Universidade Estadual de Campinas
Instituto de Computação

Rómulo Walter Condori Bustincio

Strategies for Reducing Communication Overhead in Federated Learning

Estratégias para Reduzir o Custo de Comunicação no Aprendizado Federado

CAMPINAS
2024

Rómulo Walter Condori Bustincio

Strategies for Reducing Communication Overhead in Federated Learning

Estratégias para Reduzir o Custo de Comunicação no Aprendizado Federado

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Luiz Fernando Bittencourt
Co-supervisor/Coorientador: Prof. Dr. Allan Mariano de Souza

Este exemplar corresponde à versão final da Dissertação defendida por Rómulo Walter Condori Bustincio e orientada pelo Prof. Dr. Luiz Fernando Bittencourt.

CAMPINAS
2024

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

C754s
Condori Bustincio, Rómulo Walter, 1988-
Strategies for reducing communication overhead in federated learning / Rómulo Walter Condori Bustincio. – Campinas, SP : [s.n.], 2024.

Orientador(es): Luiz Fernando Bittencourt.
Coorientador(es): Allan Mariano de Souza.
Dissertação (mestrado) – Universidade Estadual de Campinas (UNICAMP), Instituto de Computação.

1. Aprendizado federado. 2. Computação de borda. 3. Computação e processamento de informação. I. Bittencourt, Luiz Fernando, 1981-. II. Souza, Allan Mariano de, 1992-. III. Universidade Estadual de Campinas (UNICAMP). Instituto de Computação. IV. Título.

Informações complementares

Título em outro idioma: Estratégias para reduzir o custo de comunicação no aprendizado federado
Palavras-chave em inglês:
Federated learning
Edge computing
Computing and information processing
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Luiz Fernando Bittencourt [Orientador]
Edson Borin
Júlio Cezar Estrella
Data de defesa: 06-08-2024
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-3971-5105
- Currículo Lattes do autor: http://lattes.cnpq.br/8741962345946039

Universidade Estadual de Campinas
Instituto de Computação

Rómulo Walter Condori Bustincio

Strategies for Reducing Communication Overhead in Federated Learning

Estratégias para Reduzir o Custo de Comunicação no Aprendizado Federado

Banca Examinadora:

- Prof. Dr. Luiz Fernando Bittencourt
  Instituto de Computação - UNICAMP
- Prof. Dr. Edson Borin
  Instituto de Computação - UNICAMP
- Prof. Dr. Júlio Cezar Estrella
  Instituto de Ciências Matemáticas e de Computação - USP

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 06 de agosto de 2024

The best way to predict the future
is to invent it.

(Alan Kay)

Agradecimentos

Nesta importante etapa da minha trajetória acadêmica, desejo expressar minha sincera gratidão àqueles que contribuíram significativamente para o desenvolvimento desta pesquisa. Um reconhecimento especial ao Professor Luiz Bittencourt e ao Professor Allan Souza, cuja sabedoria e orientação foram fundamentais na minha formação e no avanço deste estudo.

Meu agradecimento também se estende ao HIAAC, cujo apoio e recursos foram cruciais para a consecução dos resultados aqui apresentados. Ao Joahannes Costa, pelo seu apoio e contribuição nos artigos que foram produzidos nesta pesquisa.

Em nível pessoal, devo um eterno agradecimento à minha família. Ao meu pai, que, embora já não esteja conosco, continua sendo uma fonte de inspiração e aprendizado. À minha mãe, cuja força e amor têm sido o suporte constante no meu caminho para a realização dos meus sonhos. À minha irmã e sobrinhos, por sua inabalável alegria e apoio. E à minha namorada Eloisa, cuja compreensão, amor e motivação foram essenciais nos momentos mais desafiadores desta jornada.

Este trabalho é o resultado de um esforço coletivo, e cada um de vocês desempenhou um papel insubstituível nele. Obrigado pelo seu apoio inestimável.

Resumo

No rápido avanço do cloud computing, numerosos desafios surgiram, incluindo os custos de comunicação e o overhead de transferir informações de um host para outro, conhecido como overhead de comunicação. Para mitigar esses problemas e reduzir os custos de comunicação, o edge computing foi desenvolvido, distribuindo o processamento para dispositivos finais ou edge. Com o avanço do hardware, começou o desenvolvimento de machine learning em grande escala, unindo o poder de recursos computacionais em dispositivos terminais com a necessidade de treinar modelos de aprendizagem de máquina usando dados sensíveis ou privados. Essa convergência levou ao desenvolvimento da aprendizagem federada, uma estratégia robusta para treinar modelos de aprendizagem de máquina de maneira distribuída, focando na proteção de dados. Neste contexto, o problema do overhead de comunicação emerge como um problema crítico que precisa ser abordado para melhorar e sustentar modelos e infraestruturas eficientes. Esta dissertação de mestrado foca em abordar o overhead de comunicação na aprendizagem federada, desenvolvendo uma taxonomia para identificar as abordagens atualmente utilizadas na literatura para resolver esse problema. Na segunda parte, propõe um algoritmo de seleção de clientes visando reduzir os custos de comunicação baseado em métricas de adequação para escolher dispositivos ou clientes para treinamento. Por último, introduz um algoritmo de poda de esparsidade para reduzir o número de parâmetros ou pesos em uma rede neural a ser compartilhada com o servidor, focando na eliminação de pesos que podem ser considerados irrelevantes. Todas essas contribuições visam abordar de maneira sistemática o problema descrito.

Abstract

With the rapid advancement of cloud computing, various challenges have emerged, including the costs associated with communication and the transfer of information from one host to another, a process known as communication overhead. To alleviate these issues and reduce communication costs, edge computing has been developed, distributing processing to end devices or the edge. Furthermore, advancements in hardware have enabled the development of large-scale machine learning. This environment of computational power at the terminal devices, combined with the need to train machine learning models with sensitive or private data, has led to the emergence of federated learning. Federated learning is an effective strategy for training machine learning models in a distributed manner with a focus on data protection. Within this context, communication overhead emerges as a significant challenge that must be addressed to improve and maintain efficient and sustainable models and infrastructure. This master's dissertation focuses on tackling the issue of communication overhead in federated learning by developing a taxonomy to identify approaches currently being utilized in the literature. Additionally, it proposes a client selection algorithm aimed at reducing communication costs based on suitability metrics for choosing appropriate devices or clients for training. Finally, it introduces a sparsity pruning algorithm to decrease the amount of parameters or weights in a neural network that are shared with the server, focusing on reducing or nullifying weights that may be considered irrelevant. All these contributions are aimed at systematically addressing the aforementioned problem.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In recent years, the increasing reliance on mobile and edge devices has sparked a significant interest in deploying Machine Learning (ML) directly on these platforms [46, 72]. This shift aims to cater to a growing need for AI applications that are both personalized and capable of delivering immediate responses. In response to this trend, Federated Learning (FL) has emerged as a pioneering approach [34] that bases the interaction between the client and the server on a communication mechanism. It facilitates a distributed form of machine learning by leveraging a potentially vast network of edge devices, also known as clients, to perform collaborative computation tasks. This model is especially noteworthy as it obviates the need for aggregating training data in a central repository, thereby addressing privacy and data centralization concerns.

One significant challenge in FL is the communication overhead, which arises from the need to transmit model updates instead of raw data due to privacy concerns and the sheer scale of devices involved [12]. As highlighted by Li*et al.* [42], the communication between the server and potentially millions of devices can be much slower than local computation, making it essential to develop methods that either reduce the number of communication rounds or decrease the size of the messages sent during each round. Essentially, communication involves transmitting information from one point to another, encompassing steps like message creation, symbol representation, encoding for transmission, and delivery to the recipient, where it is decoded and interpreted. However, in FL, this process can introduce quality degradation due to system imperfections, affecting model performance [53, 42, 32]. Therefore, enhancing communication efficiency emerges as a critical objective to mitigate performance issues and improve the overall effectiveness of FL systems. Various strategies have been identified in the literature to enhance communication efficiency in FL scenarios; according to Shahid *et al.* [62], we have the following strategies.

- **Local Update strategy** is notable for its emphasis on minimizing necessary communication. It enables devices to perform extensive calculations locally before transmitting updates to the central server, significantly curtailing network traffic and enhancing the overall efficiency of the FL system [42].

- **Client Selection** involves selecting a specific subset of devices for participation in each training round, rather than engaging all available devices. The selection

criteria—device availability, data quality, and computing capacity—optimize communication efficiency and foster model convergence [38, 21, 19].

- **Model Update Reduction strategy** focuses on techniques designed to reduce the frequency and size of model updates exchanged between devices and the server. Employing data compression techniques, quantization, and efficient coding schemes achieves this goal, enhancing communication efficiency [62].

- **Decentralized Training and Peer to Peer Learning** approaches mitigate the dependency on a central server by facilitating direct communication among devices for exchanging model updates. Peer learning is particularly effective in distributing the workload uniformly and utilizing local network connections to enhance efficiency [42].

- **Compression Schemes**, including advanced quantization, sparsification, and coding techniques, is crucial. These methods aim to reduce the size of model updates before their transmission, managing bandwidth limitations and augmenting the scalability of FL [62].

1.1 Motivation

FL offers a promising approach to decentralized ML while addressing privacy concerns. Improving model performance and managing low-quality data are common challenges across ML. However, communication efficiency is a unique concern for FL due to its reliance on data exchange for maintaining privacy. In scenarios where FL replaces centralized data servers, communication must be exceptionally efficient to facilitate this exchange. Therefore, enhancing communication efficiency emerges as a pivotal challenge, distinguishing FL research within the broader ML discipline [3]. However, its widespread adoption faces a critical barrier in communication overhead. The considerable exchange of parameters between devices and the server in FL introduces substantial communication overhead, recognized as a significant impediment [82].

This overhead, primarily due to the transfer of vast data volumes, escalates communication costs [4]. To mitigate this challenge, innovative solutions such as **client selection** and **model pruning** have emerged as pivotal strategies. Client selection determines which clients should participate in the training round based on various criteria, including the performance of the model [67], data relevance, availability [38], and device capabilities [49, 48]. This selective approach aims to optimize the training process by involving only the most contributory clients, thereby reducing unnecessary communication and enhancing system efficiency [35]. Concurrently, strategies for parameter reduction, particularly **model pruning**, have gained attention. Model pruning involves systematically removing non-critical parameters from the model without significantly compromising its accuracy. This process effectively reduces the size of the models to be communicated, further curtailing the communication overhead. These approaches mark crucial progress in curtailing communication overhead, thus enhancing FL's practicality in diverse applications [35, 52].

1.2 Research Goals

The main objective of this dissertation is to address the communication challenges in FL, considering non-IID scenarios, where data distribution differs among clients. To achieve this, the following specific objectives have been established:

1. **Contributing to FL Literature:** Enrich the field with new perspectives and solutions, based on empirical findings from conducted studies.
2. **Enhancing Communication Efficiency in FL:** Focus on improving communication efficiency in federated learning through a novel client selection mechanism based on data entropy and model divergence.
3. **Implementing and Assessing Single-Stage Model Pruning Techniques:** Develop and evaluate a single-stage model pruning method to reduce communication overhead while maintaining accuracy in federated learning.

1.3 Problem Statement

In the realm of FL, the core challenges revolve around optimizing communication efficiency and model accuracy, while safeguarding data privacy. Despite FL's promise for decentralized machine learning, its practical implementation faces significant hurdles due to the high communication overhead and challenges in managing heterogeneous data distributions across devices. This research addresses these crucial issues by developing innovative approaches for client selection and model pruning, aiming to enhance the overall efficiency and effectiveness of FL systems.

1.4 Organization of the Dissertation

This dissertation is structured as follows: In Chapter 1, we provide an overview of Federated Learning, highlighting its importance and the specific challenges this research aims to address. Chapter 2 discusses foundational concepts and terminologies in FL, setting the stage for a deeper exploration of the subject matter. In Chapter 3, we review existing literature on FL, with a focus on communication efficiency and data privacy, outlining the state of the art and identifying gaps this research intends to fill. Chapter 4 details the novel strategies proposed in this dissertation, including their theoretical foundation, development process, and potential impact on FL. Chapter 5 describes the strategies developed to tackle communication overhead in Federated Learning, along with the simulation settings used for evaluation. In Chapter 6, we present the findings from the implementation of the proposed strategies, offering a critical analysis and discussion on their effectiveness, implications, and areas for improvement. Finally, Chapter 7 presents the conclusions, describes future work, and identifies potential research opportunities provided by the studies presented in this dissertation.

Chapter 2

Definitions and Basic Concepts

This chapter defines key concepts and models crucial to this dissertation. At the core is Federated Learning (FL), a model that allows devices to collaboratively learn a model without centralizing data, thus preserving privacy. We also explore Information Theory to understand and optimize information flow in FL. Additionally, we discuss Neural Network Pruning, a technique aimed at reducing neural network complexity for FL by minimizing the parameters exchanged between clients and servers, enhancing efficiency.

2.1 Mathematical Concepts

Mathematics provides the foundation for many techniques and theories in science and engineering. Among these, norms and metrics stand out as essential tools for quantifying and analyzing the properties of mathematical objects.

2.1.1 Norms

A norm is a function that assigns a positive length or size to vectors in a vector space, except for the zero vector, which is assigned a length of zero. Norms are denoted as $\|\cdot\|$ and are used to measure the magnitude of vectors, serving as a foundational tool in optimization problems, regularization techniques, and error measurement [58].

Definition

Formally, a norm on a vector space V over a field F is a function $\|\cdot\| : V \to \mathbb{R}$ that satisfies the following properties for all $u, v \in V$ and all scalars $a \in F$:

1. **Non-negativity**: $\|v\| \geq 0$ with equality if and only if $v = 0$.
2. **Scalar multiplication**: $\|av\| = |a|\|v\|$.
3. **Triangle inequality**: $\|u + v\| \leq \|u\| + \|v\|$.

Common Norms

Norms are integral to multiple facets of Machine Learning, encompassing regularization, optimization, and the assessment of model performance. The principal norms utilized in this context include the following:

L1 Norm (Manhattan Distance) The L1 norm of a vector $v \in \mathbb{R}^n$ is defined as:

$$\|v\|_1 = \sum_{i=1}^{n} |v_i|,$$

where v_i is the i-th component of v. This norm is also known as the Manhattan distance.

L2 Norm (Euclidean Distance) The L2 norm, or Euclidean norm, of a vector v is the most familiar:

$$\|v\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

This norm measures the "straight-line" distance between the origin and the point represented by v.

Infinity Norm (Maximum Norm) The infinity norm of a vector v is defined as:

$$\|v\|_\infty = \max_{1 \le i \le n} |v_i|.$$

It represents the maximum absolute value of the components of v.

2.1.2 Methods for Optimizing

Optimization lies at the heart of machine learning, providing the essential algorithms that enable models to learn from data[6]. Optimization techniques guide the model towards minimizing a loss function (or maximizing an objective function) by iteratively adjusting model parameters. A primary component of these methods is the learning rate. This tuning parameter determines the step size at each iteration while moving toward a minimum of a loss function. This function maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event [16].

- **Gradient Descent** is a cornerstone optimization technique that iteratively adjusts parameters to minimize a given objective function. For a function $f(\theta)$, where θ represents the parameters of the model, gradient descent updates θ by moving in the direction of the negative gradient of the function at the current point:

  $$\theta_{\text{new}} = \theta_{\text{old}} - \alpha \nabla f(\theta_{\text{old}}),$$

  where α is the learning rate, controlling the size of the steps taken towards the minimum [57].

- **Stochastic Gradient Descent (SGD)** modifies the traditional gradient descent approach by updating parameters using the gradient of a single sample or a small batch of samples. This stochastic nature significantly reduces computational costs and can help in avoiding local minima, making SGD particularly effective for large-scale data sets common in machine learning [9].

While these methods are widely used in machine learning algorithms, other techniques like Adaptive Moment Estimation (Adam), an advanced version of SGD that introduces an adaptive learning rate for each parameter, can further improve the efficiency and effectiveness of the training process [68].

2.2 Fundamentals of Machine Learning

Machine Learning (ML) is the foundation of modern Artificial Intelligence (AI) systems, empowering computers to learn from data, improve through experience, and make decisions based on vast amounts of information. At its core, ML seeks to develop algorithms that can identify patterns, make predictions, or take actions based on input data, without being explicitly programmed to perform the task [22]. These capabilities are realized through various learning paradigms and algorithms, which will be discussed in the following section.

2.2.1 Learning Paradigms and Algorithms

ML offers a wide range of tools for extracting patterns and insights from data. These tools are categorized into several core learning paradigms. Traditionally, these paradigms are divided into supervised, unsupervised, and reinforcement learning [20]. Each of these paradigms plays a distinct role in addressing different types of problems and employs specific algorithms to achieve their goals, as detailed below:

- **Supervised learning** is a machine learning paradigm in which a model is trained on input data along with corresponding target outputs. This allows the model to learn the relationship between the inputs and outputs, enabling it to make predictions on new unseen data. Classification problems, where the objective is to assign input data to discrete categories, and regression problems, where the goal is to predict continuous variables, are examples of supervised learning tasks [8].

- **Unsupervised learning** Unlike supervised learning, unsupervised learning involves training models on data without labeled responses. The goal is to discover inherent patterns or groupings in the data, which can be useful for clustering, dimensionality reduction, or associative tasks. Techniques such as k-means clustering and principal component analysis (PCA) are common examples [28].

Table 2.1 presents a comparison between supervised and unsupervised learning, highlighting their distinct characteristics and applications. Beyond these paradigms, the literature recognizes additional approaches such as Reinforcement Learning [69] and Semi-Supervised Learning [14]. These methodologies extend the spectrum of machine learning

Table 2.1: Comparison of supervised and unsupervised learning.

Feature	Description
Supervised Learning	
Goal	Learn from labeled data to predict or classify new data.
Data Type	Requires datasets with labeled inputs.
Tasks	Classification: Assign data to predefined categories. Regression: Predict continuous values.
Examples	Digit recognition, spam detection, stock price prediction.
Advantages	High accuracy for well-defined tasks.
Challenges	Requires manual data labeling, which can be costly and time-consuming.
Common Algorithms	Linear regression, logistic regression, decision trees, SVM, neural networks, Convolutional Neural Networks(CNN) .
Unsupervised Learning	
Goal	Discover patterns and structures in unlabeled data.
Data Type	Works with unlabeled datasets.
Tasks	Clustering: Group similar data. Dimensionality Reduction: Decrease the complexity of the data space. Density Estimation: Model the probability distribution of data.
Examples	Customer segmentation, social network analysis, anomaly detection.
Advantages	Can discover unexpected patterns.
Challenges	Can be difficult to interpret results and determine model quality.
Common Algorithms	K-means, PCA, independent component analysis (ICA), hierarchical agglomerative clustering (HAC).

by incorporating both the exploitation of labeled data and the exploration of data-driven learning where explicit labels may not be extensively available.

2.2.2 Optimizing Model Performance

Loss functions like Mean Squared Error, Cross-Entropy Loss, and Hinge Loss guide training. Performance is measured via metrics such as accuracy and MSE. **Data preprocessing** and **hyperparameter tuning** (*e.g.*, batch size, learning rate) are critical for model accuracy and efficiency.

2.3 Federated Learning

FL represents a paradigm shift in machine learning, emphasizing a decentralized and collaborative approach. In FL, rather than collecting data and training on a central server, the training process is distributed across multiple devices or nodes, each with its own data. This methodology not only preserves data privacy but also leverages the computational resources of the participating nodes, becoming a powerful solution for scenarios where data centralization is impractical or undesirable.

Within the scope of FL, a formal mathematical representation of model training can

be formulated as follows: *(i)* Consider a set $C = \{c_1, c_2, \ldots, c_D\}$, where each client c_k in the FL system has a local dataset D_k; *(ii)* The goal in an FL scenario is to minimize a global objective function, which is a composition of the loss functions calculated by each client based on their private data. This composition can be mathematically represented as:

$$\min_{x \in \mathbb{R}^t} f(x) = \frac{1}{D} \sum_{k=1}^{D} f_k(x; D_k) \tag{2.1}$$

where $f_k(x; D_k)$ denotes the loss value calculated by client k based on their local dataset D_k.

As can be observed, this representation reinforces the decentralized and collaborative nature of FL, where each client independently calculates the loss based on their data and contributes to the training of the global model.

2.3.1 Federated Learning Model

FL models usually necessitate an initial training process, typically encompassing the primary steps, as depicted in Figure 2.1. Consider, furthermore, that for each trained model, whether at the local or global level, there are associated weights. This arises from the fact that they are machine learning models.

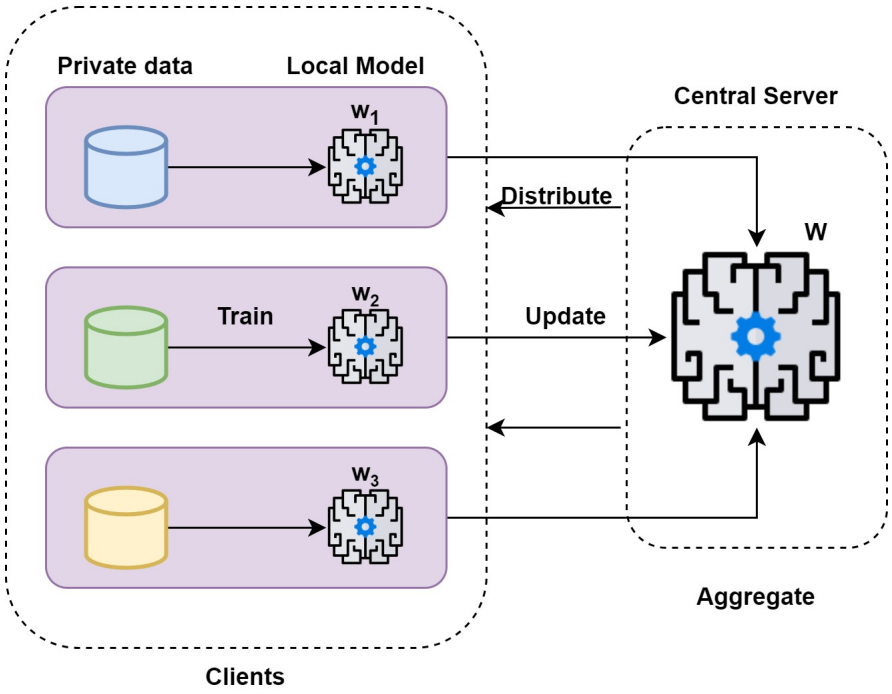


Figure 2.1: Visual representation of a typical FL training process.

Figure 2.1 depicts the standard aggregation process employed in federated learning, wherein client cooperation and server communication are established via the Federated Learning Model. The combination of data and models that make up the isolated system ensures data privacy preservation, as communication is limited to the distribution and uploading of model updates. In each isolated system, each local model is characterized

by its associated weights w. When updates are received, the server uses an aggregation algorithm to obtain a global model W distributed among the clients.

2.3.2 The Federated Learning Training Process

The FL training process, as introduced by McMahan *et al.* [52], embodies a novel approach to machine learning. This methodology preserves data privacy while leveraging distributed data sources for model training. Unlike traditional centralized machine learning methods, FL enables a model to learn from diverse datasets located on various devices without the need to aggregate data in a central repository.

Initialization and Client Selection

The process begins with the server initializing a global model. This model is then distributed to a subset of selected clients. The selection is based on criteria such as data availability, device capability, and previous contributions to the learning process. This phase ensures that only the most suitable clients participate in the training process, optimizing both the efficiency and effectiveness of the learning.

Local Model Training

Upon receiving the global model, each selected client trains the model on its local dataset. This step leverages the principle of data locality, which refers to processing data close to where it is generated or stored, ensuring that personal or sensitive data never leaves the device, thus significantly enhancing privacy and security

Model Updates Aggregation

After local training, clients send their model updates, typically in the form of gradients or model weights, back to the server. The server aggregates these updates to improve the global model. The Federated Averaging (FedAvg) algorithm, a common method for aggregation, computes a weighted average of these updates, facilitating the synthesis of a more robust and accurate global model.

Iteration and Convergence

The cycle of distributing the global model, conducting local training by clients, and aggregating updates is repeated iteratively. With each iteration, the global model's accuracy and performance are enhanced. The process continues until the model converges, achieving or surpassing predefined performance metrics.

2.3.3 Key Concepts for Communication Optimization in FL

In this part, we define the key concepts that are crucial for understanding communication overhead reduction strategies in Fl.

- **Compression**: The process of reducing the size of data or model updates sent from clients to the server in order to minimize communication costs.
- **Local Updates**: A strategy where clients perform several training iterations on local data before communicating with the server, thus reducing the frequency of updates and overall communication.
- **Adaptive Sparsification**: Selecting and transmitting only the most important or relevant model parameters from the client to the server, reducing the volume of data exchanged.
- **Adaptive Sampling**: Dynamically selecting a subset of data or updates based on their importance or relevance, with the goal of minimizing the amount of transmitted information.
- **Domain Adaptation**: Adjusting models to align with new client data distributions, reducing the need for continuous communication between clients and the server.
- **Clustering Data or Clients**: Grouping similar data or clients to optimize communication, allowing updates to be shared within clusters rather than across the entire network.
- **Knowledge Distillation**: A technique for transferring knowledge from a larger model to a smaller one, thereby reducing the size of updates that need to be communicated between clients and the server.
- **Reduce Model Size**: Shrinking the number of model parameters in order to decrease the amount of information exchanged during communication rounds.
- **Model Aggregation**: The process of combining model updates from multiple clients into a single global model at the server, which helps to minimize the total communication required.
- **Adaptive Scheduling**: Adjusting the timing or frequency of communication between clients and the server based on network conditions or model performance, to improve communication efficiency.

2.4 Information Theory

The concept of Information Theory, introduced by Claude Shannon in 1948, stands as a cornerstone in the realms of signal processing, telecommunications, and computer science [64]. Shannon's groundbreaking work laid the foundation for understanding how information is measured, transmitted, and encoded efficiently.

Information, in the context of Information Theory, is essentially a measure of one's uncertainty about an event. The more uncertain or surprised one is about an event, the more information that event is said to contain. Entropy, denoted as $H(X)$ for a

random variable X, quantifies this uncertainty. It provides a limit on how much a message can be compressed without loss of information, embodying the essence of information efficiency [18].

In the domain of ML, the principles of Information theory find extensive application, guiding the design and evaluation of algorithms [50].

2.4.1 Applications in Machine Learning

In ML, Information Theory supports many algorithmic strategies and evaluation metrics. Entropy, for instance, is used in decision tree algorithms to determine the best splits that will maximize information gain - effectively reducing uncertainty in the dataset's classification [10]. Similarly, information mutual between variables indicates how much knowledge the presence of one variable contributes to understanding another, guiding feature selection and model complexity [5]. The principles of Information Theory, as outlined by MacKay [50], find extensive application in guiding the design and evaluation of machine learning algorithms, highlighting the interplay between entropy, information gain, and mutual information in the development of efficient and effective models.

2.4.2 Information Theory in FL

FL, with its decentralized approach to model training, places a premium on the efficient transmission and encoding of information. The challenge of minimizing communication costs while maintaining model accuracy invokes the principles of Information Theory. In this setting, strategies to reduce the entropy of model updates or to maximize the information mutual between local and global models can significantly impact the efficiency and effectiveness of the learning process [34, 52]. Understanding the theoretical limits of information transmission helps in designing more efficient federated learning systems, where the goal is to transmit as much relevant information as possible using the least amount of communication overhead [35]. The work by Konečnỳ et al. and Smith et al. illustrates the application of these principles, demonstrating how optimizing the flow of information between devices and servers is crucial for the scalability and viability of federated learning models.

2.4.3 Shannon Entropy

Shannon entropy plays a key role in quantifying uncertainty and characterizing the average information content of a data source, as discussed by Orlandi *et al.* [55]. The Shannon entropy is computed as follows.

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \tag{2.2}$$

where:

$H(X)$ represents the entropy of the random variable X.
$P(x_i)$ is the probability of event x_i.

Clients with high-entropy datasets have the potential to enhance the federated learning model's performance by providing diverse and informative data.

2.5 Optimizing Client Selection in FL

In FL, the process of selecting clients is crucial for determining the participants in the model's distributed training phase [52]. This selection is pivotal for enhancing federated learning's overall efficiency and effectiveness, primarily by reducing communication overhead and safeguarding the global model's integrity [32]. Optimal client selection strategies meticulously evaluate factors such as the diversity of data across devices, their computational power, and the relevance of their data to the specific learning objective at hand.

2.5.1 Factors Influencing Optimal Client Selection

Optimizing client selection involves balancing several critical factors:

- Data Heterogeneity: Clients often hold data that is non-IID (non-independent and identically distributed), meaning the data across clients can vary significantly. Ensuring that selected clients represent a diverse range of data types is important for reducing bias and improving the model's ability to generalize to unseen data.
- Computational Resources: The computational power of client devices can differ widely. Selecting clients with sufficient computational power ensures that training tasks are completed in a timely manner without causing delays due to slow devices.
- Communication Capabilities: Clients with poor network conditions can introduce significant delays in the aggregation process. Optimizing for clients with stable and high-bandwidth connections helps reduce communication overhead and accelerates the training rounds.

2.6 Pruning Techniques in Machine Learning

Model pruning is a critical technique in deep learning aimed at reducing model complexity and enhancing computational efficiency. This section explores various pruning methodologies and their implications, and applications within FL environments. Pruning refers to the process of removing unnecessary parameters from a neural network without significantly affecting its performance. As highlighted by Torsten Hoefler *et al.* , the integration of pruning techniques can significantly enhance the performance of neural networks by reducing the redundancy in the model parameters without substantially sacrificing accuracy [26]. In the literature focusing on the training process, we find:

- **One-Shot Pruning** involves removing parameters at a single instance before the training process begins, based on certain criteria like weight magnitudes.
- **Iterative Pruning** contrasts with one-shot by gradually removing parameters across several training cycles, allowing the model to adapt and potentially recover performance between pruning steps.

Moreover, **Fine-tuning** refers to the process of retraining a pruned model to regain or improve its accuracy by adjusting the remaining parameters with continued training. In one-shot pruning, this process is often unnecessary.

2.6.1 Structured vs. Unstructured Pruning

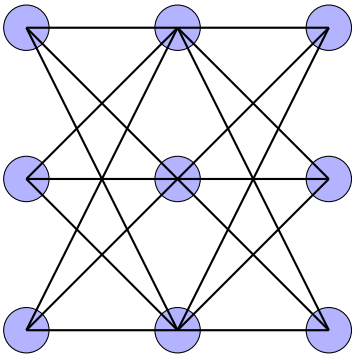
- **Unstructured Pruning** targets individual weights across the network, leading to sparse matrices without altering the network architecture.
- **Structured Pruning** removes entire channels or filters, potentially yielding models that are more hardware-friendly due to the reduction in model dimensions

Table 2.2: Comparative between structured and unstructured pruning.

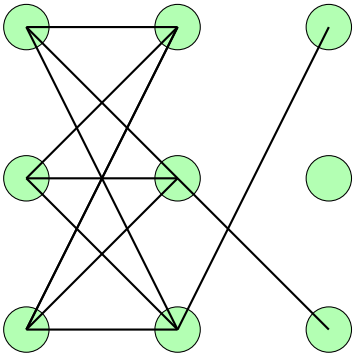
Aspect	**Structured Pruning**	**Unstructured Pruning**
Target	Channels/filters	Individual weights
Matrix	Dense matrices	Sparse matrices
Hardware	More friendly	Less friendly
Performance	May alter significantly	Minor changes
Complexity Reduction	High	Variable
Recovery	Requires retraining	Easier to recover

Table 2.2 summarizes the main differences between structured and unstructured pruning techniques, highlighting their impact on model performance, hardware compatibility, and complexity reduction.

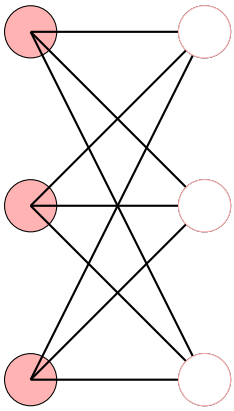
Figure 2.2 compares the original network, non-structural pruning, and structural pruning techniques. In Figure 2.2(a), we illustrate the original neural network, where all neurons are connected in a fully dense manner. This serves as the baseline for comparing the pruning methods. Figure 2.2(b) demonstrates non-structural pruning. In this approach, individual connections are removed between neurons, while the overall structure of the network remains intact. The random removal of some connections reduces the model's complexity without drastically altering its architecture. In contrast, Figure 2.2(c) shows structural pruning. This method involves the removal of entire neurons, along with their associated connections. This significantly changes the architecture by reducing the number of active neurons, resulting in a more compact network.



(a) Original network



(b) Non-structural pruning



(c) structural pruning

Figure 2.2: Original network, non-structural pruning, and structural pruning

2.6.2 Model Pruning in Federated Learning

In FL, model pruning techniques are essential for adapting to decentralized data. These techniques not only aim to compress and personalize models but also reduce communication and computational costs, thereby enhancing communication efficiency. Crucially, they maintain model accuracy across distributed devices while preserving data privacy. Such strategies are specifically designed to operate efficiently within FL scenarios [30, 31].

Chapter 3

Related Work

This chapter reviews existing research on communication overhead in FL, adopting a comprehensive approach to understanding its impact and mitigation strategies. It sets the stage for the innovative solutions introduced in chapter 4. The structure of this chapter is organized around key issues:

- Communication overhead in FL from a general perspective.
- The role of client selection in minimizing communication overhead.
- Strategies for model pruning and masking to alleviate communication overhead in FL.

3.1 Perspectives on Communication Overhead in FL: A Review

The literature on FL addresses communication overhead from various perspectives. However, much of the existing work tends to focus on specific solutions rather than providing a comprehensive analysis of the underlying problem. In this section, we take a more fundamental approach, focusing on understanding the root causes and implications of communication overhead in FL systems, without being limited to particular strategies or algorithms. Table 3.1 compiles research that explicitly addresses or mentions communication overhead, highlighting the attention given to this issue rather than the solutions proposed. Our goal is to provide a clearer characterization of the problem itself, laying the groundwork for the development of strategies aimed at addressing communication challenges in FL.

Table 3.1: Works focused on communication overhead in FL.

Paper	**Key Focus**	**Contribution**	**Relation to Communication Overhead**
Abdulrahman *et al.* (2021) [2]	Overview of federated learning	Proposes new taxonomies for FL	Explores FL broadly, with a particular emphasis on delineating communication challenges.
Guendouzi *et al.* (2023) [23]	Challenges, aggregation methods	Systematic review of FL challenges	Highlights the significance of communication overhead in the broader spectrum of FL challenges.
Almanifi *et al.* (2023) [3]	Communication and computation efficiency	Strategies for enhancing efficiency in FL	Specifically targets improving communication efficiency among FL devices.
Wang *et al.* (2023) [74]	Communication compression techniques	Reviews compression strategies for distributed learning	Focuses on reducing the data transmitted to mitigate communication overhead.
Janbi *et al.* (2023) [29]	Distributed AI taxonomy	Framework for provisioning DAI services	Discusses communication overhead within the broader context of DAI provisioning.
Pfeiffer *et al.* (2023) [56]	FL for constrained devices	Surveys FL implementation in heterogeneous devices	Considers communication overhead as part of the challenges in implementing FL on diverse devices.
Ye *et al.* (2023) [76]	Heterogeneity in FL	Reviews solutions for HFL challenges	Addresses communication strategies within heterogeneous FL environments.
Soni *et al.* (2022) [66]	ML in cloud computing paradigms	Taxonomy and review of ML techniques	Indirectly touches on communication overhead through the lens of cloud computing.
Sabah *et al.* (2024) [59]	Model optimization in PFL	Survey on optimization techniques in PFL	Focuses on model optimization as a means to reduce communication overhead.

The works compiled in Table 3.1 reflect a diverse range of approaches to addressing communication overhead in FL. While many studies, such as [2] and [23], provide broad overviews and taxonomies of FL, they also emphasize the significance of communication challenges in distributed systems. More focused contributions, like those in [3] and [74], specifically target communication efficiency, either through enhanced strategies

or compression techniques, which are essential for reducing the data transmitted between devices.

Several works, including [29] and [56], consider communication overhead within the context of constrained and heterogeneous environments, highlighting the importance of optimizing communication in scenarios where device resources are limited. Moreover, studies like [76] delve deeper into how heterogeneity in data and devices impacts communication strategies, showing that addressing non-uniform client participation remains a critical issue.

The review of the literature indicates that while communication overhead is widely recognized as a fundamental challenge in FL, the specific strategies to tackle this problem vary significantly. There is a clear need for solutions that are tailored to specific environments, whether focused on constrained devices, heterogeneous data, or system-wide optimizations, as different approaches may be required depending on the nature of the FL system.

3.2 Client Selection Strategies

This section explores various strategies for selecting clients in federated learning environments. Effective client selection is crucial for optimizing learning outcomes, reducing communication overhead, and ensuring the scalability of federated learning systems. We review criteria and algorithms proposed for selecting clients that contribute most effectively to the learning process, taking into consideration factors such as data quality, availability, and computational capabilities.

A notable approach is CMFL (Communication-Mitigated Federated Learning) introduced by Wang *et al.* [72]. CMFL employs sign-based metrics to determine whether to transmit local updates to the server, contingent on a threshold. This threshold is reliant on data distribution and necessitates an exploratory investigation. CMFL also guarantees convergence. In contrast, EntropicFL dynamically computes the threshold in each round using divergence weights between local and global models, allowing each client to determine whether to send updates to the server, consequently reducing the communication overhead.

Zhao *et al.* [81], weight divergence in non-IID data is examined, revealing higher divergence compared to IID data. They propose enhancing accuracy in non-IID data by establishing a shared, small global dataset among clients. Our research leverages model divergence to detect irrelevant client updates.

The FedMCCS framework, introduced by Abdulrahman et al. in their 2021 work (Abdulrahman, 2021) [1], focuses primarily on improving client selection and participation in federated learning, particularly addressing issues related to client diversity and resource constraints. However, one significant aspect that has not been addressed in this framework is the optimization of client model update efficiency during the federated learning process. In our research, we introduce an approach where we integrate a divergence model to predict the influence of each client's update on the global model and decide if this contribution is relevant or irrelevant to consider communicating to the server.

CSFedAvg in [80] utilizes weight divergence to measure the non-iid degree and prefers participants with lower divergence. However, it necessitates server-side model training and auxiliary data, offering no communication overhead reduction assurance. In our proposal, clients determine server updates based on weight divergence criteria, and reducing communication overhead is guaranteed.

In the work by Orlandi [55], they introduce FedAvgBE a framework designed to mitigate the impact of non-IID by identifying problematic data blocks within clients' local datasets using mean global entropy calculations. While FedAvgBE effectively reduces communication costs, it falls short in addressing the issue of "communication overhead". This oversight could potentially necessitate an increase in the number of training rounds or the addition of more local epochs to improve the model's accuracy.

In the state of the art, the reduction of "communication overhead" is approached from multi-objective perspectives and through data analysis to establish server update criteria. However, it is important to note that data analysis and the search for appropriate thresholds may require additional experiments in some cases. This article presents EntropicFL, an approach designed to leverage entropy metrics for client selection, allowing us to gauge the data uncertainty associated with each client's dataset and select suitable participants for federated learning. Moreover, our methodology establishes criteria on the client side to determine whether an update should be communicated, effectively reducing communication overhead. This dual-pronged approach enhances the extraction of valuable insights from client data while safeguarding data privacy and optimizes the communication process.

3.3 Model Pruning

Here, we examine the techniques of model pruning strategies to further reduce the computational and communication load in FL. Model pruning involves systematically removing less significant parameters from neural networks to create sparser models that require less communication bandwidth for updates. Masking strategies, on the other hand, focus on selectively updating only the most impactful parameters. Both approaches aim to streamline model training and deployment across distributed networks, thereby enhancing the efficiency of FL systems.

This section describes a set of related works considering the problem of reducing communication cost based on model pruning techniques. For instance, FedMP is an adaptive model pruning framework in FL, which enhances both resource efficiency and model accuracy [31]. The solution excels in heterogeneous scenarios, delivering performance up to 4.1 times faster than other methods. However, challenges such as limited communication bandwidth and device heterogeneity were previously overlooked. Namely, this indicates the need for more efficient solutions that address these challenges in FL environments. FedMP, while innovative in FL with adaptive pruning in heterogeneous environments, faces challenges. Its adaptation to each device increases implementation complexity and may be problematic in environments with a diversity of devices. Moreover, this adaptability can cause variations in model quality among nodes, potentially affecting the coherence

and generalization of the global model.

Li *et al.* [41] introduce LotteryFL, an approach that adapts the lottery ticket hypothesis to the context of FL. This hypothesis posits that it is feasible to identify sub-networks (winning tickets) that perform as well or better than the unpruned network and with high generalization capability. Thus, LotteryFL focuses on both model personalization and communication cost reduction. This allows each client to train a specialized sub-network to enable efficient communication. LotteryFL was tested on non-IID datasets, where it showed notable improvements in personalization and communication efficiency compared to other approaches. This methodology is particularly innovative in addressing the challenges associated with non-IID data distribution in FL. However, LotteryFL presents some drawbacks, such as excessive personalization that may limit generalization and the identification/training of optimal sub-networks for each client, which can be complex and resource-intensive.

Lee *et al.* [40] introduced a method for pruning neural networks before training, called SNIP. This approach, based on connection sensitivity, identifies structurally important connections, allowing for a significant reduction in network complexity without the need for iterative cycles of pruning and retraining. It is applicable to various architectures, including convolutional, residual, and recurrent networks. The method offers simplicity and versatility, making it robust to architectural variations and eliminating the need for pre-training and complex pruning schedules. However, this approach is centralized and may encounter scalability issues.

Jiang *et al.* [30] introduced PruneFL, a FL approach that features adaptive and distributed parameter pruning. The approach aims to efficiently train models on edge devices with limited computational and communication resources. PruneFL achieves this by adapting the model size during training, reducing both communication and computational overhead. Furthermore, it incorporates initial pruning on a selected client, followed by additional distributed pruning during the FL process. The approach effectively maintains model accuracy while significantly reducing training time. However, selecting only one client for pruning may introduce bias.

Isik *et al.* [27] presented Federated Probabilistic Mask Training (FedPM). This solution improves communication efficiency in FL by training a stochastic binary mask instead of the model weights. It maintains the weights at their initial random values and discovers an ideal sparse random network within the original dense network. While this is not a model pruning technique, it involves searching for subnetworks within dense architectures, aiming to optimize communication without altering the entire model structure. This approach shows improvements in accuracy, communication efficiency, and convergence speed, making it suitable for scenarios with limited communication resources. However, its use may degrade other trained model properties, such as fairness, robustness, and data leakage. Moreover, the approach is complex to implement compared to SNIP.

Vallapuram *et al.* [70] introduced a FL framework that addresses challenges such as statistical heterogeneity and resource constraints on client devices, called HideNSeek. The approach combines server-side pruning at initialization and the use of a signal supermask, where the terms satisfy the condition of belonging to the set $\{-1, 1\}$. HideNSeek employs one-shot pruning on the server side to determine a sub-network, enabling faster

model convergence with compression rates similar to existing methods. HideNSeek reduces communication cost while maintaining superior learning capability for data with diverse heterogeneities. However, bias may occur when performing server-side pruning. Moreover, HideNSeek's performance decreases in scenarios with high data heterogeneity.

Chang *et al.* [13] propose a model pruning method for FL to address local computation and communication bottlenecks, while maintaining accuracy comparable to that of the original model. The method improves communication efficiency by reducing the number of transmitted model parameters and optimizing both the pruning ratio and device selection. The server determines the number of model layers, K, for which clients are required to randomly send updates. After collecting the local model parameters from the clients, the server separates the parameters of each layer for aggregation, based on the current layer's parameters.

Chapter 4

Proposed and developed strategies

This chapter presents our strategies to address the challenge of communication overhead in FL. Our approach encompasses several key strategies designed to optimize the efficiency and effectiveness of FL processes.

We propose a comprehensive taxonomy designed to classify strategies to reduce communication overhead in FL. This taxonomy serves as a framework for understanding and categorizing various approaches, facilitating a clearer insight into the landscape of communication efficiency in federated learning. Through this taxonomy, we aim to highlight the diverse methodologies employed to tackle communication challenges, providing a structured overview of the field's current state and future directions.

The first strategy, EntropicFL, is an algorithm for client selection in FL. This algorithm enhances the decision-making process, allowing clients to determine the relevance and necessity of communicating updates from the client to the server during the FL process. This selective participation is crucial for minimizing unnecessary communications, reducing overhead, and improving overall system performance.

Additionally, we introduce FedSNIP, a strategy focused on reducing the amount of parameters that must be communicated between clients and the server. This is achieved through a neural network pruning technique, which effectively decreases the complexity of the model without sacrificing accuracy. By pruning redundant or less significant parameters, FedSNIP ensures that only essential information is transmitted, further alleviating the communication burden.

4.1 Taxonomy in Communication Overhead in Federated Learning

In this part, we present a structured overview of our taxonomy, categorized into three main areas based on the operational domain: client-side strategies, server-side strategies, and hybrid strategies. Each area addresses unique challenges in reducing communication overhead, employing specific techniques and methodologies to enhance federated learning efficiency.

4.1.1 Review Technique

This section outlines the methodology employed to conduct a comprehensive literature review focusing on communication overhead strategies in federated learning. It details the electronic databases and digital platforms leveraged for the search and acquisition of relevant scholarly articles. Furthermore, it elucidates the selection of general and specific keywords pivotal for identifying pertinent literature, as well as the criteria and processes adopted for the qualitative assessment of these articles.

4.1.2 Search Criteria

The systematic approach to this literature, was designed to encompass a broad spectrum of research contributions within the domain of FL, particularly emphasizing strategies that address communication efficiency. Recognized electronic databases such as IEEE Xplore, ACM Digital Library, and ScienceDirect served as primary sources for retrieving articles, supplemented by targeted searches on Google Scholar to ensure comprehensive coverage.

Keywords and search terms were meticulously chosen to capture the multifaceted nature of communication overhead in federated learning. These included but were not limited to "federated learning communication efficiency", "data compression in federated learning", "client-side optimization in federated learning", "server-side aggregation strategies", and "hybrid communication strategies in federated learning". The search strategy was iteratively refined to include emerging terms and concepts discovered during the initial review phases.

A multistep assessment procedure was implemented to ascertain the relevance and quality of the articles. Initially, titles and abstracts were screened to exclude studies not directly related to communication strategies in federated learning. Subsequently, a thorough evaluation of the full text of selected articles was conducted, focusing on the novelty, methodological rigor, and impact of the findings. Special attention was given to studies that proposed innovative approaches, provided empirical evidence of effectiveness, or contributed significantly to the theoretical understanding of communication challenges in FL environments.

Table 4.1: Research question and motivation focused on communication overhead in FL.

#	**Research Question**	**Motivation**
1	Which strategies and techniques have been proposed to reduce communication overhead in federated learning?	Review existing and emerging solutions for minimizing communication costs.

4.1.3 Overview of the Taxonomy

Based on the analysis of existing literature and the process describe in subsection 4.1.2, we have selected key and representative articles to construct the taxonomy depicted in

Figure 4.1. This taxonomy illustrates the strategies employed by clients, servers, or both to minimize communication overhead in FL.

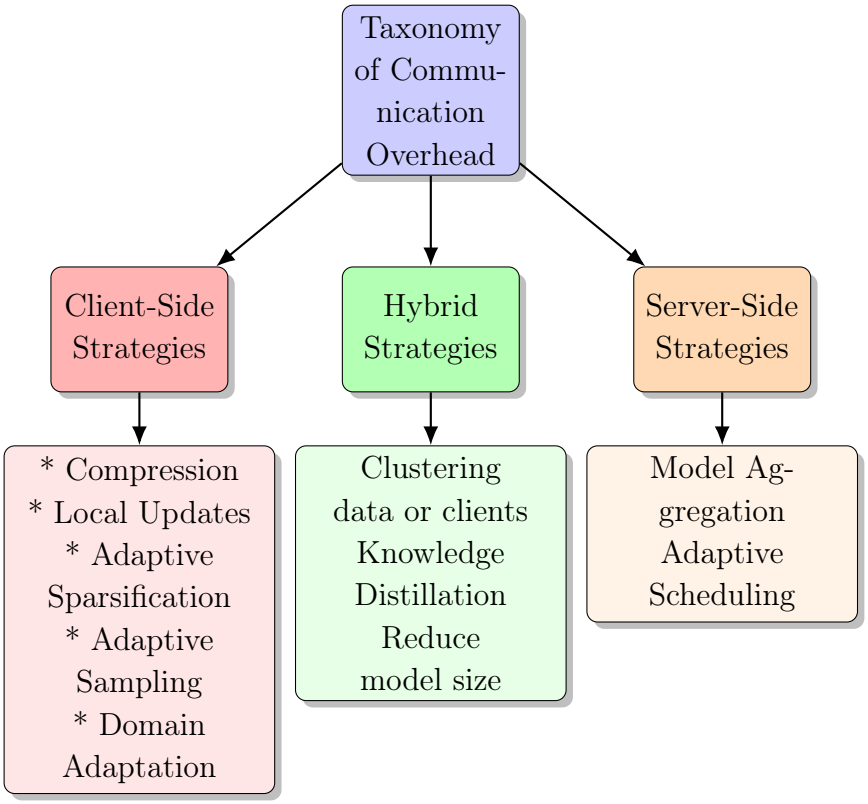


Figure 4.1: Visual taxonomy of communication overhead strategies in federated learning.

- **Client-Side Strategies** Focus on reducing the data size transmitted from clients to the server. This includes data compression techniques and local model update to minimize the frequency and size of communications.
- **Server-Side Strategies** Aim at optimizing the aggregation process and managing communications from the server's perspective. Strategies involve efficient model aggregation and adaptive scheduling to handle incoming client updates.
- **Hybrid Strategies** Involve coordinated efforts between clients and the server to jointly optimize communication efficiency. These strategies balance computational load and communication requirements across the federated network.

4.1.4 Strategies the literature for Reducing Communication Overhead

This part of disertation provides a synthesis of research strategies aimed at addressing the challenges of reducing communication overhead, as outlined in Table 4.1.

Table 4.2: Overview of recent FL research categorized by strategies and approaches.

Paper	**Category**	**Subcategory**	**Description**
Wang et al. [72]	Client-Side	Local Model Updates	Offers a feedback system for clients to synchronize their updates with overall trends, leading to significant boosts in communication efficiency and reducing the quantity of updates to the server.
Li et al. [43]	Hybrid Strategies	Clustering	FedOES surpasses federated averaging by employing cluster training and top-k gradient sparsification, enhancing communication efficiency and reducing overhead through gradient compression.
Wen et al. [75]	Hybrid Strategies	Knowledge Distillation	Introduces Fed2KD, a concise, communication-efficient FL framework leveraging Two-step Knowledge Distillation to tackle communication bottlenecks and non-IID data challenges in IoT.
Lyu et al. [47]	Hybrid Strategies	Knowledge Distillation	Proposes a knowledge distillation-based framework for federated learning that facilitates collaborative learning across heterogeneous clients and servers with diverse model structures, architectures, and resource capabilities.
Mohamed et al. [54]	Server-Side	Adaptive Scheduling	Proposes an algorithm to significantly lower communication overhead and enhance convergence in FL.
Wang et al. [71]	Client-Side	Adaptive Sparsification	An algorithm that enhances FL efficiency through adaptive gradient sparsification, drastically cutting the data volume transmitted to the central server and boosting communication effectiveness.

Continued on next page

Table 4.2 – continued from previous page

Paper	Category	Subcategory	Description
Wang et al. [73]	Client-Side	Adaptive Sampling	Introducing two novel communication-efficient learning algorithms, rFedAvg and rFedAvg+, which surpass current methods in reducing communication rounds.
Jiang et al. [30]	Client-Side	Reduce Model Size	Introduces an innovative Fl method that maintains accuracy on par with the original model while enhancing both communication and computational efficiency during training and inference phases.
Chen et al. [15]	Client-Side	Compression	Minimize communication overhead in federated learning by leveraging data normalization and exploiting statistical properties of local gradients to cut both computational and communication costs.
Liu et al. [44]	Server-Side	Model Aggregation	Adaptive quantization dynamically adjusts quantization resolution based on gradient norm changes per training round, while heterogeneous quantization assigns lower resolutions to slower clients, aligning training times across devices to alleviate communication bottlenecks.
Shu et al.[65]	Client-Side	Adaptive Sampling	Proposed an adaptive sampling method for federated learning that efficiently balances computation and communication, significantly cutting communication costs while maintaining an optimal accuracy-efficiency trade-off.

Continued on next page

Table 4.2 – continued from previous page

Paper	Category	Subcategory	Description
Zhou et al [83]	Hybrid Strategies	Knowledge Distillation	A local model pruning and bidirectional distillation (Bi-distillation) strategy is designed for efficient training, alongside a global model splitting and lightweight data augmentation approach that optimizes aggregation weights within a divided neural network structure (encoder and classifier) for targeted improvement. This integrated method significantly cuts communication costs and mitigates the non-IID issue.
Lou et al. [45]	Client-Side	Clustering	Introduce the DFL-DF technique, blending data feature transfer with neighbor selection to tackle high communication expenses and non-IID local datasets efficiently.
Yue et al [78]	Client-Side	Compression	Introduce FedVote an innovative federated learning approach that enhances communication efficiency, learning dependability, and deployment speed, while also improving resistance to Byzantine attacks.
Zhang et al [79]	Client-Side	Adaptive Sparsification	Introduce a stochastic gradient descent algorithm with a delay compensation feature (FedDgd) optimized for asynchronous federated training.
Sattler et al. [61]	Client-Side	Adaptive Sampling	Introducing Sparse Ternary Compression (STC), a tailored framework for federated learning that enhances top-k gradient sparsification with innovative downstream compression, ternarization, and efficient Golomb encoding for weight updates.

Continued on next page

Table 4.2 – continued from previous page

Paper	Category	Subcategory	Description
Kang et al. [33]	Client-Side	Domain Adaptation	Introduces a Federated Domain Adaptation framework merging FL with Unsupervised Domain Adaptation, offering optimization strategies to lower both computation and communication burdens during training.
Sattler et al [60]	Hybrid Strategies	Knowledge Distillation	Introduces Federated Distillation as an innovative approach within Federated Learning, characterized by unique communication properties.
Yue et al [77]	Hybrid Strategies	Knowledge Distillation	Introduces a predictive coding compression approach for federated learning that dynamically selects predictors to enhance compression efficiency, substantially lowering bandwidth and communication expenses.
Mao et al [51]	Client-Side	Compression	Introduces an efficient FL framework Adaptive Quantized Gradient (AQG) that dynamically modifies quantization levels in response to local gradient updates, optimizing the use of local data distribution heterogeneity to minimize unnecessary data transmission.

Table 4.2 summarizes the primary focus areas within FL, highlighting the diverse methods employed to enhance communication efficiency, model performance, and adaptability across various client and server-side implementations.

In examining the taxonomy generated, it becomes evident that most approaches are predominantly focused on the client side, followed by hybrid strategies. This observation aligns well with the underlying paradigm of Edge Computing, which emphasizes shifting computational processes towards the edge devices. Such a trend underscores the growing emphasis on leveraging the computational capabilities of edge devices, thereby minimizing the reliance on central servers and reducing communication overheads.

Furthermore, the analysis reveals that hybrid strategies play a pivotal role in refining the model's performance and efficiency. These strategies primarily concentrate on optimizing the identification of hyperparameters and thresholds, which are crucial for the adaptive tuning of federated learning models. Through such optimization, it is possible to

achieve significant improvements in model accuracy and generalization while simultaneously addressing challenges related to scalability and heterogeneity of data distributions. The focus on hybrid strategies highlights the ongoing efforts to balance between leveraging edge devices capabilities and the need for centralized coordination to achieve optimal federated learning outcomes.

4.2 EntropicFL

This section presents the system model and outlines the design of our metrics-based approach in EntropicFL, which aims to reduce communication overhead through client selection and criteria for determining updates to submit to the server. Our approach builds on the concepts of statistical and system heterogeneity in client selection discussed by Fu *et al.* [21]. Specifically, we incorporate the Model-Based Utility Measurement to effectively address these challenges.

4.2.1 Determining Update Relevance

One of the key challenges in FL is determining the relevance and impact of each client's update on the global model. This becomes more difficult when client updates are unreliable, as this can lead to model divergence, hindering efficient learning and delaying convergence, as noted by Shanmugarasa et al. [63]. Additionally, the heterogeneity of data and devices complicates the situation further, making it essential to measure the similarity or divergence between local and global models. Various metrics can be used to assess this and decide whether local updates should be transmitted to the server. In EntropicFL, we use the following metrics:

- *Normalized Model Divergence* refers to a computational approach that quantifies the dissimilarity between local and global models by employing a particular norm. It calculates the mean value of a norm function applied to the weight disparity between client i and the global model. Specifically, when utilizing the L_1 norm, the equation is as follows:

$$dv_i = \frac{1}{|w|} \sum_{j=1}^{|w|} \left| \frac{w_{ij} - \bar{w_j}}{\bar{w_j}} \right| \tag{4.1}$$

  Where, w represents the model's weights, while $\overline{w}$ denotes the weights of the global model. The terms w_{ij} and $\overline{w}_j$ refer to the j-th weight of client i and the global model, respectively. A small value implies a close alignment between the models, while a larger distance implies a significant discrepancy.

Our approach employs the Normalized Model Divergence metric with a threshold computed based on the updates clients share with the central server. This threshold is determined as a weighted average of the normalized model divergence from each client. This approach enables us to evaluate the significance of a client's update before transmitting it to the central server. By implementing this methodology, we can significantly reduce communication overhead and enhance the overall efficiency of the FL process.

4.2.2 Suitability Score

Let $\mathcal{C}$ be a set of clients participating in a federated learning system. For each client $c \in \mathcal{C}$, let A_c denote the accuracy, and E_c denote the entropy of client c. The suitability D_c of client c is defined as the weighted sum of its accuracy and entropy, representing its contribution to the global model, as shown in Equation 4.2.

$$D_c = \gamma \cdot A_c + (1 - \gamma) \cdot E_c \tag{4.2}$$

Where γ is a non-negative coefficient representing the relative importance of each attribute. This metric effectively assesses a client's performance by considering two crucial factors: the accuracy of its local model and entropy, indicating data diversity and richness. It is a valuable tool to evaluate individual client data quality and their contribution to the FL training process.

4.2.3 Details and Algorithms

EntropicFL encompasses activities that require execution on both the client and server sides. In this section, we refer to "information" on the client side as a set comprising entropy, accuracy, and Normalized Model Divergence.

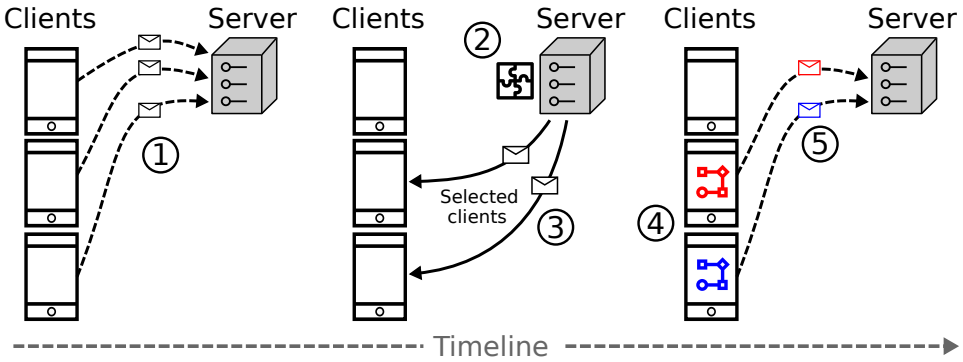


Figure 4.2: Overview of EntropicFL.

Figure 4.2 illustrates the process of our proposed architecture. In ①, clients who have previously received the model from the server train and communicate their updates and information, which in our case are specifically the weights, back to the server. This process is detailed in Algorithm 1. Then, the server receives client information accuracy, entropy, and a quantity of data in ② and calculates the Normalized Model Divergence. In ③, the server performs client selection, algorithm 1 line11, which takes the client's information and the server's capacity K as input. This algorithm normalizes entropy and uses the "ChooseByWeights" method based on Equation 4.2 to perform a random weighted selection, where the weight is associated with Equation 4.2 for each client. Subsequently, two clients from the selected group are prioritized randomly. Finally, in ④, clients execute Algorithm 2, and in step ⑤, the clients calculate the entropy of its data and assess the Normalized Model Divergence between the weights communicated by the server and the local weights obtained after training using Equation 4.1. Subsequently, they transmit their updates (line 9).

Algorithm 1: Central Server

```
   // Initialize the model
1  foreach round t ∈ T do
2      if round == 0 then
3          w_t ← RandomInitialization( ) ;
4          MD_t ← MeanDivergence(∞, ∞) ;
5          foreach client n ∈ N do
6              w^n_{t+1} ← LocalUpdate(MD_t, w_t, d_n);
7      else
8          E, A, D, |d|, w_t ← ReceiveClientInformation( );
9          MD_{t+1} ← (Σ_{i=1}^{|N|} D_i × |d_i|) / (Σ_{i=1}^{|N|} |d_i|);
           // Aggregates the gradients received
10         w_{t+1} ← Σ_{i=1}^{|N|} w^n_{t+1} ;
           // Client selection phase
11         ClientSelecion(E, K);
12 ClientSelection (E, K):
13     C ← NormalizeEntropy(E);
14     L ← ∅;
15     for a_c, e_c ∈ C do
16         c, w ← D_c(a_c, e_c);
17         L ← L ∪ {c, w};
18     S ← ∅;
19     i = 0;
20     while i ≠ K do
21         c_t ← ChooseByWeights(L);
22         S ← S ∪ {c_t};
23         delete c_t from L;
24         K ← K + 1;
25     return S
```

4.3 Applying SNIP for Federated Learning

This section introduces the `FedSNIP` strategy, which utilizes the SNIP (Single-shot Network Pruning) algorithm [40] to decrease the number of parameters communicated from clients to the server in a federated learning context. By leveraging SNIP, `FedSNIP` aims to enhance communication efficiency without compromising the learning model's performance.

4.3.1 Parameter Reduction via SNIP

According to the SNIP algorithm, a certain percentage of neural network parameters, denoted by W, are identified as zeros, indicating parameter sparsity (θ). This approach is classified under One-Shot Pruning, a method where pruning is executed once before the training process begins. This preemptive pruning generates a mask that is applied to the network, effectively reducing its complexity by eliminating non-essential connections.

Algorithm 2: Client

```
   // Receives global model w_t and mean global divergence MD_t
1  LocalUpdate (w_t, MD_t):
       // Entropy calculation using Shannon
2      E ← CalcEntropy( );
       // Compute divergence with global and client weights
3      D ← ComputeDivergence(w_t, w_t^c);
       // Accuracy
4      A ← 0;
5      if client is selected then
6          w_t^c ← w_t;
7          w_{t+1} ← ExecuteLocalTrain( );
8          D ← ComputeDivergence(w_t, w_{t+1}^c);
9          if D ≤ MD_t OR client is priorized then
10             return E, A, D, |d|, w_t
11     return E, A, D, |d|, w_t
```

4.3.2 Detailed Implementation of `FedSNIP`

`FedSNIP` employs SNIP in the federated learning framework to systematically reduce the quantity of parameters required for effective model training and communication. This subsection delves into the specifics of integrating SNIP within federated learning, outlining the steps to achieve optimal parameter sparsity while maintaining or enhancing model accuracy. The methodology for creating and applying the sparsity mask before model training commences is described, highlighting the advantages of this approach in a distributed learning environment.

Figure 4.3 illustrates the workflow of FedSNIP. In Stage 1, the server distributes the model to the clients (Label ①). In Stage 2, after receiving the model, the clients perform pruning using the SNIP technique, resulting in a high zero rate in the weights (Label ②). After the pruning process, the clients train their models. For retransmitting these weights to the server, compression techniques based on sparse matrices are employed, and then the pruned weights are sent to the server (Label ③). In Stage 3, after receiving the weights from the clients, the server applies a sparse matrix representation and proceeds with the aggregation of the data, setting up the model for the next training phase (Label ④). Finally, the server sends the aggregated model back to the clients (Label ⑤) and initiates a new training round (Stage 1).

FedSNIP applies the SNIP technique, known to be a pruning method without the need for fine-tuning and that preserves good accuracy, in a federated environment. Thus, the proposed approach adapts SNIP to the context of FL, demonstrating its efficacy in this new scenario, integrating the concept of one-shot pruning, based on connection sensitivity. Each client in the FL network independently applies a sensitivity-based pruning method to their local model during training, effectively reducing the number of parameters without significant loss of model accuracy.

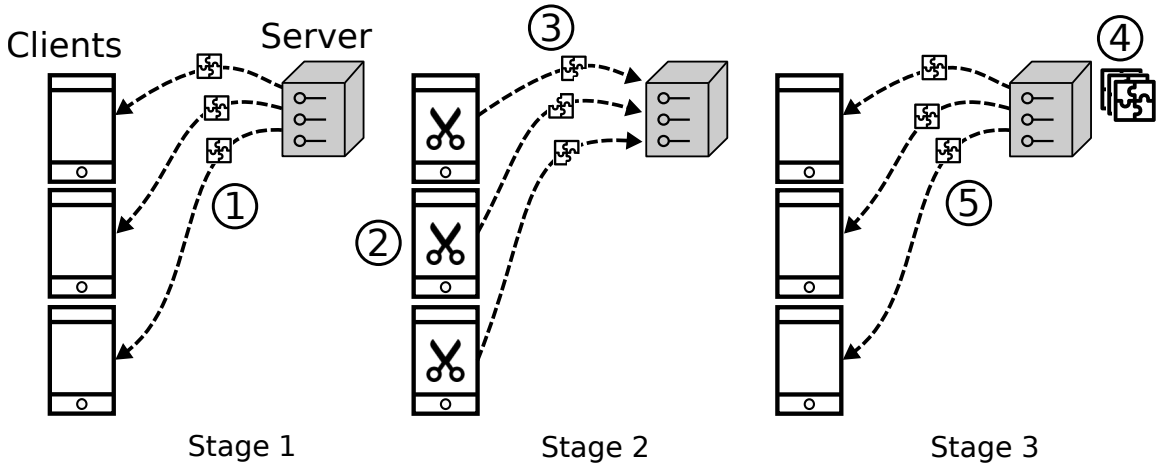


Figure 4.3: System architecture employed by FedSNIP.

The core of FedSNIP lies in its unique pruning process, represented by:

$$W_{pruned} = W \odot M \tag{4.3}$$

In this context, W denotes the weight matrix integral to the neural network architecture. Alongside, M, a binary mask, is produced via sensitivity analysis. This analysis crucially identifies the least sensitive connections within the network, marking them as zero within the mask. Based on this, FedSNIP initiates its process by establishing a global model symbolized by W. This model is characterized by its initialization with randomly assigned weights.

The algorithm 3 describes the main steps involved in the operation of FedSNIP. FedSNIP is a FL approach that integrates one-shot sensitivity-based pruning into the federated training process. Each client (line 3) in the federated network first prunes its local model based on connection sensitivity, ensuring that the remaining parameters are below a defined threshold θ (Line 5). This step is crucial to reduce communication overhead and maintain model efficiency. Subsequently, clients locally train their pruned models on their datasets (line 6). The updated models or their parameters are then sent back to the server for aggregation (line 7). The server aggregates these updates to refine the global model, which is redistributed to clients for further training (lines 8 and 9). The process repeats until the global model converges or meets specified stopping criteria (line 2).

Additionally, the PruneModel function used in FedSNIP is inspired by the SNIP approach. algorithm 4 exemplifies the process employed in this step. This method involves a one-shot pruning process applied at the initialization of the neural network, based on the sensitivity of connections. The connection sensitivity is determined by calculating the impact of each connection on the loss function, allowing the identification of the most crucial connections for the task (line 4). The process involves creating a sensitivity score for each connection and retaining only the top κ percent based on the defined sparsity level k (lines 4 and 7). The pruned model N_{pruned} is generated based on the retained connections and returned by the function (lines 9 and 10). This approach allows for effective model compression without the need for iterative pruning or pre-training, making it particularly suitable for FL environments.

Algorithm 3: FedSNIP

Input: Global model M, pruning threshold θ
Output: Pruned and trained global model

Initialize global model M
foreach round t until convergence **do**
    foreach client c_k **do**
        Receive global model M from the server
        `/* Sensitivity-based pruning */`
        $M_k \leftarrow \text{PruneModel}(M, \theta)$
        `/* Local training */`
        Train M_k on local data D_k
        Send model updates to the server
    `/* Server aggregation */`
    $M \leftarrow AggregateUpdates(M, M_k)$
    `/* Transmission of updated model */`
    Send updated model M to all clients

Algorithm 4: Function PruneModel

Input: Untrained neural network N, training dataset D, sparsity level κ
Output: Pruned model N_{pruned}

Initialize network N with random weights
Sample a mini-batch D_b from D
foreach connection $j \in N$ **do**
    Calculate gradient g_j of the loss with respect to the connection
    Calculate sensitivity score $s_j = |g_j|$
Normalize sensitivity scores
Sort connections by sensitivity and retain the top κ percent
Prune connections with lower sensitivity scores
Generate pruned model N_{pruned} based on retained connections
return N_{pruned}

Chapter 5

General Simulation Settings

In this chapter, we detail the environment and configuration settings employed to evaluate the efficacy of the strategies and algorithms proposed for minimizing communication costs. An in-depth exploration of these settings is critical to understanding the context in which the proposed solutions operate and their potential impact on federated learning systems. By meticulously outlining the experimental setup, we aim to provide a comprehensive foundation for replicating and assessing the effectiveness of these strategies in reducing the communication overhead associated with distributed learning models.

5.1 Simulator Environment

The experimental framework for our study is built on the Flower framework [7], chosen for its versatility and support for federated learning scenarios across various versions of TensorFlow, specifically versions 1.6.0 and 2.15.0. Flower facilitates the simulation of a federated learning environment where each client trains a model with local data and communicates with a central server using gRPC technology. This setup is further streamlined by employing Ansible [1] for efficient configuration of one server and 10 client machines, allowing for seamless deployment and management of experiments.

5.2 Dataset and Generation

Our experiments leverage the widely recognized MNIST dataset [39] and CIFAR-10 [36], comprising $70,000$ grayscale images of handwritten digits, divided into $60,000$ training images and $10,000$ test images. Each image, sized at 28×28 pixels, is labeled with the corresponding digit, offering a comprehensive benchmark for evaluating machine learning model performance in classification tasks. The CIFAR-10 dataset, in contrast, consists of 60,000 color images across 10 classes, with 6,000 images per class. It is segmented into 50,000 training images and 10,000 testing images, with each image being 32×32 pixels in size. This assortment provides a broad base for appraising machine learning models in various visual object recognition tasks, thereby enriching the benchmarking scope alongside grayscale image datasets.

[1] `https://www.ansible.com/`

In FL research, addressing the challenges of non-Identically and Independently Distributed (non-IID) data is crucial. In this dissertation two approaches were utilized to simulate non-IID-scenarios: a custom implementation using the Dirichlet distribution and the FedArtML toolkit [24]. The custom code directly manipulates data allocation per client by adjusting the Dirichlet distribution's concentration parameter (α), offering granular control over the degree of non-IID-ness. In contrast, FedArtML provides a high-level interface for data partitioning, supporting both Dirichlet-based and alternative methods for inducing label and feature distribution skew across nodes. Non-IID data, characterized by unequal and diverse data distributions among clients, poses significant challenges to federated learning models, requiring robust strategies to ensure model accuracy and generalization.

The scenario presented in Figure 5.1 shows the non-IID distribution of data across different clients. The data for each client was partitioned using a Dirichlet distribution with a concentration parameter $\alpha = 0.1$, which leads to a highly imbalanced distribution of labels across clients. This setting simulates real-world scenarios where data is heterogeneously distributed among participants, often referred to as non-IID data.

Figure 5.1(a) shows the distribution of training labels across clients, while the Figure 5.1(b) presents the distribution for test labels. In this FL setting, each client has access to its own partition of data, which is used for local training and evaluation. To assess the overall model performance, a distributed test accuracy metric is used, aggregating the results from all clients after local testing on their respective datasets.

Figure 5.2 shows the data distribution across ten clients. Figure 5.2(a) illustrates the quantity and distribution of training data per client, while Figure 5.2(b) presents the validation data. Unlike the previous scenario, all clients have the same validation data, which is used to assess the generalization of the method, as reflected in the test accuracy.

During the process of generating the non-IID scenario, we observed that the FedArtML tool is more flexible and offers more configuration options for data generation. This contrasts with other implementations that rely heavily on random generation, limiting their control over the scenario.

5.3 Client Configuration in FL Environments

In our exploration, we developed scenarios tailored to FL environments to evaluate the strategies shown in Chapter 4. This approach aimed to evaluate actionable insights to refine FL systems under varied operational frameworks, emphasizing optimizing client configurations and data distribution strategies.

5.3.1 Simulation with Constrained Capacity

To accurately reflect the limitations present in real-world federated learning (FL) deployments, we employed a simulation tool from the Flower framework [7]. This simulation instantiated 30 client nodes while restricting the server's capacity to handle only 15 clients concurrently. This approach was designed to mimic scenarios where the server possesses limited resources, impacting the FL training and aggregation processes. The simulation

highlights the challenges and considerations necessary when designing FL systems under resource constraints, emphasizing the need for efficient client and server coordination.

5.3.2 Realistic Client Deployment Scenario

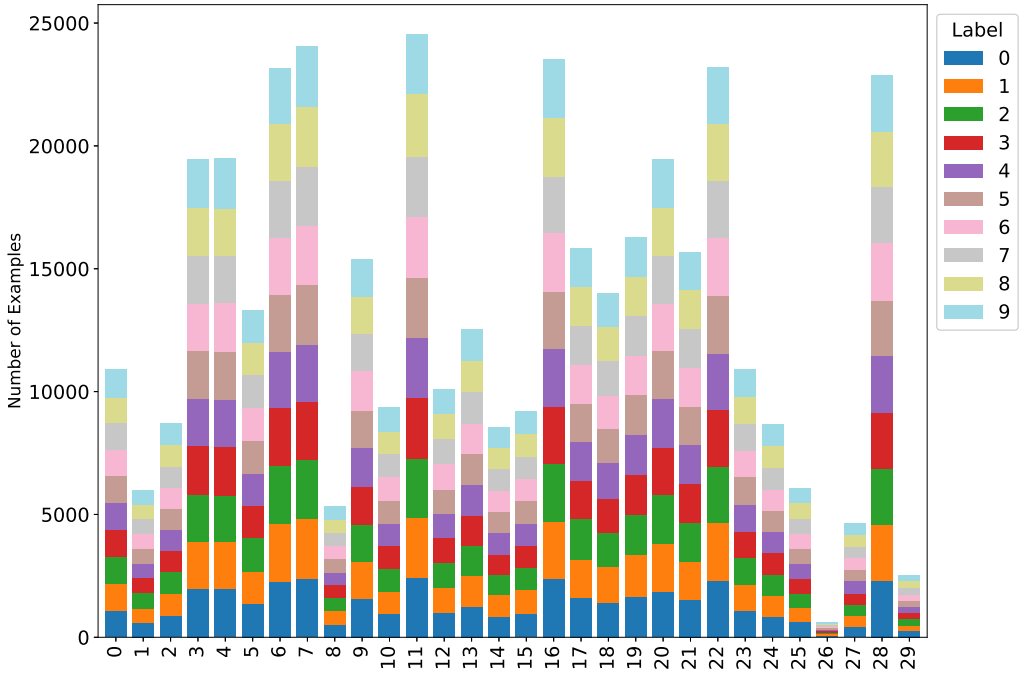
The fidelity and effectiveness of federated learning simulations are heavily dependent on the realistic configuration of client nodes. Utilizing the Ansible automation tool, we configured 10 client machines, each designed to represent a unique computing environment within the federated network. These clients were each allocated a segment of the *dataset* dataset, distributed according to a non-IID model to mimic the diverse and uneven data distributions encountered in real-world settings. This configuration underlines the critical role of client diversity in evaluating the resilience, performance, and scalability of federated learning models. Through this approach, we aim to contribute insights into optimizing federated learning architectures for heterogeneous and distributed environments.

Figure5.3 shows a FL system where clients (Client 1 to Client N) communicate with a central server using the Flower framework for orchestration. Ansible is responsible for provisioning both the server and the clients, ensuring automated setup and deployment. The connections between the server and the clients are bidirectional (Send/Receive), while SSH is used to manage secure connections to the clients for additional operations or management.

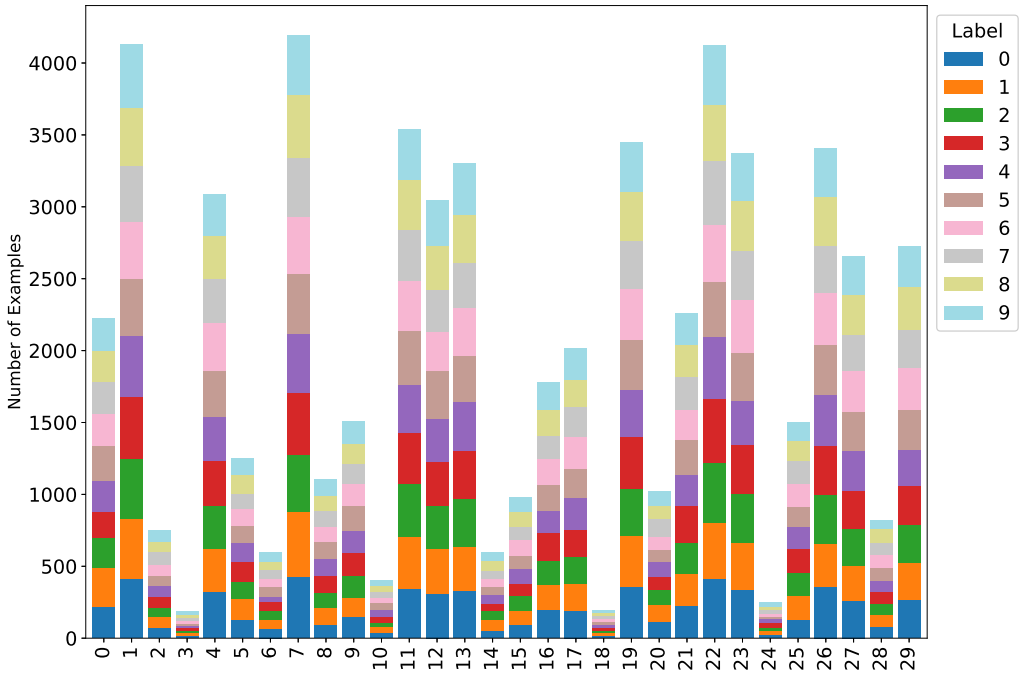
5.4 Architectures of Neural Networks Employed

In this dissertation, we examine the implementation of two seminal neural network architectures: AlexNet, as introduced in [37], and LeNet-5, proposed in [39]. AlexNet, known for its pivotal role in advancing deep learning within the field of computer vision, showcases the power of convolutional neural networks (CNNs) in image recognition tasks. On the other hand, LeNet-5, one of the earliest CNNs, laid the foundational concepts for modern deep learning approaches in digit recognition. This exploration not only underscores the evolution of neural network designs but also their applicability in solving complex pattern recognition problems.

Figure 5.4 shows the architectures used in the experiments for the proposed algorithms. Figure 5.4(a) presents the layers of the LeNet-5 architecture, which is applied to the MNIST dataset. In contrast, Figure 5.4(b) illustrates the AlexNet architecture, which has been adapted for the CIFAR-10 dataset, even though AlexNet was originally designed for ImageNet. In both architectures, the output consists of 10 classes, representing the categories the networks are trained to identify.

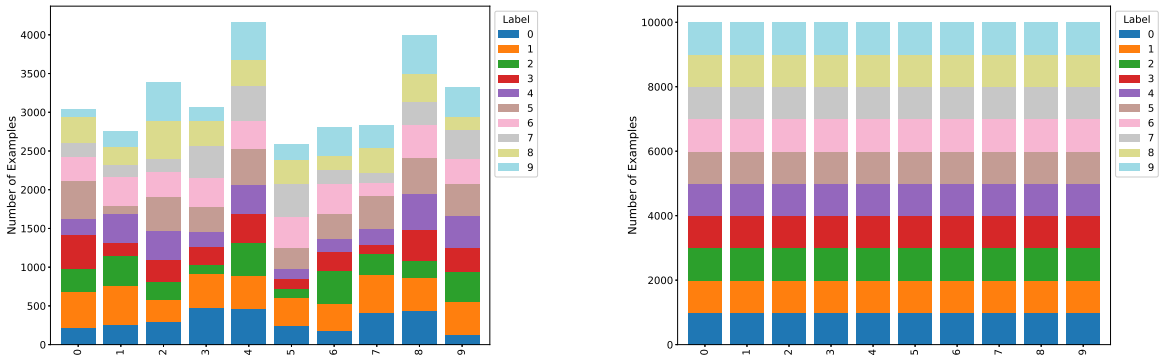


(a) Training data distribution CIFAR-10



(b) Test data distribution CIFAR-10

Figure 5.1: Distribution CIFAR-10 across clients using a Dirichlet distribution with $\alpha = 0.1$.



(a) Training data distribution MNIST

(b) Test data distribution MNIST

Figure 5.2: Distribution MNIST across clients using a Dirichlet distribution with $\alpha = 0.5$.

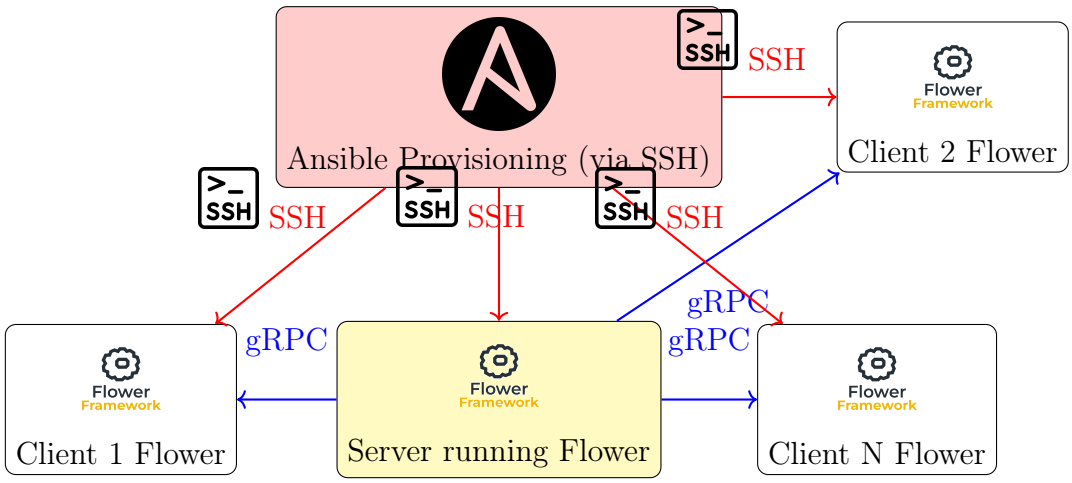
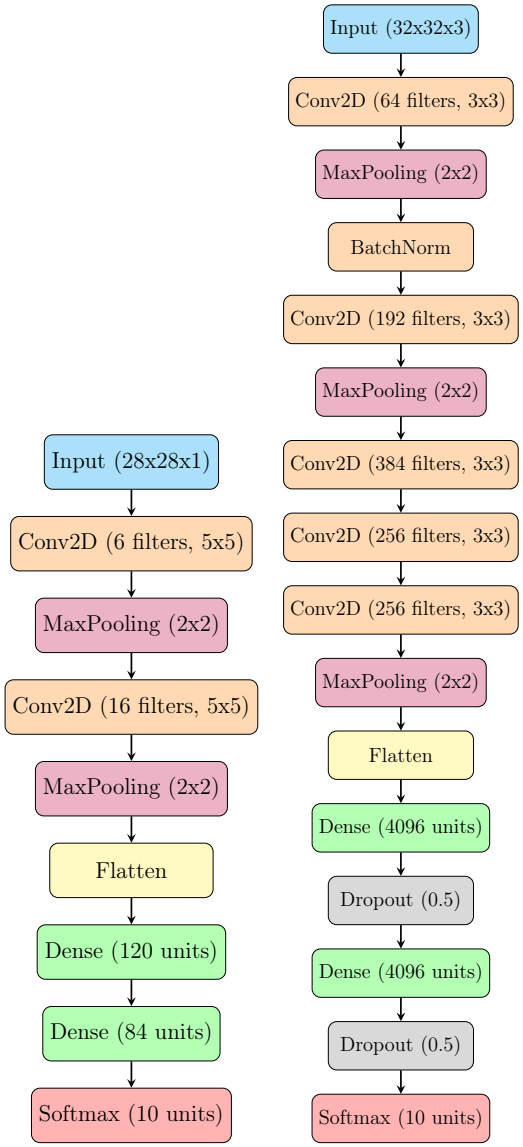


Figure 5.3: FL system where Ansible provisions the server and clients over SSH. Flower runs on the server and clients, with gRPC used for communication.



(a) Architecture of LeNet-5 (b) Architecture of AlexNet for CIFAR-10

Figure 5.4: LeNet-5 and AlexNet architectures.

Chapter 6

Results and Discussion

In this chapter, we present and analyze the results obtained from the experiments conducted to evaluate the performance of EntropicFL and FedSNIP, comparing them with strategies from the literature and considering non-IID scenarios.

6.1 EntropicFL Analysis

As previously described, to test EntropicFL, we trained it using the AlexNet architecture, adapted to the non-IID dataset. The model was trained with 30 clients, as defined in Subsection 5.3.1, while the server was configured to handle up to 15 clients simultaneously. This scenario was designed to establish a criterion for client selection and assess its impact on performance. For comparison, we selected the CMFL algorithm, as it offers a fast criterion to optimize FedAvg and works in non-IID scenario. In our approach, we use model weights to evaluate divergence, contrasting with the original paper, which used gradients. Another strategy we selected is FedAvgBE, designed to handle the impact of non-IID data environments by using entropy as a criterion for batch selection and mitigating the effects of non-IID scenarios. We included FedAvg as a baseline because it is widely used in the literature and employs random client selection. This same random selection strategy is also adopted by other methods, such as CMFL and FedAvgBE.

Other strategies, such as Oort, presented by Fan *et al.* [38], were not considered in this experiment, were not considered, as they combine criteria for both data and system heterogeneity, which focus on data and hardware differences, respectively.

During the execution of our experiments, the accuracy achieved by EntropicFL was found to be comparable to that of the FedAvg method throughout the FL process. However, a discernible difference emerges due to a performance decline observed in certain rounds. This decline can be attributed to the process of client selection and the evaluation of model divergence, which influences the decision of each client regarding the sharing of their model weights with the server.

EntropicFL is configured as previously outlined, and successfully addresses the issue of communication overhead, as demonstrated in Figures 6.1(a), 6.1(b), and 6.1(c), with the parameter γ set at 0, 0.5, and 1, respectively. When γ is set to 0, the selection criteria primarily rely on the metric defined in equation 4.2, with entropy playing a pivotal role.

The accuracy observed under this configuration closely aligns with the results when γ is fixed at 1. Remarkably, adjustments to the γ parameter were found not to significantly alter accuracy levels, which consistently mirrored those achieved by the conventional FedAvg method. The noteworthy aspect of this observation is the substantial reduction in communication overhead costs despite the maintained accuracy.

Figure 6.2 present the reduction in communication overhead for different values of γ. In particular, Figure 6.2(a) shows the results when $\gamma = 0$, demonstrating the communication overhead reduction achieved under this setting. Similarly, Figure 6.2(b) displays the overhead reduction for $\gamma = 0.5$, while Figure 6.2(c) illustrates the case where $\gamma = 1$. All figures compare the performance of the proposed method with the FedAvg baseline, highlighting the effectiveness of the approach across varying values of γ.

Table 6.1: Comparison of EntropicFL with other methods

Method	**Non-IID Data**	**Comm. Reduction**	**Accuracy**	**Evaluation Metric**
CMFL	Yes	Yes	High	Gradients / Weights
FedAvgBE	Yes	None	Moderate	Entropy
FedAvg	No	No	High	Random
EntropicFL	Yes	Yes	High	Model Weights

Table 6.1 compares our proposed method with CMFL, FedAvgBE, and FedAvg. All methods, except FedAvgBE, maintain high accuracy. However, FedAvgBE is noted for its moderate accuracy, as it is tailored to handle non-IID data environments.

In terms of communication reduction, EntropicFL stands out as it reduces communication overhead, unlike CMFL and FedAvgBE, which do not reduce communication at all. Furthermore, CMFL utilizes both gradients and model weights for divergence evaluation, while FedAvgBE relies on entropy, and FedAvg applies a random selection strategy.

Our proposal, EntropicFL, combines model weights for divergence evaluation and reduces communication overhead, providing a more comprehensive approach to address federated learning challenges, especially in non-IID scenarios.

According to the taxonomy shown in Figure 4.1, EntropicFL is a client-server strategy that reduces updates to the server based on specific criteria. It requires interaction with the server to compute and receive information, such as model divergence, to determine whether an update should be communicated. This process helps reduce uplink communication. Additionally, the server uses data entropy and client accuracy to establish a selection criterion.

One issue that may arise is potential bias, as two clients should be prioritized, even in situations where their updates are not relevant. This could happen when the model has not yet reached the required number of iterations. Another problem is that the server assumes the information provided by clients is accurate, which is not addressed by this strategy, nor is it evaluated by other approaches.

Finally, throughout the experiment, entropy is identified as a useful metric for selecting clients, but it should be combined with other metrics. In this case, it was combined with accuracy, as the goal is to maintain the highest possible accuracy while selecting high-quality data. This approach also reduces communication overhead through a criterion

based on model divergence.

6.2 FedSNIP Analysis

As described in Algorithm 3, FedSNIP is a strategy where clients perform SNIP pruning once they receive the model. To validate FedSNIP, the CMFL strategy from the literature was selected for comparison. Other strategies, such as stochastic training or pruned structures, were not considered. FedAvg was used to validate accuracy performance, and CMFL was used to assess the reduction of communication overhead based on the model parameters.

The MNIST dataset was used, and the simulation setup, as described in Subsection 5.3.2, involved 10 clients and one server. The clients were trained using the LeNet-5 network, which was chosen due to its relatively low number of parameters, making it suitable for resource-constrained devices. Since SNIP requires multiplying weights by a mask, simpler networks are more realistic for devices with limited computational power.

An exploratory analysis was conducted to determine the optimal hyperparameters for the solutions discussed in the literature. As a result, the learning rate was set to 0.001, with a batch size of 32, using the Stochastic Gradient Descent (SGD) method. These parameters were selected to establish a balanced criterion for training, as hyperparameters significantly influence training time. For example, reducing the batch size increases the time required for training.

The experiments were conducted at various sparsity levels—60%, 70%, 80%, and 90%—represented by the parameter θ in Algorithm 3. In the figures, these levels are labeled directly as "sparsity" rather than using the symbol θ. The aggregated results, shown in Figure 6.3, indicate each sparsity level and include comparisons with algorithms from the literature, such as CMFL [72] and FedAvg [53]. It is worth noting that CMFL does not require complex configuration, making it easier to implement. However, the threshold chosen for CMFL depends heavily on the distribution of the data, which can affect its overall performance. The results underscore the feasibility of the proposed approach, demonstrating a notable balance between reducing communication overhead and preserving model accuracy. Figures 6.3(a), 6.3(b), 6.3(c), and 6.3(d) display the global model's accuracy at different sparsity levels. During the experiments, it was observed that a sparsity level of 90% led to a decrease in convergence rate. However, a positive reversal of this trend was noted over the communication rounds, suggesting continuous improvement in model performance. The CMFL approach manages to reduce communication costs and achieve acceptable accuracy, although there were instances of accuracy drops. This is due to the adopted criterion within the approach, which determines whether the total parameters will or will not be communicated, thus influencing accuracy in certain rounds.

Figure 6.3 presents the impact of various sparsity levels on model accuracy. Notably, this figure demonstrates that reducing the number of parameters to zero does not substantially degrade the accuracy of the model.

Furthermore, Figures 6.4(a), 6.4(b), 6.4(c), and 6.4(d) illustrate a comparison in the

volume of communicated parameters, with FedAvg serving as the reference standard. The results from these experiments indicate that FedSNIP consistently outperforms CMFL in reducing the amount of communicated parameters, showcasing a significant efficiency improvement. This reduction was largely uniform across different test scenarios, albeit with some minor fluctuations.

Although FedSNIP reduces communication overhead, our analysis shows that the number of accumulated parameters is consistent with the selected sparsity level (θ). When a higher level of sparsity is chosen, accuracy is lower in the initial rounds. On the other hand, selecting a lower sparsity level leads to higher accuracy, but the number of parameters communicated to the server increases.

Regarding the convergence of the algorithm, we can intuitively observe that as the sparsity level decreases, it approaches the behavior of FedAvg, suggesting a potential for convergence. However, this requires further study and should be formally proven with a convergence analysis.

The architecture's impact is also significant, influencing the time to convergence or the need for more communication rounds. Larger models with more parameters increase the number of operations required, generate more masks, and significantly raise the computational overhead. While FedSNIP effectively reduces communication overhead in smaller networks, further studies are necessary to assess its performance in larger networks and to determine, based on resource capacity, which clients can do pruning of the model.

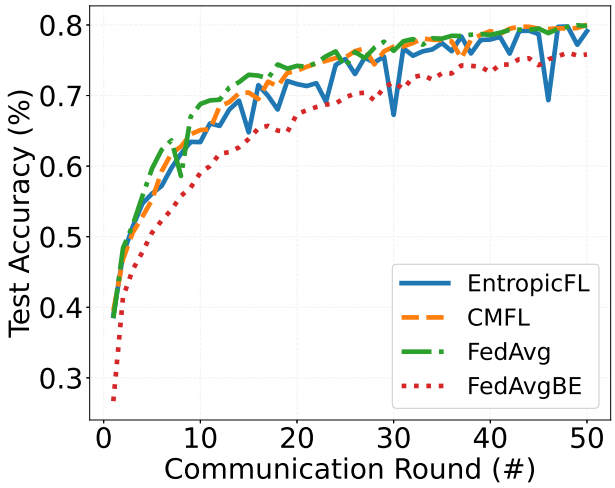
Finally, after analyzing the development of both strategies, EntropicFL and FedSNIP, it is important to note that these approaches should not be viewed as direct competitors, but rather as complementary strategies. Each method addresses different aspects of federated learning challenges. EntropicFL focuses on optimizing client selection and reducing communication overhead by leveraging model divergence and entropy, making it well-suited for environments with limited communication resources. On the other hand, FedSNIP integrates model pruning to further reduce the number of parameters shared with the server, which is particularly beneficial in resource-constrained environments with short and medium models.

By combining the strengths of both strategies, it is possible to achieve more efficient federated learning systems that adapt to varying client capabilities and network constraints. Instead of comparing the two approaches, they should be seen as complementary solutions that can work together to enhance overall system performance.

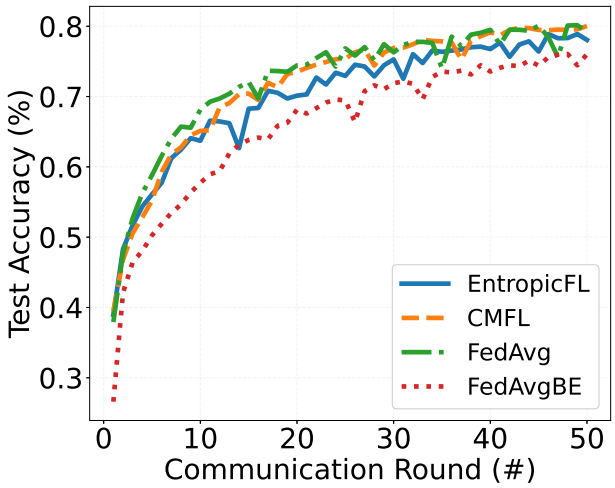
Table 6.2 highlights the key differences and complementarities between EntropicFL and FedSNIP. While EntropicFL focuses on optimizing client selection based on model divergence and entropy, FedSNIP reduces communication overhead through SNIP pruning. Both strategies were evaluated on different models, with EntropicFL tested on AlexNet (CIFAR-10) and FedSNIP on LeNet-5 (MNIST).

Table 6.2: Comparison between EntropicFL and FedSNIP

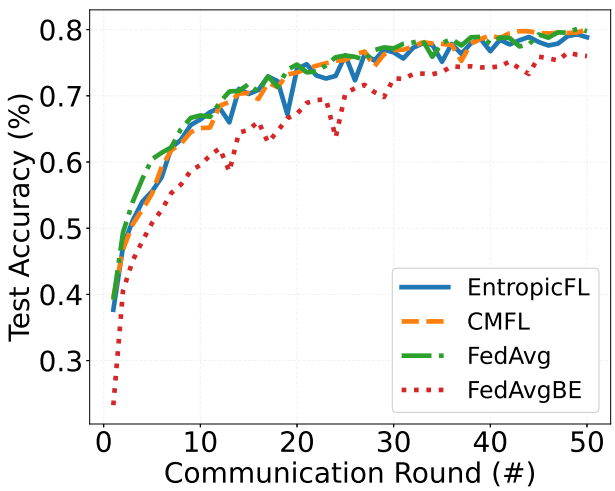
Feature	**Description**
Objective	EntropicFL: Optimize client selection FedSNIP: Reduce communication overhead
Primary Mechanism	EntropicFL: Model divergence and entropy FedSNIP: SNIP pruning to reduce parameters
Impact on Communication	EntropicFL: Fewer updates sent FedSNIP: Fewer parameters communicated
Model Evaluated	EntropicFL: AlexNet (CIFAR-10) FedSNIP: LeNet-5 (MNIST)
Sparsity Levels	EntropicFL: Not applicable FedSNIP: 60%, 70%, 80%, 90%
Target Scenario	EntropicFL: Limited communication FedSNIP: Resource-constrained environments
Complementarity	EntropicFL: Combines well with FedSNIP FedSNIP: Can be used with EntropicFL for optimization



(a) Accuracy for $\gamma = 0$

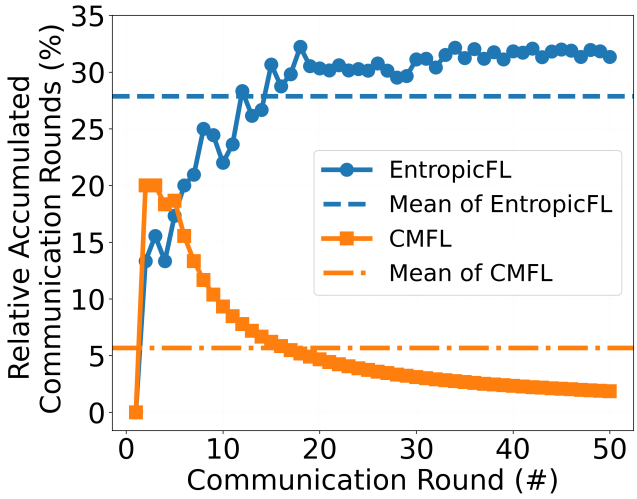


(b) Accuracy for $\gamma = 0.5$

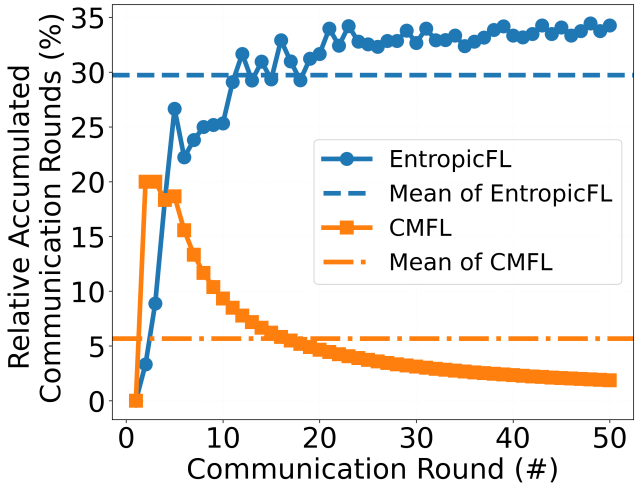


(c) Accuracy for $\gamma = 1$

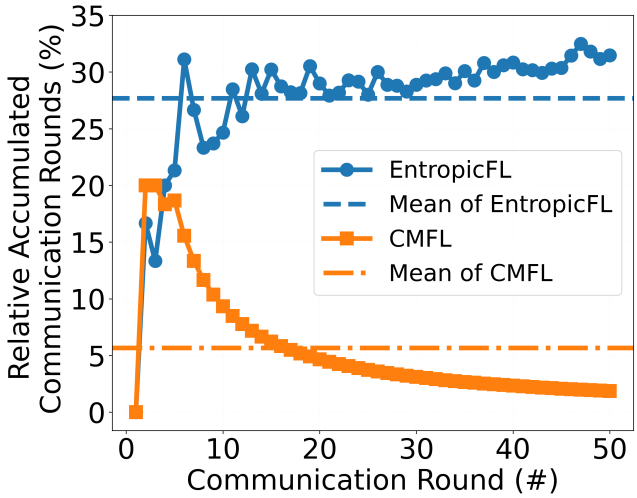
Figure 6.1: Distributed test accuracy with varying γ values in Equation 4.2 using AlexNet.



(a) Communication overhead reduction for $\gamma = 0$

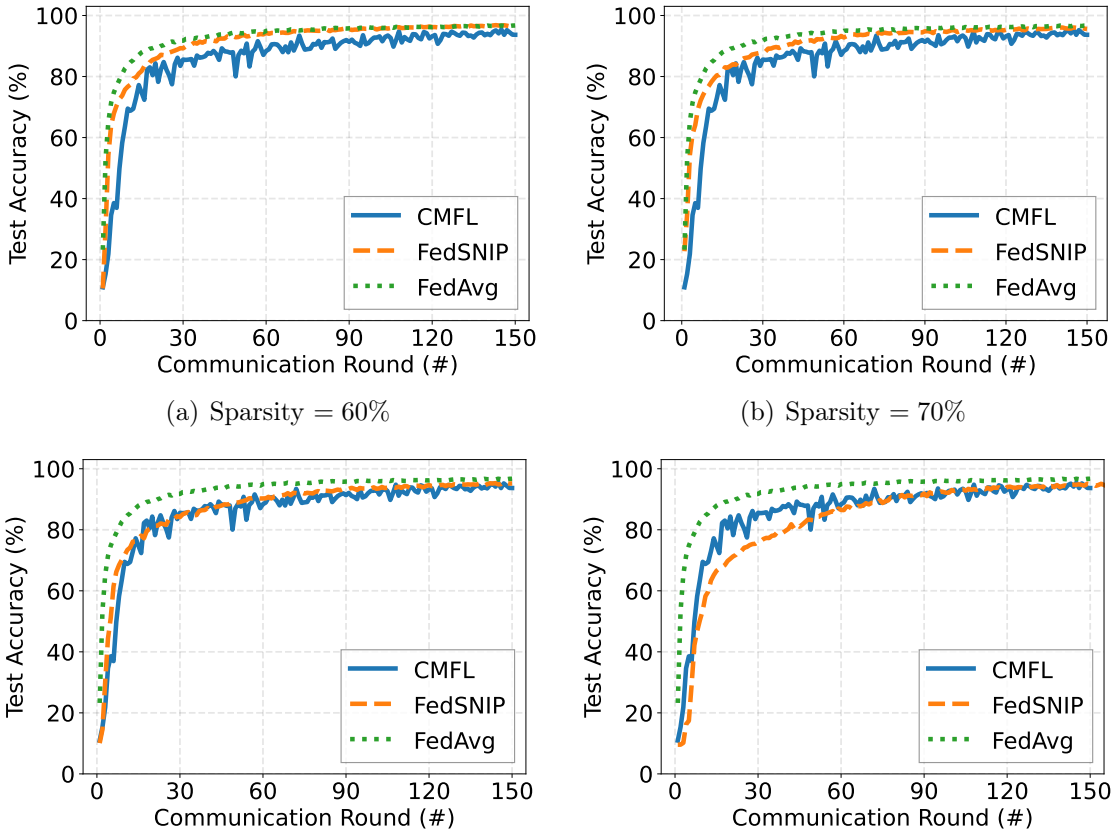


(b) Communication overhead reduction for $\gamma = 0.5$



(c) Communication overhead reduction for $\gamma = 1$

Figure 6.2: Communication overhead reduction with varying γ values compared to the FedAvg baseline.



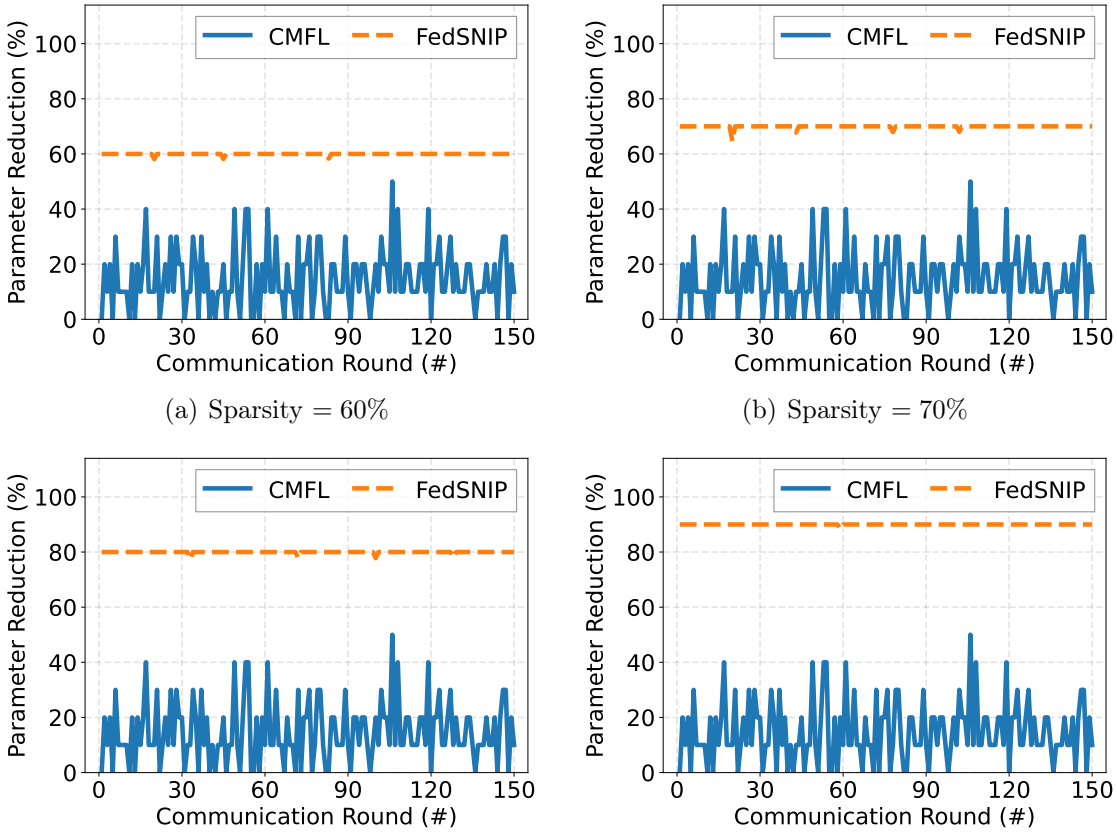
(a) Sparsity = 60%

(b) Sparsity = 70%

(c) Sparsity = 80%

(d) Sparsity = 90%

Figure 6.3: Test accuracy of the global model at different sparsity levels with LeNet-5.



(a) Sparsity = 60%

(b) Sparsity = 70%

(c) Sparsity = 80%

(d) Sparsity = 90%

Figure 6.4: Parameter reduction communicated to the server in relation to FedAvg per communication round considering different levels of sparsity.

Chapter 7

Conclusion

The problem of communication overhead in FL is crucial, particularly when focusing on the processes involving edge devices and their capability to train complex neural networks. In this dissertation, we analyze approaches to reduce communication overhead and present two algorithms designed to address this issue. These algorithms have been validated in non-IID scenarios, demonstrating their effectiveness. The following contributions are provided:

1. **Effectiveness of the Implemented Strategies:** The experiments and simulations conducted have demonstrated that client selection strategies, one based on entropy and the other on one-shot model pruning, effectively reduce communication overhead in FL. The first strategy employs entropy metrics as a means to identify high-quality client data, thus offering a method to select clients based on data quality while maintaining FL's core principle of data privacy. This strategy reduces communication overhead by leveraging model divergence to determine whether a client's update should be transmitted or not. Meanwhile, the strategy based on model pruning decreases communication overhead by removing connections, effectively reducing the number of parameters that need to be shared with the server.

2. **Impact on Communication:** In both proposed algorithms, a reduction in the volume of data transmitted between clients and the server was observed, using relative communication rounds to measure this change. This confirms the initial hypothesis that these strategies could enhance communication efficiency in federated learning scenarios. One approach involved using model divergence as a measure of the quality of an update. It was noted that in some rounds, accuracy decreased but subsequently recovered in the following rounds.

3. **Contributions to the Field of FL:** This dissertation contributes to the field FL by presenting and validating approaches that minimize communication overhead, utilizing strategic client selection and model pruning. Additionally, it identifies promising directions for future research, which include refining pruning techniques and developing dynamic client selection methods that leverage real-time metrics. Furthermore, this study proposes the investigation of more sophisticated compression techniques, supported by robust mathematical foundations.

7.1 Future Work

One of the main challenges in Federated Learning is the high communication cost associated with distributed training. This issue is critical because communication overhead can affect both the efficiency and scalability of FL systems. As part of our future work, we will explore methods to reduce communication overhead based on criteria to preserve the quantity of information and quality. Additionally, we plan to conduct a theoretical study of the convergence of the proposed algorithms. Understanding and improving convergence is key to ensuring the reliability of the algorithms as we work to reduce communication overhead. We also aim to develop a compression scheme that minimizes data transmission and establish a benchmark to evaluate the performance of the FL training process..

Another direction will be the incorporation of context-aware techniques to enhance decision-making processes. By adapting to the environment in which the system operates, we expect to further reduce communication costs while optimizing the hyperparameters of the model. In addition, we plan to discuss possible adaptations for a peer-to-peer (P2P) network. A P2P network could offer a more decentralized structure, which may help reduce communication demands and make the system more scalable. We will also explore how different types of data influence the pruning parameters. By adjusting pruning based on the data, we hope to minimize the amount of data communicated, contributing to lower communication costs.

Moreover, we plan to analyze how the behavior of the network affects decision-making in the model. Understanding network conditions such as latency or bandwidth can help optimize when and how communication occurs, potentially reducing unnecessary data transfer. Finally, we will consider implementing the system using OpenMPI or OpenMP. These tools are designed for efficient data handling and parallel processing, which could further reduce both computational and communication costs. Alongside this, we will conduct a study of the computational costs to ensure that any improvements are cost-effective in terms of both resources and time.

7.2 Production

- [17] **Condori Bustincio, R. W.**, de Souza, A. M., Da Costa, J. B., & Bittencourt, L. *EntropicFL: Efficient Federated Learning via Data Entropy and Model Divergence*. 12th International Workshop on Cloud and Edge Computing, and Applications Management, part of UCC -23: Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing, December 2023, Article No.: 55, Pages 1–6. `https://doi.org/10.1145/3603166.3632611`

- [11] **Condori Bustincio, R. W.**, de Souza, A. M., Da Costa, J. B., Gonzalez, Luis F. G. & Bittencourt, L. *FedSNIP: Método baseado em Poda de Modelo de Etapa Única para Comunicação Eficiente em Aprendizado Federado*. In Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, (pp. 980-993). Porto Alegre: SBC. doi:10.5753/sbrc.2024.1520 `https://doi.org/10.5753/sbrc.2024.1520`

Bibliography

[1] Sawsan Abdulrahman, Hanine Tout, Azzam Mourad, and Chamseddine Talhi. Fedmccs: Multicriteria client selection model for optimal iot federated learning. *IEEE Internet of Things Journal*, 8(6):4723–4735, 2021.

[2] Sawsan Abdulrahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497, April 2021.

[3] Omair Rashed Abdulwareth Almanifi, Chee-Onn Chow, Mau-Luen Tham, Joon Huang Chuah, and Jeevan Kanesan. Communication and computation efficiency in Federated Learning: A survey. *Internet of Things*, 22:100742, July 2023.

[4] Muhammad Asad, Saima Shaukat, Dou Hu, Zekun Wang, Ehsan Javanmardi, Jin Nakazato, and Manabu Tsukada. Limitations and future aspects of communication costs in federated learning: A survey. *Sensors*, 23(17), 2023.

[5] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.

[6] Kristin P. Bennett and Emilio Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7(46):1265–1281, 2006.

[7] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[9] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.

[10] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[11] Rómulo Bustincio, Allan Souza, Joahannes Costa, Luis Gonzalez, and Luiz Bittencourt. Fedsnip: Método baseado em poda de modelo de etapa Única para comunicação eficiente em aprendizado federado. In *Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 980–993, Porto Alegre, RS, Brasil, 2024. SBC.

[12] Cláudio G. S. Capanema, Allan M. de Souza, Joahannes B. D. da Costa, Fabrício A. Silva, Leandro A. Villas, and Antonio A. F. Loureiro. A novel prediction technique for federated learning. *IEEE Transactions on Emerging Topics in Computing*, pages 1–16, 2024.

[13] Min-Kuan Chang, Yu-Wei Chan, and Ting-En Wu. Communication-Efficient Federated Learning with Model Pruning. In Jason C. Hung, Neil Y. Yen, and Jia-Wei Chang, editors, *Frontier Computing*, volume 1031, pages 67–76. Springer Nature Singapore, Singapore, 2023. Series Title: Lecture Notes in Electrical Engineering.

[14] O. Chapelle, B. Scholkopf, and A. Zien, Eds. Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[15] Guojun Chen, Kaixuan Xie, Yuheng Tu, Tiecheng Song, Yinfei Xu, Jing Hu, and Lun Xin. NQFL: Nonuniform Quantization for Communication Efficient Federated Learning. *IEEE Communications Letters*, 28(2):332–336, February 2024.

[16] Lorenzo Ciampiconi, Adam Elwood, Marco Leonardi, Ashraf Mohamed, and Alessandro Rozza. A survey and taxonomy of loss functions in machine learning, 2023.

[17] Rómulo Walter Condori Bustincio, Allan M. de Souza, Joahannes B. D. Da Costa, and Luiz Bittencourt. Entropicfl: Efficient federated learning via data entropy and model divergence. In *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, UCC '23, New York, NY, USA, 2024. Association for Computing Machinery.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2 edition, 2006.

[19] Allan M. de Souza, Filipe Maciel, Joahannes B.D. da Costa, Luiz F. Bittencourt, Eduardo Cerqueira, Antonio A.F. Loureiro, and Leandro A. Villas. Adaptive client selection with personalization for communication efficient federated learning. *Ad Hoc Networks*, 157:103462, 2024.

[20] F. Emmert-Streib and M. Dehmer. Taxonomy of machine learning paradigms: a data-centric perspective. *WIREs Data Mining and Knowledge Discovery*, 12, 2022.

[21] Lei Fu, Huanle Zhang, Ge Gao, Mi Zhang, and Xin Liu. Client selection in federated learning: Principles, challenges, and opportunities. *IEEE Internet of Things Journal*, pages 1–1, 2023.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] Badra Souhila Guendouzi, Samir Ouchani, Hiba EL Assaad, and Madeleine EL Zaher. A systematic review of federated learning: Challenges, aggregation methods, and development tools. *Journal of Network and Computer Applications*, page 103714, 2023.

[24] Daniel Mauricio Jimenez Gutierrez, Aris Anagnostopoulos, Ioannis Chatzigiannakis, and Andrea Vitaletti. Fedartml. `https://pypi.org/project/FedArtML/`, 2023. Federated Learning for Artificial Intelligence and Machine Learning library.

[25] Daniel Mauricio Jimenez Gutierrez, Aris Anagnostopoulos, Ioannis Chatzigiannakis, and Andrea Vitaletti. Fedartml: A tool to facilitate the generation of non-iid datasets in a controlled way to support federated learning research. *IEEE Access*, 12:81004–81016, 2024.

[26] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, 2021.

[27] Berivan Isik, Francesco Pase, Deniz Gunduz, Sanmi Koyejo, Tsachy Weissman, and Michele Zorzi. Communication-Efficient Federated Learning through Importance Sampling, June 2023. arXiv:2306.12625 [cs, stat].

[28] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Publishing Company, Incorporated, 2014.

[29] Nourah Janbi, Iyad Katib, and Rashid Mehmood. Distributed artificial intelligence: Taxonomy, review, framework, and reference architecture. *Intelligent Systems with Applications*, 18:200231, 2023.

[30] Yuang Jiang, Shiqiang Wang, Víctor Valls, Bong Jun Ko, Wei-Han Lee, Kin K. Leung, and Leandros Tassiulas. Model Pruning Enables Efficient Federated Learning on Edge Devices. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10374–10386, December 2023.

[31] Zhida Jiang, Yang Xu, Hongli Xu, Zhiyuan Wang, Chunming Qiao, and Yangming Zhao. FedMP: Federated Learning through Adaptive Model Pruning in Heterogeneous Edge Computing. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 767–779, Kuala Lumpur, Malaysia, May 2022. IEEE.

[32] Peter Kairouz and H. Brendan et al. McMahan. *Advances and Open Problems in Federated Learning*. arXiv, 2021.

[33] Hua Kang, Zhiyang Li, and Qian Zhang. Communicational and Computational Efficient Federated Domain Adaptation. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):3678–3689, December 2022.

[34] Jakub Konečnỳ, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[35] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency, 2017.

[36] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[38] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pages 19–35. USENIX Association, July 2021.

[39] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[40] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

[41] Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li. LotteryFL: Personalized and Communication-Efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets, August 2020. arXiv:2008.03371 [cs, stat].

[42] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, May 2020.

[43] Yue Li, Zengjin Liu, Yunfa Huang, and Peiting Xu. FedOES: An Efficient Federated Learning Approach. In *2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE)*, pages 135–139, February 2023.

[44] Heting Liu, Fang He, and Guohong Cao. Communication-Efficient Federated Learning for Heterogeneous Edge Devices Based on Adaptive Gradient Quantization. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10, May 2023. ISSN: 2641-9874.

[45] Wenxiao Lou, Yang Xu, Hongli Xu, and Yunming Liao. Decentralized Federated Learning with Data Feature Transmission and Neighbor Selection. In *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 688–695, January 2023. ISSN: 2690-5965.

[46] Bing Luo, Xiang Li, Shiqiang Wang, Jianwei Huang, and Leandros Tassiulas. Cost-effective federated learning in mobile edge networks. *IEEE Journal on Selected Areas in Communications*, 39(12):3606–3621, 2021.

[47] Feng Lyu, Cheng Tang, Yongheng Deng, Tong Liu, Yongmin Zhang, and Yaoxue Zhang. A Prototype-Based Knowledge Distillation Framework for Heterogeneous Federated Learning. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pages 1–11, July 2023. ISSN: 2575-8411.

[48] Filipe Maciel, Allan M. De Souza, Luiz F. Bittencourt, and Leandro A. Villas. Resource aware client selection for federated learning in iot scenarios. In *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, pages 1–8, 2023.

[49] Filipe Maciel, Allan M. de Souza, Luiz F. Bittencourt, Leandro A. Villas, and Torsten Braun. Federated learning energy saving through client selection. *Pervasive and Mobile Computing*, 103:101948, 2024.

[50] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[51] Yuzhu Mao, Zihao Zhao, Guangfeng Yan, Yang Liu, Tian Lan, Linqi Song, and Wenbo Ding. Communication efficient federated learning with adaptive quantization, 2021.

[52] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[53] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[54] Aissa Hadj Mohamed, Nícolas R. G. Assumpçáo, Carlos A. Astudillo, Allan M. de Souza, Luiz F. Bittencourt, and Leandro A. Villas. Compressed Client Selection for Efficient Communication in Federated Learning. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pages 508–516, January 2023. ISSN: 2331-9860.

[55] Fernanda C. Orlandi, Julio C. S. Dos Anjos, Valderi R. Q. Leithardt, Juan Francisco De Paz Santana, and Claudio F. R. Geyer. Entropy to mitigate non-iid data problem on federated learning for the edge intelligence environment. *IEEE Access*, 11:78845–78857, 2023.

[56] Kilian Pfeiffer, Martin Rapp, Ramin Khalili, and Jörg Henkel. Federated Learning for Computationally Constrained Heterogeneous Devices: A Survey. *ACM Comput. Surv.*, 55(14s), July 2023. Place: New York, NY, USA Publisher: Association for Computing Machinery.

[57] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[58] W. Rudin. *Principles of Mathematical Analysis*. International series in pure and applied mathematics. McGraw-Hill, 1976.

[59] Fahad Sabah, Yuwen Chen, Zhen Yang, Muhammad Azam, Nadeem Ahmad, and Raheem Sarwar. Model optimization techniques in personalized federated learning: A survey. *Expert Systems with Applications*, 243:122874, 2024.

[60] Felix Sattler, Arturo Marban, Roman Rischke, and Wojciech Samek. CFD: Communication-Efficient Federated Distillation via Soft-Label Quantization and Delta Coding. *IEEE Transactions on Network Science and Engineering*, 9(4):2025–2038, July 2022.

[61] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3400–3413, September 2020.

[62] Osama Shahid, Seyedamin Pouriyeh, Reza M. Parizi, Quan Z. Sheng, Gautam Srivastava, and Liang Zhao. Communication efficiency in federated learning: Achievements and challenges, 2021.

[63] Yashothara Shanmugarasa, Hye-young Paik, Salil S. Kanhere, and Liming Zhu. A systematic review of federated learning from client's perspective: challenges and solutions. *Artificial Intelligence Review*, 56(2):1773–1827, 2023.

[64] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423 & 623–656, 1948.

[65] Jiangang Shu, Weizhe Zhang, Ying Zhou, Zhengtao Cheng, and Laurence T. Yang. FLAS: Computation and Communication Efficient Federated Learning via Adaptive Sampling. *IEEE Transactions on Network Science and Engineering*, 9(4):2003–2014, July 2022.

[66] Dinesh Soni and Neetesh Kumar. Machine learning techniques in emerging cloud computing integrated paradigms: A survey and taxonomy. *Journal of Network and Computer Applications*, 205:103419, 2022.

[67] Allan Souza, Luiz Bittencourt, Eduardo Cerqueira, Antonio Loureiro, and Leandro Villas. Dispositivos, eu escolho vocês: Seleção de clientes adaptativa para comunicação eficiente em aprendizado federado. In *Anais do XLI Simpósio Brasileiro*

de Redes de Computadores e Sistemas Distribuídos, pages 1–14, Porto Alegre, RS, Brasil, 2023. SBC.

[68] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A survey of optimization methods from a machine learning perspective, 2019.

[69] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[70] Anish K. Vallapuram, Pengyuan Zhou, Young D. Kwon, Lik Hang Lee, Hengwei Xu, and Pan Hui. HideNseek: Federated Lottery Ticket via Server-side Pruning and Sign Supermask, June 2022. arXiv:2206.04385 [cs].

[71] Bin Wang, Jun Fang, Hongbin Li, and Bing Zeng. Communication-Efficient Federated Learning: A Variance-Reduced Stochastic Approach With Adaptive Sparsification. *IEEE Transactions on Signal Processing*, 71:3562–3576, 2023.

[72] Luping Wang, Wei Wang, and Bo LI. CMFL: Mitigating Communication Overhead for Federated Learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 954–964, July 2019. ISSN: 2575-8411.

[73] Yansheng Wang, Yongxin Tong, Zimu Zhou, Ruisheng Zhang, Sinno Jialin Pan, Lixin Fan, and Qiang Yang. Distribution-Regularized Federated Learning on Non-IID Data. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 2113–2125, April 2023. ISSN: 2375-026X.

[74] Zeqin Wang, Ming Wen, Yuedong Xu, Yipeng Zhou, Jessie Hui Wang, and Liang Zhang. Communication compression techniques in distributed deep learning: A survey. *Journal of Systems Architecture*, 142:102927, 2023.

[75] Hui Wen, Yue Wu, Jia Hu, Zi Wang, Hancong Duan, and Geyong Min. Communication-Efficient Federated Learning on Non-IID Data Using Two-Step Knowledge Distillation. *IEEE Internet of Things Journal*, 10(19):17307–17322, October 2023.

[76] Mang Ye, Xiuwen Fang, Bo Du, Pong C. Yuen, and Dacheng Tao. Heterogeneous Federated Learning: State-of-the-art and Research Challenges. *ACM Comput. Surv.*, 56(3), October 2023. Place: New York, NY, USA Publisher: Association for Computing Machinery.

[77] Kai Yue, Richeng Jin, Chau-Wai Wong, and Huaiyu Dai. Communication-Efficient Federated Learning via Predictive Coding. *IEEE Journal of Selected Topics in Signal Processing*, 16(3):369–380, April 2022.

[78] Kai Yue, Richeng Jin, Chau-Wai Wong, and Huaiyu Dai. Federated Learning via Plurality Vote. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2022.

[79] Tianyu Zhang, Tianhan Gao, and Qingwei Mi. A Stochastic Asynchronous Gradient Descent Algorithm with Delay Compensation Mechanism. In *2022 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 167–171, July 2022.

[80] Wenyu Zhang, Xiumin Wang, Pan Zhou, Weiwei Wu, and Xinglin Zhang. Client selection for federated learning with non-iid data in mobile edge computing. *IEEE Access*, 9:24462–24474, 2021.

[81] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv*, 2018.

[82] Zihao Zhao, Yuzhu Mao, Yang Liu, Linqi Song, Ye Ouyang, Xinlei Chen, and Wenbo Ding. Towards efficient communications in federated learning: A contemporary survey, 2022.

[83] Xiaokang Zhou, Xuzhe Zheng, Xuesong Cui, Jiashuai Shi, Wei Liang, Zheng Yan, Laurence T. Yang, Shohei Shimizu, and Kevin I-Kai Wang. Digital Twin Enhanced Federated Reinforcement Learning With Lightweight Knowledge Distillation in Mobile Networks. *IEEE Journal on Selected Areas in Communications*, 41(10):3191–3211, October 2023.

Appendix A

Generating Non-IID Data for FL

In this appendix, we outline methods for generating non-IID data distributions among clients using the Dirichlet distribution. This includes an implementation guide with the FedArtML library and a custom code example, both aimed at controlling data heterogeneity.

A.1 Generating non-IID Data Using Dirichlet Distribution

The challenge of simulating real-world FL scenarios involves creating non-IID data distributions among clients. Here, we use the Dirichlet distribution to control data heterogeneity, providing an implementation guide with the FedArtML library and a custom code snippet for tailored distribution.

A.1.1 Generating non-IID Data with Dirichlet Distribution

non-IID data distributions can be simulated effectively using the Dirichlet distribution, which allows for the control over the degree of data heterogeneity among clients through the concentration parameter α. A lower value of α results in higher data disparity among clients, simulating a more challenging federated learning environment.

Method:

The following steps outline the general procedure for generating non-IID data among clients using the Dirichlet distribution. This approach ensures controlled data partitioning, allowing for varying levels of heterogeneity across clients. The method can be adapted to different datasets and federated learning environments as needed:

1. Initialize the necessary parameters for data distribution among clients, including the total number of clients and the desired α value for the Dirichlet distribution, which controls the level of data heterogeneity.

2. Assign indices for the training and testing datasets based on the Dirichlet distribution, ensuring that each client receives a specific portion of the data.

3. Extract the data subsets for each client according to the assigned indices, maintaining the intended level of non-IID distribution across the clients.

This method provides flexibility by allowing researchers to experiment with various levels of data skewness, enabling more realistic simulations for FL research. Additionally, the tool incorporates metrics like Jensen-Shannon Distance (JSD) and Hellinger Distance (HD) to quantify the degree of non-IID-ness, ensuring that the heterogeneity across clients is measurable and consistent [25].

A.1.2 Custom Implementation for non-IID Data Generation

In addition to the standard approach, a custom method was developed to further refine the data distribution process. This method ensures that the indices for both training and testing datasets are generated based on the Dirichlet distribution, thereby controlling the degree of heterogeneity among clients.

Method:

The process begins by defining the type of distribution to be used for partitioning the data, such as the Dirichlet distribution. Once the distribution is selected, the following steps are carried out:

1. **Define the distribution type**: The Dirichlet distribution is chosen as the basis for generating indices that will be used to partition the dataset.

2. **Generate indices for training and testing data**: For both the training and testing datasets, indices are generated using a method that allocates data samples according to the specified Dirichlet distribution. This ensures that the degree of non-IID-ness is controlled by the concentration parameter α.

    - For each dataset, a function is called to generate these indices, ensuring that the data is distributed across clients in a way that reflects the chosen heterogeneity.

3. **Extract and assign data**: Based on the generated indices, the corresponding subsets of data and their associated labels are extracted for both the training and testing sets.

4. **Return the partitioned data**: Finally, the subsets of data for training and testing are returned, ready for use in the federated learning process, with each client receiving a portion of the data according to the previously defined distribution.

This custom implementation allows for precise control over how data is distributed among clients, generating indices based on the Dirichlet distribution to achieve the desired level of heterogeneity. The method ensures that the training and testing data subsets are partitioned in a way that reflects real-world scenarios, where data is often non-IID.

These approaches to generating non-IID data using the Dirichlet distribution provide two effective methods to simulate realistic federated learning (FL) environments.

The FedArtML tool, which relies on predefined functions, offers a straightforward and efficient way to distribute data among clients based on the Dirichlet distribution. It allows for control over heterogeneity through the concentration parameter α, providing flexibility in the distribution of data. This method is particularly useful for quickly setting up non-IID scenarios with minimal customization and is ideal for general use cases where a simple, predefined structure is sufficient.

In contrast, the custom implementation goes a step further by allowing more granular control over the partitioning process, enabling the generation of indices for both training and testing datasets based on specific requirements. While this approach offers flexibility, the FedArtML tool stands out due to its integration of advanced metrics like Jensen-Shannon Distance (JSD) and Hellinger Distance (HD). These metrics provide a quantifiable measure of the degree of non-IID-ness among clients, ensuring that the data distribution is not only controlled but also measurable.

This capacity to quantify heterogeneity makes FedArtML the superior tool for most applications, as it allows researchers to fine-tune the non-IID characteristics with precision, offering better insights into the behavior of FL algorithms in diverse and imbalanced data scenarios.

Appendix B

Computational Resources for Simulation

This appendix provides detailed information about the computational resources employed for the simulations conducted in this research. The simulations were carried out using both local and cloud-based infrastructure to ensure robustness and scalability of the experiments.

B.1 Computational Resource

The simulations and testing were conducted on a dedicated local machine with high computational capacity. This machine was specifically selected to ensure efficient processing and handling of the large-scale datasets used for FL simulations. The specifications of the machine are as follows:

Table B.1: Specifications of the local machine used to simulations.

Specification	**Details**
CPU Model	Intel(R) Xeon(R) Silver 4210R @ 2.40 GHz
CPU Cores and Threads	20 cores, 40 threads
CPU Frequency	Max: 3.20 GHz, Min: 1.00 GHz
Cache Memory	L1: 640 KiB, L2: 20 MiB, L3: 27.5 MiB
RAM	251 GiB total, 236 GiB available
Primary Storage (SSD)	98 GiB
Secondary Storage (HDD)	5.5 TiB
Network Attached Storage (NAS)	8.2 TiB
CPU Load (Idle)	93.7% idle, 5.9% user
Operating System	Ubuntu 20.04 LTS (64-bit)
Virtualization	VT-x enabled

The specifications outlined in Table B.1 detail the computational resources of the machine used for simulations in this dissertation, which was instrumental in conducting the simulations for this research. These capabilities encompass a high-performance processor,

extensive RAM, and substantial storage capacity, ensuring the efficient execution of complex computational tasks inherent to our simulation processes. Based on the hardware configuration of the system, the theoretical floating-point performance can be estimated. The Intel Xeon Silver 4210R CPU supports AVX-512 instructions, which can execute 16 floating-point operations per cycle in single precision (8 in double precision). Given the processor's specifications, the theoretical FLOPS performance is calculated as follows:

- **Number of cores:** 20
- **FLOP per cycle (single precision):** 16
- **Processor frequency:** 3.20 GHz

Thus, the estimated theoretical performance is:

$$\text{FLOPS (single precision)} = 20 \times 16 \times 3.2 \times 10^9 = 1.024\,\text{TFLOPS}$$

In double precision, the theoretical performance is:

$$\text{FLOPS (double precision)} = 20 \times 8 \times 3.2 \times 10^9 = 0.512\,\text{TFLOPS}$$

These values represent the peak theoretical performance of the system.

B.2 Cloud-Based Infrastructure: Digital Ocean

For scalable simulations and larger experiments, cloud-based resources from Digital Ocean were utilized. The primary configuration of the droplets used is detailed below:

- Memory (RAM): 4 GB
- CPUs: 2 virtual CPUs
- Disk: 80 GB SSD
- Network Transfer: 4 TB

The selection of Digital Ocean droplets was driven by their optimal balance of computational power and cost-efficiency, enabling extensive simulations without substantial financial burden. These droplets are well-adapted for distributed computing tasks, crucial for federated learning simulations.

The combination of a powerful local machine and flexible cloud-based droplets provided a comprehensive computational environment that supported all stages of the research, from development and testing to large-scale simulation.