    end for
    return w_k to server
end procedure

1. Randomly select A subset S_t of K devices. The selection process can be randomized or influenced by specific factors such as the availability of devices, their network bandwidth, or their past reliability in contributing to practical training sessions. Introducing randomness in the selection distributes the computational burden more evenly and helps prevent the training process from favoring any specific group of data.

2. Distribute the current global model w_t to each chosen device. This stage is vital for aligning the training round's initial conditions, ensuring that all participating devices start their calculations with the same set of model parameters. Generally, the distribution of these parameters is conducted over a secure network to protect the privacy and integrity of the data.

3. Each device updates its model by minimizing the local loss adjusted for the proximal term. This term penalizes large deviations from the global model, particularly useful in scenarios where devices have non-IID (not identically and independently distributed) data.

4. Devices send their updated models back to the central server. This phase can require sending large amounts of data across networks, which might consume substantial bandwidth. To mitigate this, techniques such as model compression, quantization, or sparsification can be used to lessen the communication load.

5. The server aggregates these models to update the global model. This aggregation is typically a weighted average, where weights could be proportional to the number of data points on each device, ensuring that devices with more data have a higher influence on the final model.

The inclusion The proximal term helps maintain the fidelity of local updates to the global model, and it is beneficial under non-IID data distributions across devices, thus enhancing stability and robustness. FedProx allows devices with varying capabilities to contribute to the model without unduly impacting the overall convergence process. FedProx enriches the federated learning landscape by introducing a regularization mechanism that moderates the updates from diverse devices. This ensures consistent and stable convergence even when faced with significant data and system heterogeneity.

CHAPTER 3

Mixture Density Networks

3.1 Mixture Density Networks

1. Mixture Density Networks (MDNs) [2] represents an advanced neural network architecture designed to address complex problems where the relationship between the input and output is not a simple one-to-one mapping. Unlike traditional neural networks that predict a single deterministic output, MDNs can predict entire probability distributions. This feature suits MDNs for tasks with inherently uncertain or multi-modal production, where a single input might correspond to multiple potential outputs.

MDNs combine a standard neural network with a mixture model, typically a Gaussian mixture model. The neural network does not directly predict the final output. Instead, it outputs the parameters of the mixture model, including:
Mixing coefficients (π): These parameters indicate the weight or probability of each component in the mixture model. They sum up to 1 and dictate how much each element contributes to the overall distribution.
Means (μ): The means of the Gaussian distributions in the mixture. Each mean corresponds to a potential center of the cluster in the output distribution.
Variances (α): These parameters define the spread or variability of each component around the mean. A higher variance means the component covers a broader range of values.

Input Processing : The input features are processed through the neural network layers, similar to a typical neural network. Parameter Prediction: instead of outputting the final target values, the network outputs parameters for the mixture model.
Probability Distribution Construction: A probability distribution is constructed for each output using the predicted parameters. This distribution provides a probabilistic interpretation of the output, given the input. Output Sampling: outputs can be sampled from this distribution, providing different plausible predictions for each input.

Handling Non-IID Data Problem: In federated learning, data across the network can be highly heterogeneous (non-IID). Traditional models might need help with this as they typically expect data distributions that are more or less homogeneous.
MDN Solution: MDNs, by modeling outputs as probability distributions, can naturally handle diverse data scenarios. They can predict a range of possible outputs, each weighted by its likelihood, making them robust to variations in local data distributions across different clients in a federated setup.

Privacy Preservation Since MDNs are trained to output parameters of a probability distribution rather than direct data points, they inherently add an abstraction layer to the output. This characteristic can help enhance data privacy, a crucial aspect of federated learning, by ensuring that the reverse engineering private data from model outputs becomes significantly more challenging.

Improved Model Flexibility and Predictive Performance: The ability of MDNs to model complex and multimodal distributions allows for greater flexibility in adapting to various types of data and tasks, from voice recognition to financial forecasting, where multiple outcomes are plausible. Enhanced Predictive Accuracy: in scenarios where multiple outcomes are possible (scenario analysis in finance, disease prediction in healthcare), MDNs provide a richer set of forecasts, enhancing the decision-making process by offering a spectrum of probable outcomes based on the learned distributions. Implementing Mixture Density Networks in federated learning environments can significantly improve model capabilities, especially in handling privacy-sensitive, multimodal, and non-IID data. By leveraging the distributed nature of federated learning and the probabilistic modeling power of MDNs, this approach can lead to more robust, accurate, and privacy-aware predictive models in various real-world applications.

2. Gaussian Mixture Model: Federated Learning (FL) has emerged as a powerful paradigm that enables training machine learning models across multiple decentralized devices while ensuring that the data remains localized. This approach enhances privacy and facilitates the utilization of diverse data sources without central data aggregation. In this context, we explore the integration of the Gaussian Mixture Model (GMM) with the Federated Learning framework.

GMM is a probabilistic model that assumes that the data is generated from a mixture of several Gaussian distributions with unknown parameters. It is widely used in clustering tasks and unsupervised learning problems. The Expectation-Maximization (EM) algorithm is typically employed to fit the GMM to the data by iteratively refining the parameters to maximize the likelihood of the observed data.

Multiple clients train local models on their respective datasets in our federated setup. A central server coordinates the process by aggregating the client's model updates and updating the global model. The steps involved are as follows:

Client Initialization: Each client initializes a local GMM with its dataset. The clients update their models using the EM algorithm and compute metrics such as AIC (Akaike Information Criterion), BIC (Bayesian Information Criterion), and Log-Likelihood (LL).

Server Initialization: The server initializes a global GMM model using a small subset of the overall dataset. This initialization helps in stabilizing the training process.

Training Process: The training process involves multiple rounds where the server distributes the global model to the clients. The clients update their local models. The server aggregates the updates to refine the international model. This process is repeated for a predefined number of rounds.

The performance of the federated GMM was evaluated on the MNIST dataset. The MNIST dataset contains images of handwritten digits, and we used the pixel values as features for the GMM. The key metric used for evaluation was Log-Likelihood (LL).

CHAPTER 4

Federated Learning for MDN

4.1 Introduction

Federated Learning (FL) applied to Mixture Density Networks (MDNs) represents a novel approach to decentralized machine learning. Traditionally, MDNs are used for modeling complex conditional probability distributions, often for tasks involving uncertainty estimation or multimodal outputs. FL extends this by allowing MDNs to be trained collaboratively across multiple decentralized clients without sharing raw data, thereby addressing privacy concerns and data locality issues.

4.2 Need for Federated Learning for MDN

Privacy Preservation Many applications of MDNs involve sensitive data, such as medical records or personal user information. Federated learning ensures that raw data remains on the local devices, protecting user privacy. Regulatory compliance regulations such as GDPR and HIPAA impose strict requirements on data privacy. Federated learning helps comply with these regulations by minimizing data movement and exposure.

Scalability and Efficiency As in many real-world scenarios, data distribution is distributed across multiple sources. Federated learning leverages this distributed data, enabling training on more extensive and diverse datasets without centralizing the data. Resource utilization by using the computational resources of edge devices and federated learning reduces the burden on centralized servers and can lead to more efficient use of available resources.

Robustness to Data Heterogeneity data collected by different clients may not be independently and identically distributed (Non-IID). Federated learning algorithms, such

as FedProx, are designed to handle this heterogeneity, making them suitable for training MDNs on diverse data sources.

4.3 Implementation

Defining the MDN architecture model is the first step in designing the architecture suitable for the specific task. This typically involves defining the neural network layers and the output layer that predicts the parameters of the mixture distribution. The next step is to choose a federated learning framework setup, such as TensorFlow Federated, PySyft, or a custom implementation. Initialize the global model parameters that will be distributed to all clients. In the next step, we need to initialize client-server communication.

In each round, select a subset of clients to participate in training. Send the current global model parameters to the selected clients for model distribution. Next, we need to start local training on clients for local Updates. Each client trains the MDN locally using its data. This involves multiple local epochs and batch updates. For FedProx, include a proximal term in the local loss function to handle data heterogeneity. Then, in the following step, updates need to be aggregated. Collect the updated model parameters from all participating clients. Aggregate the updates, typically using a weighted average based on the number of data samples each client holds. Finally, a model update needs to be done. Update the global model parameters with the aggregated updates. Repeat the process for a predefined number of rounds or until convergence.

4.4 Challenges

Data Heterogeneity non-IID data clients may have data with different distributions, which can lead to unstable training and poor generalization. Algorithms like FedProx mitigate this, but challenges remain.
Communication Overhead bandwidth limitations, frequent communication between clients and the server can lead to significant bandwidth usage. Efficient communication protocols and reducing the frequency of updates can help alleviate this issue.

Computational Constraints resource variability, clients may have varying computational capabilities, leading to inconsistent training performance. Adaptive algorithms and resource-aware scheduling can help address this. Model Complexity, as large models of MDNs can be complex and require significant computational resources for training. Ensuring efficient training in a federated setting is challenging and may require model compression techniques.

CHAPTER 5

Federated Learning for ASR

5.1 Introduction

Federated Learning (FL) introduces a revolutionary approach to training machine learning models, particularly in sensitive and data-intensive fields such as Automatic Speech Recognition (ASR). ASR systems traditionally rely on centralized data collection and processing to train models that convert spoken language into text. This conventional approach mentioned in [17], while being practical, poses significant privacy risks and logistical challenges, especially as the volume and variety of data increase. Federated learning offers a solution by decentralizing the training process, thus enhancing privacy and data security. Federated learning (FL) is transforming the field of Automatic Speech Recognition (ASR), a technology integral to numerous applications, including voice-activated assistants, transcription services, and interactive voice response systems. Traditionally, ASR systems have relied on centralized training methods, where vast amounts of potentially sensitive speech data are transmitted to and processed on central servers. This poses significant privacy risks and often requires substantial bandwidth. FL offers a novel approach by decentralizing the training process, enhancing privacy, reducing bandwidth usage, and allowing for potentially more prosperous, more diverse data inputs.

Federated learning represents a paradigm shift in how ASR systems can be trained, offering substantial benefits in terms of privacy, security, and data diversity while posing new challenges in system design and security. As more devices become capable of performing complex computations and as concerns about data privacy grow, federated learning stands out as a promising approach to building next-generation ASR systems that are both powerful and user-centric.

5.2 Need for Federated Learning in ASR

Enhanced Privacy and Data Security: By allowing data to remain on local devices, federated learning significantly reduces the risk of data breaches and unauthorized access to sensitive voice recordings, addressing key privacy concerns for users.
Data Diversity: Federated learning benefits from diverse data sources since the model is trained across various devices in different environments. This diversity can lead to more robust and accurate ASR systems that perform well across different dialects, accents, and languages.

Reduced Latency and Network Load: Local data processing reduces the need for continuous internet connectivity and data transmission, decreasing latency in model responsiveness and reducing network infrastructure load.
Scalability: Federated learning scales efficiently as adding more clients to the network does not require proportionally more data transfer or central processing capacity.
Regulatory Compliance: FL aligns well with stringent data protection regulations such as the GDPR, as it minimizes the movement of personal data across borders and reduces the risk of non-compliance.

5.3 Implementation

Model Initialization the process begins with the central server initializing a global ASR model. This model is typically a deep neural network designed to handle the complexities of speech data, such as variations in accent, speech rate, and background noise.

Distributing the Model Instead of collecting user voice data, the initialized model is distributed to multiple client devices (e.g., smartphones, tablets, or other IoT devices). Each device acts as a node in the federated network.

Local Training Each client device uses onboard data to train the received model locally. This data could be from voice commands, dictations, or other audio inputs the

device captures during regular use. Importantly, this data never leaves the device, addressing significant privacy concerns associated with traditional ASR training methods.

Aggregating Updates after local training, each client computes model updates, typically gradients or model parameters, which are then securely sent back to the central server. Crucially, only these updates are transmitted, not the raw audio data.
Updating the Global Model: The server aggregates these updates from all participating clients using algorithms like Federated Averaging (FedAvg). This aggregation step updates the global model without ever having direct access to the individual data points.
Iteration: This cycle of distributing, local training, and updating is repeated multiple times, with the global model incrementally improving with each iteration.

5.4 Challenges

Despite its advantages , implementing federated learning in ASR systems comes with challenges. Handling non-IID data across devices can affect model convergence and performance. Communication efficiency is crucial, as the model updates must be synchronized across potentially thousands of devices. Security threats such as model poisoning and inference attacks must also be addressed to ensure the integrity and trustworthiness of the federated learning process.

The non-IID nature distributed data across devices can lead to challenges in model training, where some user data might lead the model to develop biases or underfit certain accents or speech patterns.
Communication Efficiency and Security: Ensuring that model updates are transmitted efficiently and securely remains a critical challenge, requiring robust encryption and potentially efficient networking protocols to handle thousands of devices.
System Complexity and Maintenance: Managing a federated learning system, particularly at scale, involves significant complexity in terms of system architecture, synchronization of model updates, and handling device variability in terms of computing power and availability.

CHAPTER 6

Experiments

6.1 Baseline

The baseline [11] Demonstrates an implementation of Automatic Speech Recognition (ASR) using the Whisper model from OpenAI. The Whisper model is designed for transcribing speech. Here's a detailed breakdown of how the ASR implementation is carried out: Model and Processor Initialization:
Whisper Model Components: The code uses WhisperProcessor, WhisperTokenizer, and WhisperFeatureExtractor from the transformers library. These components are crucial for preparing the audio data and processing the transcription outputs required by the Whisper model. Next is model loading, where the WhisperForConditionalGeneration model is loaded with a pre-trained version specific to English language tasks. This model will generate transcriptions based on the audio features processed by the Whisper components.

DataCollatorSpeechSeq2SeqWithPadding , is defined to handle the padding of input features and labels appropriately, ensuring that batched data fed into the model during training has uniform dimensions and properly masked labels for computing loss. The Word Error Rate metric is used to evaluate the model's performance. It measures the percentage of errors in the transcribed text compared to the ground truth, a standard metric for ASR systems. This is mentioned in [12]. The Seq2SeqTrainingArguments setup defines various training parameters such as batch size, learning rate, evaluation strategy, and hardware configuration (e.g., FP16 for faster training on compatible GPUs).

6.2 Implementations

Data Handling : The script processes audio data to convert it into a format suitable for training an ASR model. This includes converting Sphere format files (.sph) to WAV format using Sox, a powerful audio processing tool. Extracting features from the WAV files using the touch audio library, specifically computing Mel-Frequency Cepstral Coefficients (MFCCs), commonly used in ASR systems to capture the timbral aspects of the audio signal. A custom data collator is implemented to handle the padding of input features and labels appropriately for batch processing. This ensures that all batch data is the same size, which is critical for processing through deep learning models. The procedure for training an ASR model locally on each client's dataset. This includes defining the model architecture, setting up a training loop, and using a loss function suited for ASR tasks. The training arguments are configured using the transformers. Seq2SeqTrainingArguments, setting appropriate parameters such as batch size, learning rate, and evaluation strategy.

Includes functionality to visualize waveforms and MFCC features of audio files, which aids in qualitative analysis of the data being processed by the ASR system. This step is crucial for debugging and understanding the model's input and intermediate outputs. Implements the Federated Averaging algorithm, where each client's local updates (model weights) are averaged to update the global model. This step is crucial as it combines learning from diverse data distributions across multiple clients. After local training, model weights from each client are sent to a central server, aggregating them using the federated averaging method. The server then redistributes the updated global model to the clients for further training. It demonstrates how to randomly select a subset of clients for each training round, simulating a realistic federated learning environment where not all clients are available simultaneously.

Defines a structured flow where each client independently trains on their local data, contributing to the global model's improvement through iterative rounds of training and averaging. This decentralized approach maximizes data privacy and leverages local computational resources, making it ideal for sensitive applications like ASR. Training losses are plotted over multiple rounds of federated learning, offering a visual represen-

tation of the learning progression across federated rounds. This visualization helps identify trends such as improvements in model accuracy or potential issues like divergence or overfitting. The federated learning process ensures that the model progressively improves with each round of training. The convergence of the model is monitored through the average losses and WER, ensuring that the model is effectively learning from the distributed datasets.

The integration Federated learning in ASR training allows for effective model learning while preserving the privacy of the underlying data. By utilizing local computational resources and ensuring that only model updates are shared, the system adheres to privacy standards and reduces bandwidth usage. The results demonstrate the feasibility of using federated learning for complex tasks such as ASR, highlighting its potential in handling sensitive applications where data privacy is paramount. This approach enhances the learning process's scalability and contributes to more robust and generalized ASR systems.

CHAPTER 7

Results

7.1 FL with MDN

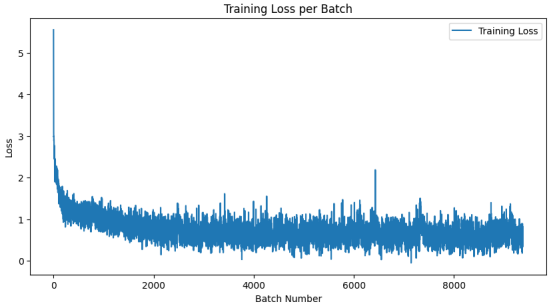

Figure 7.1: MDN for Mnist

Figure 6.1 shows how the loss decreases as the number of batches decreases for most datasets as a baseline plot for the MDN model. The plot is for loss vs number of batches.

Figure 6.2 shows how the loss decreases as the number of batches increases for the MDN model using the federated Averaging algorithm in federated settings for most datasets. The plot is for loss vs number of batches.

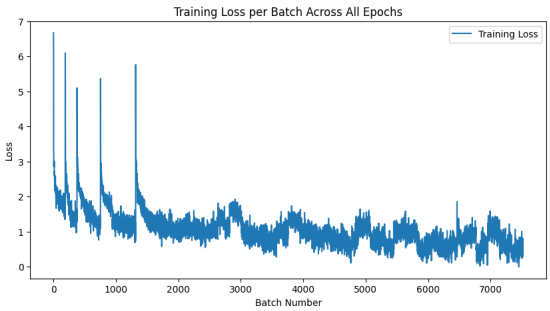

Figure 7.2: MDN using FedAvg for Mnist

Figure 6.3 shows how loss varies in a decreasing manner as the number of epochs decreases for MDN using the fedprox algorithm in federated settings for most datasets. The plot is for loss vs number of epochs.

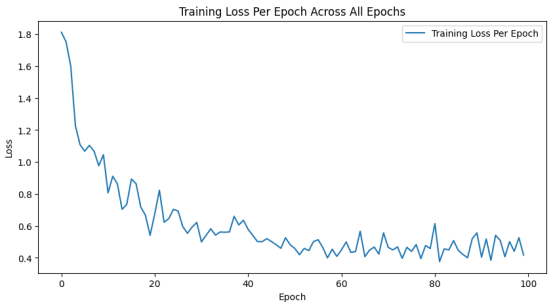

Figure 7.3: MDN using FedProx for Mnist

Figure 6.4 depicts the relationship between Log-Likelihood and the number of

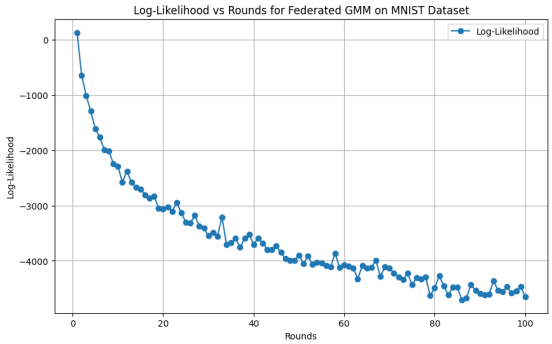

Figure 7.4: FL with GMM for Mnist

Rounds for a Federated GMM applied to the MNIST dataset; log-likelihood decreases (becomes more negative) with increasing rounds.

7.2 FL with ASR for An4

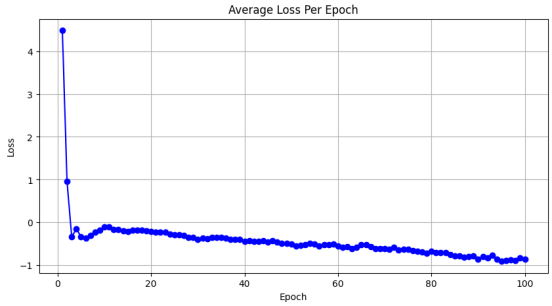

Figure 7.5: ASR with MDN

Figure 6.5 shows how loss varies in a decreasing manner as the number of epochs decreases for the mixture density network model using a federated averaging algorithm in federated settings for the ASR dataset. The plot is for loss vs number of epochs.

7.2.1 Results for ASR

The following table provides the results for several steps, training loss, validation loss, and Word Error Rate(WER), respectively, on the ASR dataset..

Step	Training Loss	Validation Loss	WER
1000	0.003300	1.392312	4.010349
2000	0.000100	1.421709	4.010349

CHAPTER 8

Conclusion

Future research on Mixture Density Networks (MDNs) within Federated Learning (FL) for Automatic Speech Recognition (ASR) should delve into enhancing robustness against Non-IID data by developing sophisticated aggregation methods and improving model personalization to adapt to individual linguistic nuances as mentioned in [20] [15] [1]. Exploring advanced federated learning architectures, including cross-device and cross-silo configurations and hybrid models that merge edge computing with cloud capabilities, could optimize resource usage and minimize latency while maintaining privacy. Furthermore, addressing vulnerabilities to adversarial attacks, devising privacy-compliant data augmentation methods, and creating standardized benchmarks for evaluating federated ASR systems across diverse datasets will be crucial. Theoretical studies [4] [19] to affirm the convergence properties of these federated models under varied data distributions will also be vital in ensuring the reliability and scalability of ASR applications in sensitive environments.

REFERENCES

[1] **Azam, S. S.**, **M. Pelikan**, **V. Feldman**, **K. Talwar**, **J. Silovsky**, and **T. Likhomanenko**, Federated learning for speech recognition: Revisiting current trends towards large-scale asr. *In International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*.

[2] **Bishop, C. M.** (). Mixture density networks.

[3] **Hard, A.**, **K. Rao**, **R. Mathews**, **S. Ramaswamy**, **F. Beaufays**, **S. Augenstein**, **H. Eichner**, **C. Kiddon**, and **D. Ramage** (). Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

[4] **Kairouz, P.**, **H. B. McMahan**, **B. Avent**, **A. Bellet**, **M. Bennis**, **A. N. Bhagoji**, **K. Bonawitz**, **Z. Charles**, **G. Cormode**, **R. Cummings**, *et al.* (). Advances and open problems in federated learning. *Foundations and trends® in machine learning*, **14**(1–2), 1–210.

[5] **Konečnỳ, J.**, **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*.

[6] **Konecnỳ, J.**, **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*, **8**.

[7] **Li, T.**, **A. K. Sahu**, **M. Zaheer**, **M. Sanjabi**, **A. Talwalkar**, and **V. Smith** (). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, **2**, 429–450.

[8] **Liu, J.**, **J. Huang**, **Y. Zhou**, **X. Li**, **S. Ji**, **H. Xiong**, and **D. Dou** (). From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems*, **64**(4), 885–917.

[9] **Mammen, P. M.** (). Federated learning: Opportunities and challenges. *arXiv preprint arXiv:2101.05428*.

[10] **McMahan, B.**, **E. Moore**, **D. Ramage**, **S. Hampson**, and **B. A. y Arcas**, Communication-efficient learning of deep networks from decentralized data. *In Artificial intelligence and statistics*. PMLR, .

[11] **McMahan, H. B.**, **E. Moore**, **D. Ramage**, **S. Hampson**, and **B. A. y Arcas** (). Communication-efficient learning of deep networks from decentralized data.

[12] **Nguyen, T.**, **S. Mdhaffar**, **N. Tomashenko**, **J.-F. Bonastre**, and **Y. Estève**, Federated learning for asr based on wav2vec 2.0. *In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, .

[13] **Nilsson, A.**, **S. Smith**, **G. Ulm**, **E. Gustavsson**, and **M. Jirstrand**, A performance evaluation of federated learning algorithms. *In Proceedings of the second workshop on distributed infrastructures for deep learning*.

[14] **Sun, T.**, **D. Li**, and **B. Wang** (). Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **45**(4), 4289–4301.

[15] **Tomashenko, N.**, **S. Mdhaffar**, **M. Tommasi**, **Y. Estève**, and **J.-F. Bonastre**, Privacy attacks for automatic speech recognition acoustic models in a federated learning framework. *In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, .

[16] **Xu, J.**, **B. S. Glicksberg**, **C. Su**, **P. Walker**, **J. Bian**, and **F. Wang** (). Federated learning for healthcare informatics. *Journal of healthcare informatics research*, **5**, 1–19.

[17] **Yu, W.**, **J. Freiwald**, **S. Tewes**, **F. Huennemeyer**, and **D. Kolossa**, Federated learning in asr: Not as easy as you think. *In Speech Communication; 14th ITG Conference*. VDE, .

[18] **Zhang, C.**, **Y. Xie**, **H. Bai**, **B. Yu**, **W. Li**, and **Y. Gao** (). A survey on federated learning. *Knowledge-Based Systems*, **216**, 106775.

[19] **Zhao, Y.**, **M. Li**, **L. Lai**, **N. Suda**, **D. Civin**, and **V. Chandra** (). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

[20] **Zhu, H.**, **J. Wang**, **G. Cheng**, **P. Zhang**, and **Y. Yan** (). Decoupled federated learning for asr with non-iid data. *arXiv preprint arXiv:2206.09102*.