



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Reducing Costs Using Normalization in Federated Learning in Heterogeneous
Data Distributions

Flávio Vieira

Orientador
Carlos Alberto Vieira Campos

RIO DE JANEIRO, RJ - BRASIL
ABRIL de 2024

Reducing Costs Using Normalization in Federated Learning in Heterogeneous
Data Distributions.

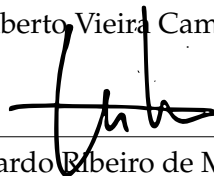
Flávio Vieira

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO
DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM IN-
FORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
(UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSI-
NADA.

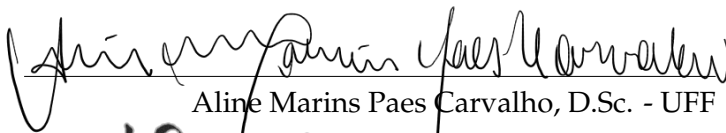
Aprovada por:



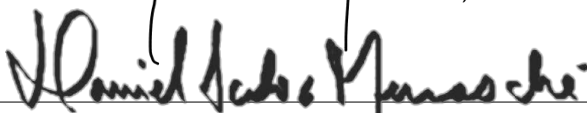
Carlos Alberto Vieira Campos, D.Sc. - UNIRIO



Carlos Eduardo Ribeiro de Mello, D.Sc - UNIRIO



Aline Marins Paes Carvalho, D.Sc. - UFF



Daniel Sadoc Menasche, D.Sc. - UFRJ

RIO DE JANEIRO, RJ - BRASIL
ABRIL de 2024

Catálogo informatizada pelo(a) autor(a)

V665 Vieira, Flávio
 Reducing Costs Using Normalization in Federated
Learning in Heterogeneous Data Distributions / Flávio
Vieira. -- Rio de Janeiro, 2024.
 70

 Orientador: Carlos Alberto Vieira Campos.
 Dissertação (Mestrado) - Universidade Federal do Estado
do Rio de Janeiro, Programa de Pós-Graduação em Informática,
2024.

 1. Federated Learning. 2. Heterogeneous Data. 3.
Normalization. I. Campos, Carlos Alberto Vieira, orient.
II. Título.

Dedico essa qualificação de dissertação a minha família: pais, esposa e filho.

Agradecimentos

Agradeço primeiramente aos meus pais Sebastião e Suely por me incentivarem a seguir o caminho do estudo e aprendizado por toda a minha vida e por serem os meus primeiros professores.

Agradeço a minha esposa e filho pelo apoio e por comemorar cada conquista na minha caminhada durante o processo do mestrado. Sem vocês a jornada não faria sentido.

Agradeço meu amigo Nicomar Fernandes de Oliveira cuja dissertação de mestrado me inspirou a entrar no programa de mestrado.

Agradeço meu professor e orientador, Carlos Alberto Vieira Campos, não só pelos ensinamentos que estão sendo cruciais como pela parceria ao longo do tempo. Sua dedicação, cuidado e atenção pelo nosso trabalho são fora do comum contribuindo de forma decisiva para o engrandecimento do trabalho e os resultados alcançados.

Agradeço também ao professor Pedro Nuno de Souza Moura cujo trabalho e ensinamentos na disciplina de Aprendizado Profundo muito contribuiu como fonte de inspiração e evolução do trabalho.

Agradeço a todo o corpo docente do PPGI da UNIRIO pelo ensino de qualidade nas disciplinas do programa e pela preocupação em preparar o aluno para superar as adversidades e desafios impostos durante o período do mestrado. Certo de que todos vocês tem em mim um admirador do vosso trabalho.

Na empresa em que trabalho agradeço a Renato Gonçalves da Silva pelo apoio e incentivo durante a caminhada em curso.

Agradeço a UNIRIO pela oportunidade e a toda comunidade científica pelas contribuições recebidas no desenvolvimento do trabalho.

Vieira, Flávio. **Reducing Costs Using Normalization in Federated Learning in Heterogeneous Data Distributions**. UNIRIO, 2024. 70 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

Num mundo cada vez mais conectado, tecnologias como smartphones, 5G, drones, Internet das Coisas e Cidades Inteligentes trazem novos desafios e oportunidades. O aumento dos dados coletados por esses dispositivos e sua facilidade de acesso permitem a utilização de técnicas de aprendizado de máquina para fornecer serviços inteligentes e de qualidade. Considerando o acesso distribuído desses serviços aos dados, o uso de Aprendizado Federado é uma ótima opção que permite o processamento descentralizado de aprendizado de máquina com maior segurança. No entanto, o acesso do cliente a dados distribuídos heterogeneamente gera um impacto no aprendizado federado, reduzindo a acurácia de teste e elevando os custos de comunicação. Para resolver esses problemas, apresentamos um novo método chamado *Federated Learning with Weight Standardization on Convolutional Neural Networks* (FedWS) que se utiliza de normalização dos pesos no treinamento local em redes neurais convolucionais, otimizando os gradientes de treinamento que reduzem o impacto da divergência de pesos causados pelas distribuições heterogêneas nas tarefas de treinamento federado. Os resultados mostraram que nosso método obteve resultados superiores em tarefas de classificação de imagens na ordem de 1,36% a 5,0% de precisão do teste em diferentes níveis de heterogeneidade. Seu comportamento mostrou redução dos efeitos da divergência em níveis mais elevados de heterogeneidade e redução de custos de comunicação variando de 20% a 100% devido a sua rápida convergência, mostrando-se mais adequada para o uso em dispositivos moveis com limitações de recursos computacionais.

Palavras-chave: Aprendizado Federado, Dados Heterogêneos, Transformação de Dados, Classificação de Imagens.

ABSTRACT

In an increasingly connected world, technologies such as smartphones, 5G, drones, the Internet of Things, and Smart Cities bring new challenges and opportunities. The increase in data collected by these devices and their ease of access allows the use of machine learning techniques to provide intelligent and quality services. Considering these services' distributed access to data, using Federated Learning is a great option that allows for decentralized machine learning processing with greater security. However, client access to heterogeneously distributed data impacts federated learning, reducing test accuracy and increasing communication costs. To address these problems, we present a new method called Federated Learning with Weight Standardization on Convolutional Neural Networks (FedWS) that uses normalization on weights in local training on convolutional neural networks, optimizing the training gradients and reducing the impact of weight divergence caused by heterogeneous distributions on federated training tasks. Results showed that our method got superior results on image classification tasks in the order of 1.36% to 5.0% of test accuracy at different levels of heterogeneity. Their behavior showed a reduction in the effects of divergence at higher levels of heterogeneity and communication cost reduction ranging from 20% to 100%, due to fast convergence and being more suitable for use in mobile devices with computational resource limitations.

Keywords: Federated Learning, Heterogeneous Data, Data Transformation, Image Classification.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem	2
1.3	Objective	2
1.4	Contribution	3
1.5	Organization	4
2	Theoretical Foundation	5
2.1	Deep Learning	5
2.1.1	Representing the Artificial Neurons	5
2.1.2	Architecture of Artificial Neural Networks	8
2.1.3	Convolutional Neural Networks	12
2.2	Federated Learning	12
2.3	Resource Costs Related to Federated Learning	15
2.4	Heterogeneous Distribution of Data and non-IID	15
2.5	Impact of Weight Divergence on FL	17
2.6	Related Work	20
3	Proposed Solution	22
3.1	Proposed Method	22
4	Experiments	27
4.1	Research Questions	27
4.2	Setup and Datasets	29
4.3	Experiment I: Classifying Images in Heterogeneous Data	30
4.4	Experiment II: Federated Learning Communication Cost	35
4.5	Experiment III: Impact of Heterogeneous Distributions on Convolutional Layers of CNN	37
4.6	Experiment IV: Computing Weight Divergence	38
4.7	Experiment V: Losses and Gradient Behavior	41
4.8	Experiment VI: Comparison with the Centralized Scenario	43
4.9	Experiment VII: Increasing Local Rounds in Federated Learning	44
4.10	General Discussion	48
5	Conclusion	51
5.1	Limitations	52
5.2	Future Works	52

List of Figures

2.1	Sigmoid curve.	7
2.2	Tanh curve.	7
2.3	Relu function.	8
2.4	Example of ANN Architecture.	8
2.5	Deep Learning Training Process.	9
2.6	Gradient Descent.	10
2.7	Example of Convolution Operation with Kernel Size 3 on a 5×5 Image.	12
2.8	Federated Learning Training Process.	13
2.9	Federated Learning Round.	14
2.10	CIFAR10 example with label size imbalance of level 2, with each client having 20 images as data sample.	16
2.11	CIFAR10 example illustrating different levels of label distribution imbalance.	16
2.12	Comparison of the weight divergence behavior between IID Settings and Non IID Settings on FL training processes.	17
2.13	Distribution of Data by Classes in 100 Clients Considering EMD Values 0, 0.37, 0.71, 1.08 and 1.41.	19
3.1	FedWS Normalization on Local Learning.	23
3.2	Detailed global and local iterations to perform FL.	24
4.1	Convergence results of test accuracy in image classification task on MNIST, CIFAR10 and EuroSAT.	33
4.2	Percentage difference between training and testing results of observed normalization techniques.	34
4.3	Histogram of Convolutional Layer on Round 1 of FL on EuroSAT Dataset	38
4.4	Weight Divergence Results on Different Datasets, Techniques, and Levels of EMD at round 1 of FL	39
4.5	Loss on FL with MNIST, CIFAR10 and EuroSAT Dataset on two distributions	40
4.6	Projection of weight divergence considering the gradients of NN, BN, LN, GN and FedWS in 10 first FL Rounds, using Lenet5 CNN architecture a learning rate of 0.01 on datasets CIFAR10 and EuroSAT with an EMD of 0.71 and 1.41. Results are in logarithmic scale.	42
4.7	Convergence analysis in classification task on EuroSAT	45

4.8	Representation of contribution of local and global iterations to achieve a target test accuracy.	47
4.9	Factors that impact the number of local iterations in achieving a target accuracy.	48

List of Tables

2.1	Notation Symbols on (2.10)	18
3.1	Expected influence of FedWS on weight divergence during FL rounds, considering the impact on η , EMD and $g_{\max}$ components, considering fixed values of η and EMD.	26
4.1	Summary of the definition of experiments	28
4.2	Datasets used on experiments.	29
4.3	CNN Model Architectures	30
4.4	Federated Test Accuracy(%) Results on Different Datasets, Techniques and Levels of EMD	31
4.5	ResNet18 Image Classification Test Accuracy(%) Results, using 200 rounds of FL, 10 clients, and 5 local epochs	32
4.6	Total Communication Cost per Round for CNN Architectures	35
4.7	FL Communication Costs on Different Target Test Accuracies, Techniques, and CNN Architectures.	36
4.8	Architecture of LeNet5.	37
4.9	Gradient Variation on Different Techniques and Datasets using Samples with EMD 0.71 and 1.41 for FL and Lenet5 Architecture for Local Training	41
4.10	FedWS vs Centralized Training	43
4.11	Gradients Variation on Centralized Training	44
4.12	Test accuracy percentage in the EuroSAT Dataset using 10 local epochs, showing the comparative gain with 5 epochs	44
4.13	Communication Cost (MB) for Normalization Techniques to Achieve 75% Test Accuracy with EuroSAT Dataset	46
4.14	Computational Cost (number of synchronizations) to Achieve 75% of Test Accuracy on EuroSAT Dataset	46
4.15	Impact on computational and communication costs of rising local epochs from 5 to 10 on FL using FedWS on Dataset EuroSAT.	48

List of Abbreviations and Acronyms

ANN	<i>Artificial Neural Network</i>
CNN	<i>Convolutional Neural Network</i>
BN	<i>Batch Normalization</i>
DL	<i>Deep Learning</i>
EMD	<i>Earth Mover's Distance</i>
FedAvg	<i>Federated Learning Average</i>
FedWS	<i>Federated Learning Algorithm with Weight Standardization on Convolutional Layers</i>
FL	<i>Federated learning</i>
GN	<i>Group Normalization</i>
ICS	<i>Internal Covariance Shift</i>
IID	<i>Independent and Identically Distributed</i>
IoT	<i>Internet of Things</i>
LN	<i>Layer Normalization</i>
ML	<i>Machine Learning</i>
non-IID	<i>non Independent and Identically Distributed</i>
RSU	<i>Road Side Units</i>
SGD	<i>Stochastic Gradient Descent</i>
WS	<i>Weight Standardization</i>

1. Introduction

1.1 Motivation

According to projections of Fortune Business Insights, the market size of Internet of Things (IoT) ¹ and Artificial Intelligence ² has a projected annual growth rate of 26.1% and 21.6%, respectively, until the year 2030, considering estimates made in 2022. We live in an environment in which people and objects are increasingly connected, and advances in concepts and technologies, such as smartphones, 5G networks, drones, IoT, and smart cities, bring new challenges and possibilities. As the number of sensors around us grows, so does the amount of data available for the design and analysis of new services [12].

Machine learning (ML) and Deep Learning (DL) technologies that can use available data to deliver intelligent services have been highlighted. Image classification using deep learning and convolutional neural networks (CNN) has an essential role and wide application on edge device services in different areas like emergency monitoring [23], agriculture [11], deforestation detection [16] and construction [59], using aerial and satellite images. However, one of the challenges emerging in the construction of these services is the data privacy obtained through distributed and interconnected devices and sensors [58].

Considering the distributed nature of these services, traditional deep learning methods require direct access to data collected on clients or devices and the need to transfer data to a central that would perform machine learning using CNN, leading to problems with privacy and network overload, compromising the service being provided [51].

Federated learning (FL) is a technique presented by MacMahan et al. (2017) [38] which proposes a type of decentralized learning using artificial neural networks (ANN), taking advantage of clients' processing capacity and their proximity to the data. A central server controls the global learning process, and edge devices execute the local learning task and transfer to the central server only what is learned locally by the ANN, represented by weights and biases, making data traffic unnecessary and reducing the privacy impact. The central server carries out an aggregation process of what was received to obtain the final model of the learning process. FL applications span a broader spectrum, encompassing areas such as transportation, healthcare, mobile devices, and IoT [2].

Communication cost is a relevant point that impacts the services available on

¹<https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307>

²<https://www.fortunebusinessinsights.com/industry-reports/artificial-intelligence-market-100114>

edge devices. The amount of information transmitted can lead to slowdowns or a drop in the quality of services, as the resource capacity of mobile devices is limited. In the context of FL, this problem is materialized by the increased frequency of sending weights to clients, caused by the low convergence of models, which can lead to low engagement of training participants [36].

1.2 Problem

The presence of heterogeneous data distributions between FL clients is a problem to be overcome with major impacts on FL and can lead to increased communication costs and processing bottlenecks [42]. In real-world applications, clients usually have access to heterogeneously distributed data, leading to low test accuracy and stagnation in training. The side effects caused by this problem are highlighted by [63] as one of the significant challenges facing FL today, presenting a taxonomy of approaches to address this problem.

Among these approaches, structure adoption techniques, represented by adaptive optimizers and normalization techniques, can accelerate the convergence of ANN training in FL and reduce communication costs in heterogeneous data distributions. However, adaptive optimizers require transmitting more information between clients and the central server for their operation, while normalization techniques do not require additional transmissions. Two normalization techniques are highlighted for presenting good results on the accuracy test in this scenario: Batch Normalization (BN) [18] and Group Normalization (GN) [56]. However, the first one does not perform appropriately on the FL task because of the combination of BN and the aggregation operation that generally results in inadequate values of weights and biases [63]. In turn, this situation does not affect GN and usually performs better than BN. In this context, the cost of communication in FL when using normalization techniques must also be taken into consideration, due to the capacity characteristics of mobile devices. This leads us to propose two research questions:

- **RQ1** - How do the state of the art in FL normalization techniques behave considering test accuracy and communication cost in a scenario with heterogeneous data distribution?
- **RQ2** - How we can reduce the communication costs in FL while maintaining good accuracy performance on heterogeneous data distributions?

1.3 Objective

To answer the research questions defined previously, we propose a solution that considers reducing the impact of heterogeneous data distribution on FL and communication costs using a normalization approach.

In this context, we propose the Federated Learning Algorithm with Weight Standardization on Convolutional Layers (FedWS), a new approach that mitigates heterogeneous data's harmful effects in FL, reducing the impact on test accuracy and communication costs, with an increased convergence even in challenging

situations. The proposed technique uses weight standardization (WS) [44] applied to convolutional layers in local FL training, smoothing the behavior of gradients and errors (losses) in the federated context and reducing the impact of divergence between the weights noted in cases with heterogeneous datasets.

To the best of our knowledge, this work is the first to theoretically relate the heterogeneity of data distributions in FL using techniques that propose to smooth the gradients and losses, including experiments with different levels of data heterogeneity and considering the cost of communication in FL.

1.4 Contribution

As contributions during the work, we can highlight the following items:

- Proposal for the FedWS technique that aims to improve the behavior of weights generated by FL using standardization on client local training when data is heterogeneously distributed.
- We empirically showed that FedWS presents a more stable behavior on heterogeneous data including scenarios using more datasets and more complex neural network architecture with varying degrees of data heterogeneity in FL training for image classification.
- We showed evidence that techniques that reduce the value of gradients can reduce the impact of weight divergence on FL in heterogeneous distribution scenarios.
- We have empirically demonstrated that the FedWS reduces the initial impact on weight divergence in the first round of FL and that on subsequent rounds your small gradients and losses reduce the impact of data distribution, resulting in more stable behavior of data with different levels of heterogeneity.
- We have empirically demonstrated that the FedWS gradient's behavior is similar to the centralized approach, resulting in accuracy results closer to centralized training.
- We show that FedWS reduction in communication cost of FedWS maintains both in simpler architectures as well as in more complex neural network architectures.
- We empirically demonstrated that FedWS provides communication costs that other techniques only obtain when applying an increase in local epochs, at the cost of an increase in computational cost. In this way, FedWS achieves better results using fewer local epochs, therefore, it is more suitable for use on devices with limited computing resources.

Furthermore, as contributions to the work, two publications [54] and [53] were generated.

1.5 Organization

This dissertation is structured as follows: Chapter 2 presents an overview of the necessary theoretical foundations for understanding the context in this work, including Deep Learning, Federated Learning, Costs Related to Federated Learning, Heterogeneous Data Distributions, and the Impact of Weight Divergence on FL. At the end of the chapter, are presented the related works that use normalization in local FL training to reduce the impact of heterogeneous data distributions in FL. In turn, Chapter 3 details the technical proposal, including theoretical definitions that advance the treatment of the problem presented and define research questions to evaluate how the solution behaves compared to related work. Chapter 4 presents the experiments carried out to answer the research questions and the configurations used in addition to presenting the results and discussion about them. Finally, Chapter 5 presents the conclusion, a summary of the results, and proposals for future work.

2. Theoretical Foundation

This chapter presents some concepts necessary for understanding the work. We will give a brief description of the following subjects: **Deep Learning**, due to its use in the local FL training stage, a minimum understanding of how it works and some of its components such as weights and biases, activation functions, losses, gradients, and normalization techniques in addition to the ANN/CNN architecture. **Federated Learning**, as the central theme of the work the functioning of FL will be highlighted, including its components, related costs, how heterogeneous distributions are characterized in FL, and how they affect its performance. At the end of the chapter, it is shown how related works propose to deal with data heterogeneity problems in FL, focusing on the use of normalization in the local training stage.

2.1 Deep Learning

Machine Learning (ML) is a set of techniques based on statistical models designed so that machines can perform tasks autonomously based on the analysis and observation of input data involved in the activity [35]. Much of the efficiency of traditional machine learning algorithms largely depends on the quality of how the input data is represented. Properly processing input data, also called feature engineering, despite the number of existing techniques for this, tends to require human effort with knowledge about the domain to which the data refers [43].

Deep learning (DL) is a set of techniques that evolved based on the concepts of neural networks, using the concept of neurons and transformation layers. These layers are composed like a network that can have a very large number of layers, which is why they are called deep [22]. One of the great advances in deep learning techniques is their ability to automatically extract features from input data, reducing the need for feature engineering [7].

2.1.1 Representing the Artificial Neurons

The first evolution towards a concept of artificial neurons occurred in 1943 by McCulloch-Pitts [37] who defined how a neuron would work. They built the first mathematical representation of a neuron called the Threshold Logic Unit (TLU). The TLU was designed to replicate the functioning of a neuron at a high level, receiving values as input and returning a Boolean as output. In the TPU, a set of input signals $X_1, X_2, \dots, X_n$ is associated with a set of weights ω , and the product of these values is computed to define the activation:

$$a = \sum_{i=1}^n X_i \cdot \omega_i \quad (2.1)$$

A threshold t is also defined that will be used to evaluate the activation and generate the value of y as

$$y = \begin{cases} 1, & \text{if } a \geq t \\ 0, & \text{if } a < t \end{cases} \quad (2.2)$$

Later in 1958, Rosenblatt [45] created a neuron representation model called Perceptron with the ability to receive a binary input from multiple neurons, aggregate the received inputs through a sum of weights, and generate an output that will be sent to other neurons if they exceed a threshold.

The sum of the input weights is calculated as:

$$\sum_{i=1}^n \omega_i \cdot x_i, \quad (2.3)$$

where ω_i is the weight of a given entry x_i . In addition to this sum, a threshold called bias (b) is defined, which aims to ensure that the input data and weights can be calibrated to adjust the output. Considering this information, the output $g(x)$ is defined as:

$$g(x) = \begin{cases} 1, & \text{if } \omega \cdot x + b > 0 \\ 0, & \text{if } \omega \cdot x + b \leq 0. \end{cases} \quad (2.4)$$

Perceptrons can only work with binary inputs and outputs, making it unfeasible to use them where the inputs are continuous, non-integer values. One of the evolutions that allowed the evolution of neurons to handle non-linear values was activation functions. The activation functions are used to generate nonlinear relationships between the input and the output and the most used functions are Sigmoid, Hyperbolic tangent, and Relu [22].

Sigmoid Function

The sigmoid [15] function aims to receive a value as input and return a value between 0 and 1 as output, being widely used in logistic regression and classification problems and is defined as:

$$\sigma(z) = \frac{1}{(1 + e^{-z})}, \quad (2.5)$$

where $z = \omega \cdot x + b$ and e is a constant starting at 2.718. The sigmoid curve is represented in Figure 2.1.

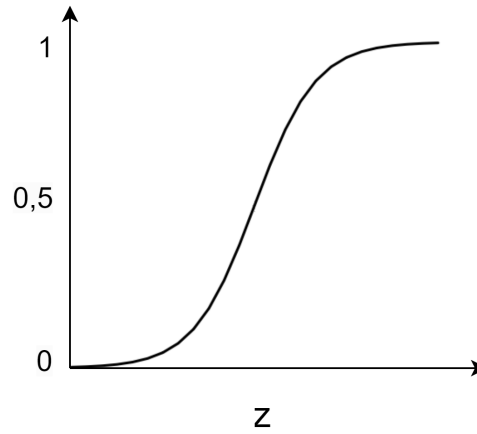


Figure 2.1: Sigmoid curve.

This type of activation function is most commonly used in the final layers of a neural network architecture for classification problems. At extreme values of z , neurons that use sigmoid will output values 0 or 1, causing updates to the values of ω and b to have little effect, falling into a situation of neuron saturation and possible training stoppage.

TanH Function

Another activation function used is the hyperbolic tangent (Tanh), which works similarly to the sigmoid and is defined as:

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \quad (2.6)$$

The difference between sigmoid and tanh is that the latter has outputs between -1 and 1, as shown in Figure 2.2 and they tend to be close to 0, without presenting such extreme values, tending to be less affected by saturation of neurons [22].

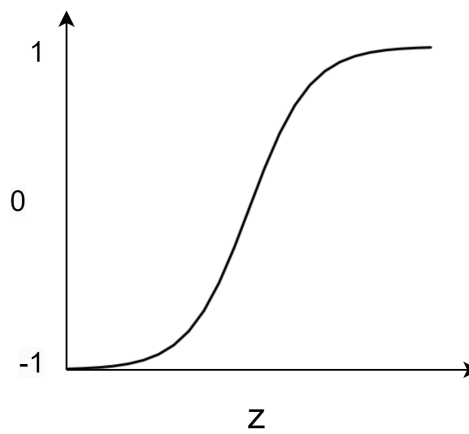


Figure 2.2: Tanh curve.

ReLU Function

The rectified linear unit or ReLU [1] function presents an evolution in relation to sigmoid and Tanh. This function is defined as $a = \max(0, z)$, so that if z is

positive it returns z and if z is 0 or negative it returns 0, as illustrated in Figure 2.3. This type of non-linear behavior is very beneficial for use in deep learning architectures. According to [29], the use of ReLU accelerates the training of the neural network compared to sigmoid, in addition to reducing the saturation of neurons. Furthermore, the derivative operations used on ReLU can be computed using simpler mathematical operations than other activation functions.

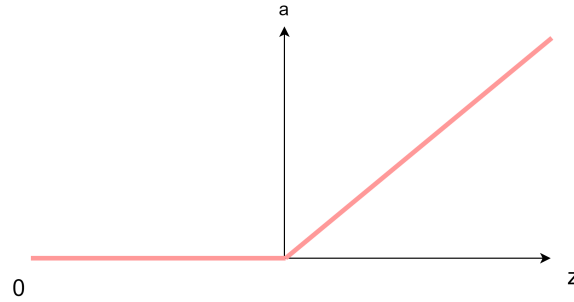


Figure 2.3: Relu function.

2.1.2 Architecture of Artificial Neural Networks

Using simple neurons is not enough to learn more complex relationships between data, therefore the neurons are grouped in a parallel manner. The layered composition of neurons is a characteristic of artificial neural networks (ANN) [8]. Figure 2.4 shows a diagram of an example of a deep neural network architecture typically composed of an input layer, several hidden layers, and an output layer.

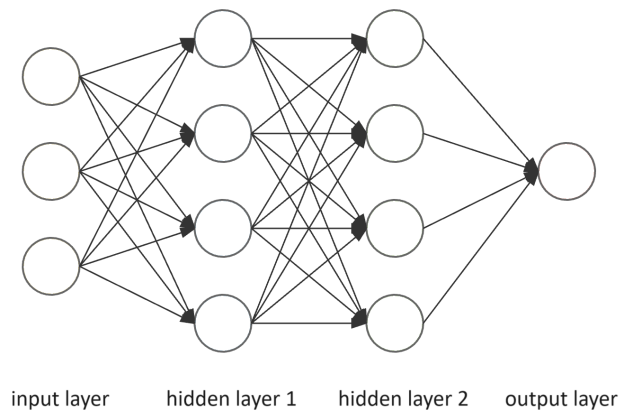


Figure 2.4: Example of ANN Architecture.

The learning process, exemplified in Figure 2.5 in a classification problem, consists of submitting the input data to an ANN that will extract the characteristics of the data [25] generating the values of the parameters (weights and biases) for each neuron in the ANN layers, using the appropriate activation functions. At the end of the pass through the network, the outputs of the last layer will be generated as a probability vector and compared with a template containing the template for calculating the error between what the machine learned and the ground truth. After this calculation, the errors will be submitted to an optimization function, also called a gradient, which will calculate the correction that must be made to the

ANN parameters to reduce the error. These adjustments will be propagated in an algorithm or backward step that adjusts the neuron parameters from the end to the beginning. The final objective is to bring the error closer to zero so that the network obtains values very close to the ground truth.

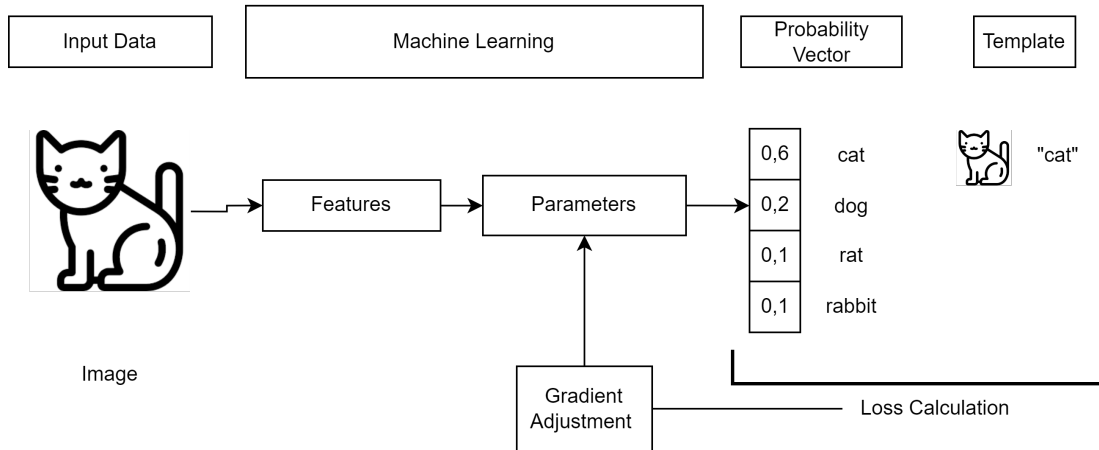


Figure 2.5: Deep Learning Training Process.

To better understand this process, it is important to understand how some components of it work.

Objective and Optimization Functions

To measure how much the y -values obtained by learning differ from the actual value typically involves the use of cost or objective functions. Normally the cost or loss functions are identified as $l(w; b)$ and are intended to calculate the error for a given training round. The most used cost functions are **quadratic cost** and **crossentropy** [14].

The quadratic cost function is calculated using the mean square error $l(w; b) = \frac{1}{n} \sum_{i=1}^n (t_i - y_i)^2$, where t_i is the template value and y_i is estimated by the network. The cross-entropy function $l(w; b) = -\frac{1}{n} \sum_{i=1}^n [t_i \ln(y_i) + (1 - t_i) \ln(1 - y_i)]^2$ and can avoid neuron saturation compared to the quadratic function.

Gradient descent (∇) is a way of adjusting model parameters to minimize errors/costs. It works iteratively influencing the adjustment of parameters of the form $\omega \leftarrow \omega - \eta \nabla l(w; b)$. As shown in Figure 2.6 the gradient descent takes steps in each round influenced by the value of the learning rate η . Very small values of the learning rate can make training take a long time to converge, while very large values may never reach the minimum value.

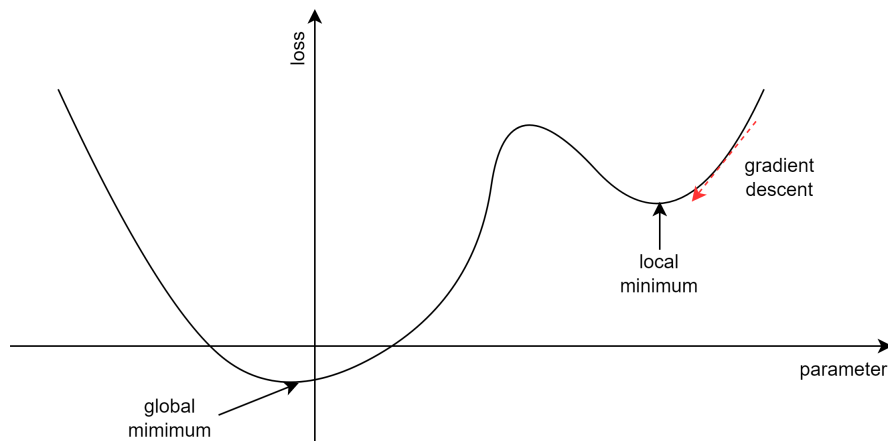


Figure 2.6: Gradient Descent.

The objective of gradient descent is to find the minimum value, but it may occur to find what is called a local minimum, which may not achieve the objective of minimizing the cost. In this case, a possible solution is to use stochastic gradient descent (SGD), which works in mini-batches composed of subsets of data submitted to the neural network.

Overfitting and Underfitting

One of the ways to validate machine learning is to compare the accuracy performance obtained in the training dataset with the accuracy obtained in a testing dataset. The difference between these two measurements may indicate different situations depending on the result. Small values indicate that the generated model achieved a generalization with the ability to respond well to inference in real problems, also called **good fit**. If the model presents a training accuracy much higher than that of tests, this indicates that we can be in a situation of **Overfitting**, where the model cannot generalize well to test data. Deep neural networks tend to overfitting situations due to the number of parameters to be learned and the number of layers used for this purpose. In the opposite case, we have a **Underfitting** situation, where the obtained model was unable to extract a pattern from the training data to achieve a good generalization [47]. In DNN, Overfitting can be attributed to a greater complexity in the architecture of the neural network, with a much greater number of parameters compared to the complexity and quantity of the data and Underfitting can be attributed to a simpler neural network, incapable of extracting the characteristics necessary for deep learning [13].

Internal Covariance Shift, Normalization and Lipschitz Constant

One of the classic machine learning problems is called the internal covariance shift (ICS) and was observed by [49] and later by [18]. It is a change in the distributions of the internal parameters of a CNN caused by the nonuniform distribution of the data, generating values of weights and biases very distant from each other, leading the optimizer to a saturated regime, resulting in slow convergence and the stagnation of the learning process, causing the effect of vanishing gradients. One of the ways to advance in the treatment of this type of situation is to use

normalization techniques that apply transformations on parameters of neural networks to produce values with a normal distribution of mean 0 and standard deviation 1. Next, we will highlight the most known and used ones.

In 2015 [18] introduced a type of normalization called Batch Normalization (BN) that works by normalizing the parameters that move between one layer and another of a neural network. BN uses the concept of mini-batches, subdividing the set of analyzed data, and using this to take the mean and standard deviation. These values are used to adjust the parameters by taking the average of the activations of a previous layer, making a difference with the batch average, and dividing by the standard deviation. The objective is to try to centralize the parameters in a normal distribution with a mean of 0 and a standard deviation of 1. These values of mean β and standard deviation γ are learned and adjusted during training, including the effect of backpropagation for adjustment.

In 2016 [4] formalized the Layer Normalization (LN) technique, which uses the same strategy of using normalization of activations predicted by BN. However, LN works by normalizing the parameters of a layer, calculating their average and standard deviation instead of calculating these values based on batches. This addressed some characteristics of BN that obtained better results in larger batches and did not have good performance in recurrent neural networks in addition to the dependence on how the batches are arranged.

In 2018 [56] advanced with another type of normalization, called Group Normalization (GN), which originated from computer vision techniques that allowed the normalization of images acting on their RGB color channels. GN works in a similar way to LN normalizing the parameters, but divides the layer parameters into groups based on the RGB channels of the images and calculates the mean and standard deviation to perform a normalization focused on the group, obtaining relevant results compared to BN and LN.

Originally, normalization techniques such as BN are known to act on the input of layers, fixing this distribution and reducing the impact of ICS. Furthermore, recently, [48] showed that one of the factors that contribute to the improvement in situations with ICS through standardization techniques is the reduction of the Lipschitz constant of losses and gradients, leading a scenario with losses and smooth gradients. According to [39] a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ has continuous Lipschitz behavior if there is a constant L such that satisfy the equation:

$$\forall x, y \in \mathbb{R}^n, \|f(x) - f(y)\| \leq L\|x - y\|. \quad (2.7)$$

The smallest value of L that makes the inequality true is called the constant Lipschitz of f . This constant is a way to check the robustness level of a neural network when subjected to disturbances in its input layer (x) and generating variations in its output layer (y) [55]. Also, according to [28], the continuity of Lipschitz plays an essential role in preserving the structure of the feature map (feature space) in normalization methods (BN, LN, and GN) and also in the robustness of CNNs.

2.1.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of architecture introduced by [24] that uses convolutional layers to extract features. This type of operation allows obtaining useful information about data with spatial characteristics for subsequent layers of a neural network.

In CNNs, convolutions are mathematical operations normally performed on data matrices, in most cases consisting of images converted into three matrices containing the RGB values of each pixel. Each convolution has as its characteristic a kernel or filter size that will slide along the matrix that represents the image from top left to bottom right. Kernels are the weights that will be learned and updated in the future through the backpropagation mechanism.

Input Image	Kernel	Convolution Result																																											
<table><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr></table>	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	1	1	0	0	<table><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr></table>	1	0	1	0	1	0	1	0	1	<table><tr><td>4</td><td>3</td><td>4</td></tr><tr><td>2</td><td>4</td><td>3</td></tr><tr><td>2</td><td>3</td><td>4</td></tr></table>	4	3	4	2	4	3	2	3	4
1	1	1	0	0																																									
0	1	1	1	0																																									
0	0	1	1	1																																									
0	0	1	1	0																																									
0	1	1	0	0																																									
1	0	1																																											
0	1	0																																											
1	0	1																																											
4	3	4																																											
2	4	3																																											
2	3	4																																											

Figure 2.7: Example of Convolution Operation with Kernel Size 3 on a 5×5 Image.

As a result of the convolution layer, an output of activations is produced in the form of a two-dimensional array, called an activation map. A CNN architecture typically consists of:

- An input layer, responsible for inputting data into the CNN.
- A group of layers is responsible for extracting features from data and, in the case of images, identifying lines, edges, and contours. As more convolutional layers are composed, more complex information is extracted from the images. This group normally consists of a convolutional layer, which performs the convolutions, generating activation maps as output. Subsequently, due to the size of the activation map, a grouping layer is used to extract the main information from the activation map, reducing its dimensionality to reduce the number of parameters resulting from the extraction of features [5].
- A group of layers responsible for making the final classification of what was extracted by the previous group. In this case, dense layers are used, which are layers with fully connected neurons with the aim of connecting the outputs of the previous layers and ultimately generating the result.

2.2 Federated Learning

Federated Learning (FL) is a decentralized machine learning approach introduced by [38] using the processing power of distributed devices, called clients, with

access to local data and coordinated by a central server. The central server is responsible for managing the learning task federated, determining its beginning and end, and consolidating the final result based on what was learned by devices.

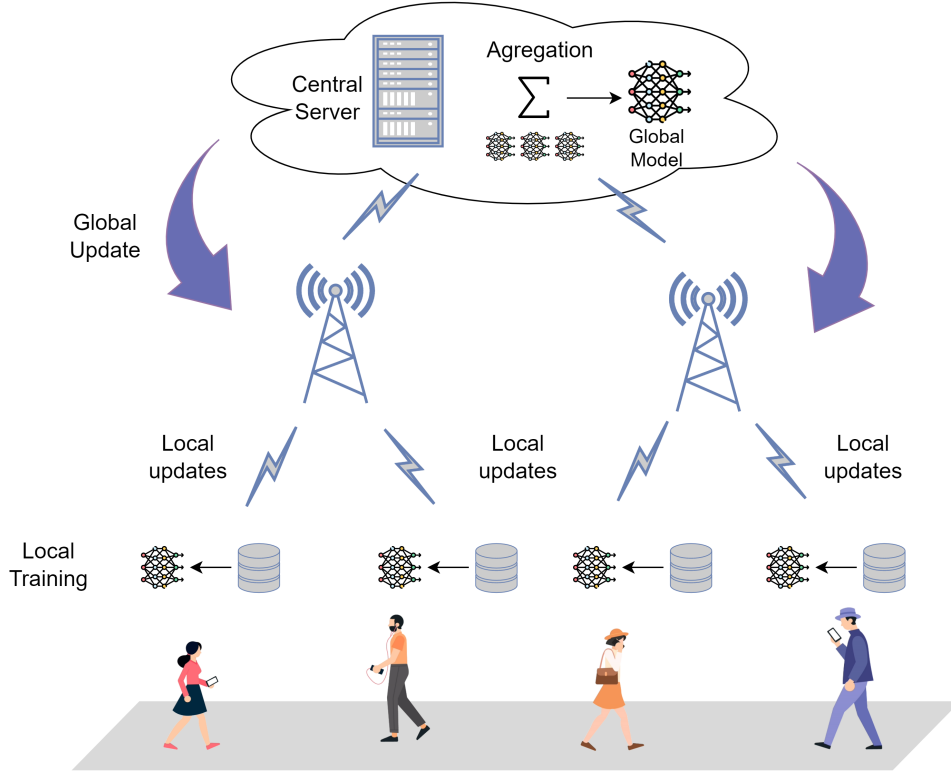


Figure 2.8: Federated Learning Training Process.

Given a machine learning task to be performed on a set of distributed data n , accessed by a set of K clients, n_k is the subset of data by a client k . Typically, performing machine learning tasks requires direct access to data, and in a distributed context, it would generate the need to transmit data from clients to the central server.

In the FL strategy, each client is responsible for performing a local machine-learning task only on its data, normally using ANN and generating the local model, which is composed of the parameters (weights and biases) ω^k resulting from this activity [31]. The central server, in turn, is responsible for generating the global model that results from the learning carried out by clients for each round of federated training through an aggregation operation. The global model in one given time $t + 1$ is the result of the aggregation, which is the weighted average of all local models (ω_{t+1}^k) defined as:

$$\omega_{t+1} = \sum_{k=1}^K \frac{n_k}{n} \omega_{t+1}^k. \quad (2.8)$$

As illustrated in Figure 2.8, federated training starts with the central server sending the global model, which consists of the current weights and biases for clients, initially with random values. The clients, in turn, perform local learning

tasks from this model on their local data, generating the local model. This step, called local update, consists of clients sending all local models to the server central. After the local update stage ends, the central server aggregates the weights and biases of the received models using (2.8). If the central server detects that the stopping criterion has been reached in this round, the training process ends; otherwise, another round begins with the central server sending the new aggregated global model for clients, carrying out what is called a global update.

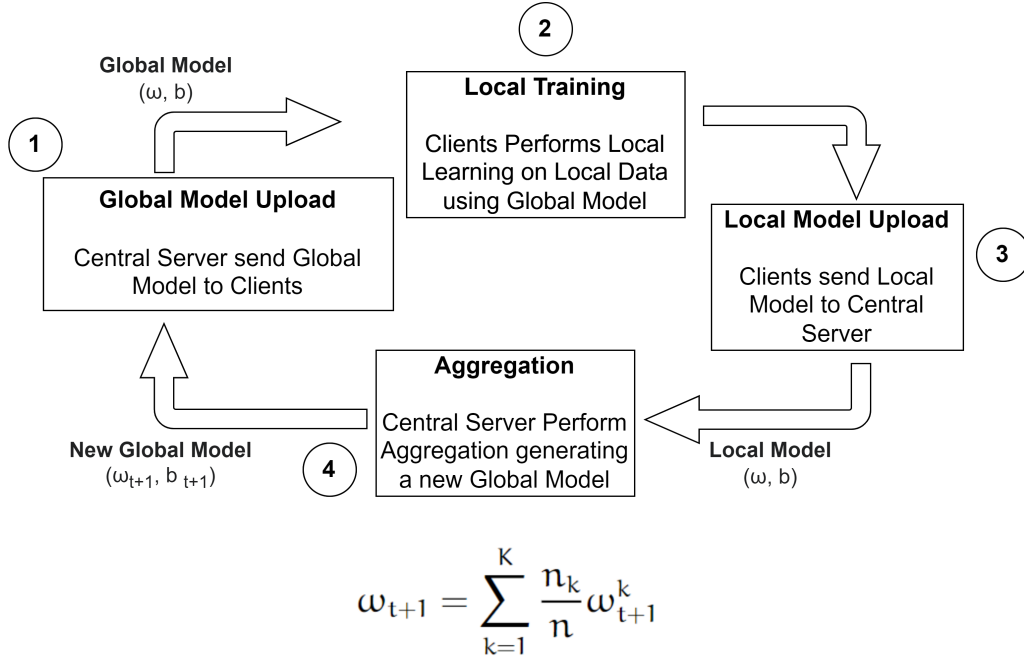


Figure 2.9: Federated Learning Round.

Figure 2.9 details a schematic of a round of FL. FL process at stage 1, with **Global Model Upload**, which consists of sending the global model, composed of weights and biases for clients participating in the learning. Then, in stage 2, clients perform what is called **Local Training**, where they perform a learning task using neural networks on their data from the global model. Upon completion of local training, clients complete stage 3 (**Local Model Upload**), which is the return of local models to the central server. After receiving all local models, at stage 4 (**Aggregation**), the central server performs local model aggregation using (2.8) and generates an update of the global model and uses this update to generate a new global model. A complete run from 1 to 4 stages is called a round of the FL. At the end of the round, if the FL objectives are achieved, FL ends; otherwise, the new global model is sent to a new set of data aimed at the continuity of training.

Using FL by edge devices allows access to data captured by sensors, enriching training with image data obtained in different locations and conditions. However, edge devices have peculiarities that must be considered when used as clients in FL, such as battery and processing power limitations, reducing the number of local iterations these clients can perform without compromising their functioning [3]. Furthermore, more complex neural network architectures tend to require more computational resources and more time to complete the local training task, increasing energy consumption.

2.3 Resource Costs Related to Federated Learning

Two types of resources are consumed in FL: processing costs and communication costs. The processing cost is related to the computational resources used on the central server and clients to reach a certain test accuracy. At the global level, the cost is related to the aggregation performed on the central server, and at the local level, it is related to the machine learning performed by the clients. In general, computational cost can be defined as the total number of iterations performed on clients to reach a certain level of test accuracy. Meanwhile, the communication cost is related to the number of global transmission rounds performed between the client devices and the central server [19]. The total value of the communication cost depends on the amount of data transmitted in each transmission and the number of federated training rounds [50].

However, the size of the transmitted weights must be taken into account and the communication overhead [32] can be calculated using

$$W = 2m(K \cdot \omega^*), \quad (2.9)$$

where m is the total number of FL rounds or synchronizations, multiplied by 2, representing the uplink phase and the downlink phase, K is the number of clients and ω^* is the size of the local model weights. Convergence of the FL algorithm heavily influences the amount of data transferred, with longer convergence times to achieve the desired test accuracy resulting in large values of m . In addition, the number of clients K and the complexity of the neural network used for local training also affect the amount of data transferred, as they determine the size of the trafficked weights ω^* .

According to [57] one of the ways to reduce the cost of communication in FL is to increase the number of local client iterations, with positive results in learning convergence and reducing the number of FL rounds to achieve a certain target accuracy and consequently the communication cost, but increasing the computational cost. However, from the perspective of using FL for mobile devices, the increase in local rounds represents an increase in energy consumption for each client, which may make using FL unfeasible for some types of devices.

2.4 Heterogeneous Distribution of Data and non-IID

In FL, the use of mobile devices as clients and their mobility characteristic allows access to a vast diversity of data, enriching learning, but also brings with it the characteristic of non-IID [38] data in local data distributions of clients and the challenges inherent to this type of distribution. There are several ways to classify the type of heterogeneity of client data in FL [63, 33], of which we will highlight the main groups:

- **Attribute Skew:** Denotes a data sampling with skewness based on feature heterogeneity. In this case, the features or characteristics can be distributed asymmetrically between the clients of FL.
- **Label Skew:** The data have a distribution in which your samplings have different classes among the clients.

- **Quantity Skew:** The amount of data across clients is unevenly distributed regardless of classes and features.
- **Temporal skew:** The data sample is unevenly distributed over time.

	Client 1		Client 2		Client 3	
						
	airplane	automobile	cat	dog	ship	truck
samples by classes	10	10	10	10	10	10

Figure 2.10: CIFAR10 example with label size imbalance of level 2, with each client having 20 images as data sample.

Considering these types of heterogeneities, in this study, we will treat the type of heterogeneity by label skew because, as highlighted by [63], this type of imbalance is caused by the variation in the location of clients who come into contact with data belonging to different class types. In the context of mobile devices, acting as clients in FL tasks and aiming to perform image classification, client sensors can access images belonging to different classes because they are in different locations. Label Skew can have two imbalance subtypes: **label size imbalance** and **label distribution imbalance**. **Label size imbalance** occurs when each client has a fixed amount of data, but this only belongs to several classes. Therefore, the lower the value of C , the higher the level of label imbalance, as seen in Figure 2.10, which exemplifies this type of occurrence using the CIFAR10 Dataset as an example.

						
	airplane	automobile	cat	dog	ship	truck
Set 1	1	1	1	28	1	28
Set 2	5	15	5	15	5	15
Set 3	10	10	10	10	10	10

Figure 2.11: CIFAR10 example illustrating different levels of label distribution imbalance.

Label distribution imbalance occurs when clients have a fixed amount of data, but the amount of data belonging to each class has a variable concentration

level. In other words, unlike **label size imbalance**, there is not necessarily a total absence of data representing a class, but rather a concentration of data in certain classes. A greater concentration in fewer classes indicates a more significant imbalance in the distribution. Figure 2.11 illustrates 3 data sets with different concentration levels and imbalance considering label distribution imbalance.

2.5 Impact of Weight Divergence on FL

According to [62], the divergence of the weights between FL and conventional learning (Stochastic Gradient Descent - SGD), impacts the accuracy results and convergence of Federated Training. In more homogeneous distributions, divergence is minor and has a low impact; conversely, more heterogeneous distributions can heavily reduce the accuracy of federated training. Taking into account the weights $\omega_t^{(c)}$ and $\omega_t^{(f)}$ as the weights resulting respectively from the SGD training and the federated training, in a heterogeneous scenario, at each step of the FL training, its divergence increases because the distance between the weights of the clients k increases cumulatively. Additionally, the divergence increases as the data becomes more heterogeneous.

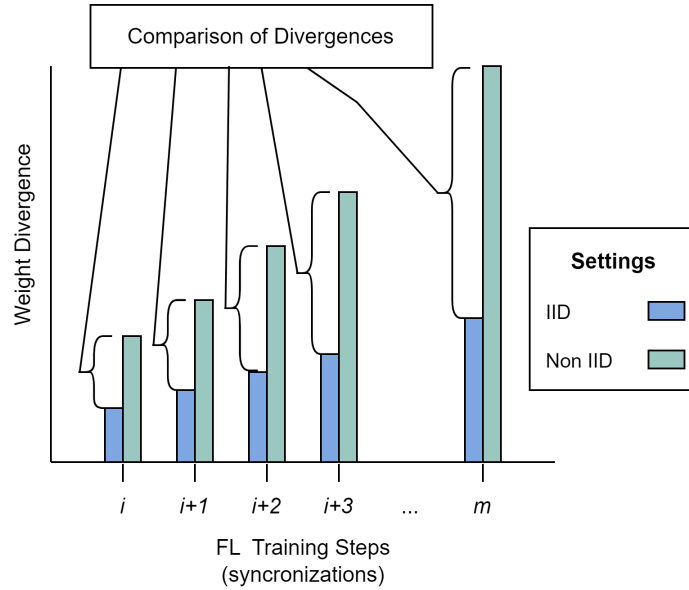


Figure 2.12: Comparison of the weight divergence behavior between IID Settings and Non IID Settings on FL training processes.

As illustrated in Figure 2.12, in an FL training with steps varying from i to m , datasets with homogeneous distribution (IID), represented in blue, have slight variation in weight divergence over time in FL training. In turn, datasets with more heterogeneous distribution diverge from their weights over time. Taking this behavior of IID distributions into account, Zhao et al. (2018) [62] defined Proposition 3.1 for weight divergence.

Consider a two-sample scenario of a classification problem with C classes and weights $\omega^{(c)}$ and $\omega^{(f)}$ generated by centralized and federated training using FedAvg with synchronizations performed every T local steps, making a total of m synchronizations, where the gradient ($g_{\max}$) is the max value of gradients

$(\nabla_{\omega} E_{x|y=1} \log f_i(x, \omega))$ and is Lipschitz $(\lambda_{x|y=i})$ for each class C , and losses are optimized. Data are organized into $n^{(k)}$ samples for K total clients with a probabilistic distribution $p^{(k)}$ between them.

$$\begin{aligned} & \|\omega_{mT}^{(f)} - \omega_{mT}^{(c)}\| \leq \\ & \sum_{k=1}^K \frac{n^{(k)}}{\sum_{k=1}^K n^{(k)}} (a^{(k)})^T \|\omega_{(m-1)T}^{(f)} - \omega_{(m-1)T}^{(c)}\| \\ & + \eta \sum_{k=1}^K \frac{n^{(k)}}{\sum_{k=1}^K n^{(k)}} \sum_{i=1}^C \|p^{(k)}(y=i) - p(y=i)\| \\ & \sum_{j=1}^{T-1} (a^{(k)})^j g_{\max}(\omega_{mT-1-k}^{(c)}) \end{aligned} \quad (2.10)$$

where, $g_{\max} = \max_{i=1}^C \|\nabla_{\omega} E_{x|y=1} \log f_i(x, \omega)\|$

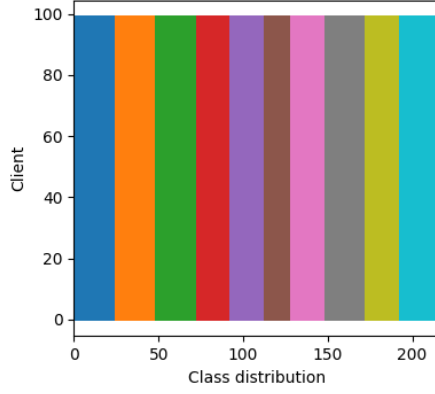
and $a^{(k)} = 1 + \eta \sum_{i=1}^C p^{(k)}(y=i) \lambda_{x|y=i}$

Table 2.1: Notation Symbols on (2.10)

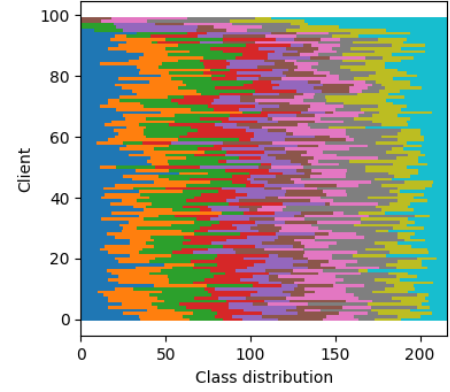
c	centralized training
C	number of classes on a classification problem
f	federated training
K	total number of clients on FL
m	number of synchronization (rounds) of FL
T	number of local epochs of training
$n^{(k)}$	data sample on client k
$\omega^{(c)}$	weights on centralized training
$\omega^{(f)}$	weights on FL
η	learning rate
$\nabla_{\omega} E_{x y=1} \log f_i(x, \omega)$	value of gradients
$g_{\max}$	max value of gradients
$p^{(k)}$	probabilistic distribution between client data
$\lambda_{x y=i}$	Lipschitz constant of gradients.

The part of (2.10) composed of $\sum_{i=1}^C \|p^{(k)}(y=i) - p(y=i)\|$ is called Earth Mover's Distance (EMD) [46] of data distributions between clients and the whole set of data and is the root cause of weight divergence on inequality and can be used to compare the impact of varying levels of data heterogeneity on the test accuracy of federated training. This metric is calculated as the ratio between the distributions p of each client k and the overall distribution when the distance measurement is defined as $\|p^{(k)}(y=i) - p(y=i)\|$. Larger EMD values indicate more heterogeneous data distributions, which have a more significant impact on the accuracy of FL, and lower EMD values have a less significant impact. For example, a sample with an EMD with a value close to 0 means that the data distribution has little impact and weight divergence between FL and centralized learning will be very small. In terms of communication cost in FL, increased divergence and EMD measures can result in more FL training rounds (m) to achieve

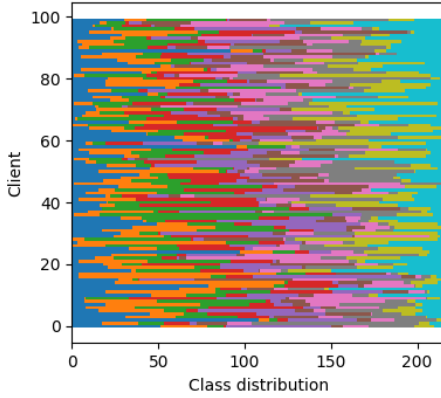
the desired accuracy, directly affecting the communication cost (W). Therefore, a dataset with a distribution with higher EMD will increase communication costs and may make FL operations unfeasible.



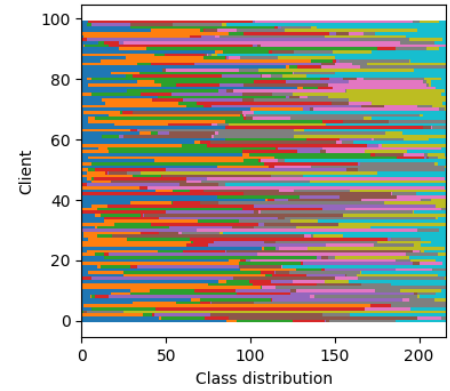
(a) EMD 0: Clients have the same amount of data per class, denoting a homogeneous distribution.



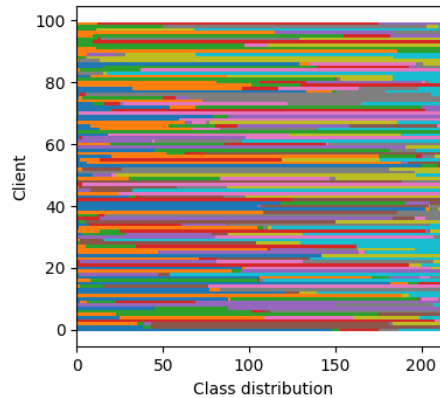
(b) EMD 0.37: Clients with data belonging to 10 classes, with imbalance only in the amount of data per class.



(c) EMD 0.71: Clients with data ranging between 5 and 10 classes, presenting class imbalance.



(d) EMD 1.08: Clients with data ranging between 4 and 10 classes, presenting class imbalance.



(e) EMD 1.41: Clients with more drastic class imbalance, with data belonging to 1 or 2 classes.

Figure 2.13: Distribution of Data by Classes in 100 Clients Considering EMD Values 0, 0.37, 0.71, 1.08 and 1.41.

Figure 2.13 exemplifies three samples with a data distribution with different EMD values: 0, 0.37, 0.71, 1.08, and 1.41, considering a scenario with 100 clients with access to local data. When observing the distribution, we notice that the lowest EMD values (0.37) present an imbalance that does not necessarily indicate the absence of a class in a given client, but rather the reduction of data that represents this class. Intermediate values (0.71 to 1.08) already show the absence of some classes, while level 1.41 shows a level of data concentration in 1 to 2 classes in the clients involved, denoting a more acute unbalanced scenario.

2.6 Related Work

FL was presented as a decentralized approach by [38] with significant advances in the issue of data privacy in training and taking advantage of decentralized computing resources, also called FedAvg. This study showed that the results obtained presented significant results even in situations of heterogeneous data distribution.

However, subsequent studies [62] point out that the federated process in situations of heterogeneity suffers from degradation in accuracy, mainly attributed to divergence of weights of local model layers caused by the heterogeneous data distribution among FL clients.

According to [63], several techniques are used to treat heterogeneous data in FL, and among them, one group stands out for accelerating the training and convergence process of deep learning in FL. This type of technique is classified as structure adaptation as it aims to use structural adaptations such as optimizers and normalization. Optimizers such as Adagrad [10] and Adam [20] are classified as adaptive, as they have the intelligence to vary the learning rate and other settings, adapting to the data context and architecture used. However, using those optimizers in FL instead of SGD requires maintaining extra information about the results of previous rounds for its correct operation, which can double communication costs in FL [63].

Several studies have attempted to address the issue of heterogeneous data in FL by using local normalization techniques. One such technique is FedBN [30], which proposes the use of BN without incorporating the specific parameters of this aggregation, with relevant results compared to the original FedAvg technique without using any normalization. The cause of heterogeneity problems was attributed to internal covariance, and the improvement achieved in the study was attributed to the reduction of covariance that BN allows in centralized scenarios. Recently, however, this type of interpretation has changed, and the results achieved by BN are a result of smoothing of training behaviors [48].

Fed2 [60] suggests using a resource allocation technique that employs GN as a normalization layer. The cause of the impact of heterogeneity was attributed to the divergence of weights and, more specifically, to permutation invariance, which would be caused by the sequence of permutations in the convolutional layers. This method proposes to align and group the characteristics using GN to optimize the convergence of results. Tests conducted with heterogeneous data considered only the presence or absence of classes in the clients, and four distributions were examined without performing a more detailed analysis of the results of these distributions. Subsequently, [61] proposed the replacement of

BN by GN in semi-supervised training, and its results showed that the use of BN achieved lower results than GN, which managed to improve the behavior of gradients and, consequently, training accuracy.

FedNorm [9] proposes the use of Layer Normalization (LN) [4] on the client's local training process in CNN architecture to reduce the impact of heterogeneous distributions on FL, presenting LN as the technique with better results in accuracy and convergence. However, some results showed that the GN technique had an accuracy very close to LN, and its convergence was not compared with LN by the study, lacking a stronger justification for not using GN. The cause of the impact of the heterogeneous distribution was also attributed to covariance shift and, as in FedBN, this type of interpretation on BN results has already been reviewed [48].

None of these studies went into depth using a heterogeneity metric that correlates the divergence of weights with the heterogeneous distribution of data present on clients in FL. Furthermore, the studies did not consider the costs of communication in FL, which is a significant factor considering its use on mobile devices. Our proposal differs from previous works in that it seeks to advance FL training, considering communication costs and a deeper understanding of the impact of heterogeneous distributions on FL.

3. Proposed Solution

3.1 Proposed Method

As discussed in Section 2.5, heterogeneous data distributions increase the divergence of weights with each round of training in FL. This affects both the local client training and the subsequent aggregation. In the context of FL, there are different ways to approach the problems caused by heterogeneous data [63]. Analyzing (2.10), we can discuss how your components can impact the weight divergence problem. Three principal components contribute to it: the learning rate η , the gradient $g_{\max}$, the EMD of distributions, and the Lipschitz constant of gradients ($\lambda_{x|y=i}$). Reducing η is not a good option, considering that it can cause a reduction in FL convergence and consequently an increase in communication costs. From these considerations, we can make the following remarks inspired by this and (2.10).

Remark 1. In the first round of FL, there is no divergence from previous rounds of training to impact the current round. In other words, at this moment, the impact of divergence comes entirely from EMD, gradients, and $a^{(k)}$. In other words, the divergence in the first round would be:

$$\begin{aligned} \|\omega_0^{(f)} - \omega_0^{(c)}\| = & \eta \sum_{k=1}^K \frac{n^{(k)}}{\sum_{k=1}^K n^{(k)}} \sum_{i=1}^C \|p^{(k)}(y=i) - p(y=i)\| \\ & \sum_{j=1}^{T-1} (a^{(k)})^j g_{\max}(\omega_{mT-1-k}^{(c)}) \end{aligned} \quad (3.1)$$

Remark 2. Any technique that produces smooth values of $g_{\max}$ and $a^{(k)}$ will have the effect of reducing the impact of the EMD for the divergence in the current round, as these two factors make a product with the value of EMD in (2.10). In this situation, the most significant portion of the current divergence will come from the divergence generated in the previous round.

Considering the above discussion, we conclude that any technique that can smooth the gradient and minimize Lipschitz constant will also reduce the impact of weight divergence. As seen before in Section 2.1.2, the normalization techniques minimize the values of the constant, considering a centralized scenario. Recently, Weight Standardization (WS) [44] achieved relevant results by smoothing the losses and gradient values, reducing their Lipschitz constants, allowing better results than other normalization techniques (BN, LN, and GN).

Based on the information presented earlier, we propose a new technique called Federated Learning Algorithm with Weight Standardization on Convolutional Layers (FedWS) to use the standardization technique in an FL context aiming to improve local training results under conditions of heterogeneous data distributions.

Local training in FL is represented as $\omega \leftarrow \omega - \eta \nabla l(w; b)$, where ω are the local weights, η is the learning rate, ∇ is the gradient descent, and $l(w; b)$ is the loss of the local training step. The gradient guides the training by indicating the direction and magnitude of the correction needed to minimize loss and maximize accuracy. In heterogeneous data distributions in FL, the gradient can direct local training toward weight divergence, affecting weight behavior in later stages of training.

Using standardization on local training will improve gradient decisions in situations of the saturated regime and smooth the gradient variation, allowing the optimizer to make less drastic decisions based on the gradient, resulting in smoother convergence during training. The standardization works by normalizing the output weights of the convolutional layer using the mean (μ) and standard deviation (σ) of the weight set, as shown on:

$$\omega^{ws} = \frac{\omega - \mu^\omega}{\sigma^\omega}. \quad (3.2)$$

FedWS aims to apply this standardization on the convolutional layers to smooth out the gradient values during local training performed by clients, allowing the aggregation performed using FedAvg strategy to reflect this characteristic in the global model of FL and reducing weight jumps that can exacerbate weight divergence, as described in Section 2.5.

When clients perform what is called **Local Training**, they perform a learning task using neural networks on their data from the global model. At this stage, FedWS normalizes the output of the convolutional layers, more precisely the activation map, using (3.2) and generating a new normalized activation map. This activation map is used in subsequent neural network layers to produce lower loss values as illustrated in Figure 3.1.

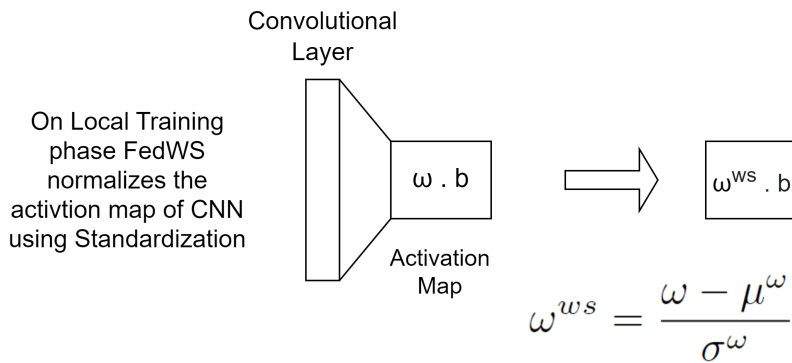


Figure 3.1: FedWS Normalization on Local Learning.

As depicted in Figure 3.2 the FL is composed of two principal functions called **ServerTraining**, which are responsible for controlling the server training and performing a major iteration from 1 to m , representing the rounds of FL, and

the Global Model Upload and Aggregation phases and **ClientUpdate** that are responsible for the execution of Local Learning, in local iterations i from 1 to E and the Model Update phases.

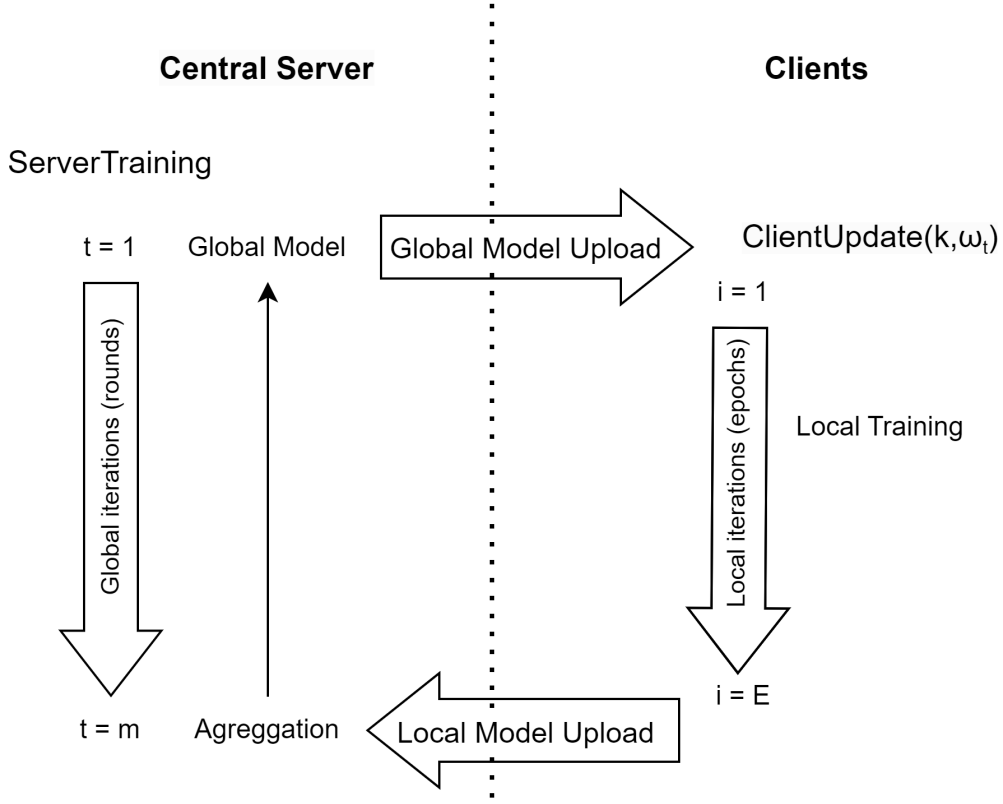


Figure 3.2: Detailed global and local iterations to perform FL.

As detailed in Algorithm 1, FL training is performed by the function "Server Training" defined by the FedAvg approach. It starts with parameter initialization ω_0 and proceeds with federated processing in t rounds ranging from 1 to m . In each round, a set of clients S_t is selected, and for each member k , local FL training is performed through the **ClientUpdate** function that sends the current parameters ω_t for each client performing the **Global Model Upload** as depicted on Figure 2.9 expecting to receive in return the local update ω_{t+1} .

Each client k performs then a **Local Training** using the ω_t received inside function **ClientUpdate**. That function starts splitting the local dataset n_k in batches with size B on line 12. Next, on line 13 local training is performed from 1 to E local epochs. For each epoch, deep learning is performed on a batch of data (b) using a neural network.

The proposed FedWS technique acts on the local training that occurs on line 15, in the **ClientUpdate** function, where the operation shown in (3.2) is applied directly to the output weights of the convolutional layers. Therefore, the average (μ^ω) and the standard deviation (σ^ω) used are not parameters of the CNN and do not suffer from the same anomalies as BN when the parameters and weights are aggregated at the global model level. Line 18 represents the final weights resulting from deep learning operation for a client. After the end of all local epochs, the client ends local learning on line 21 returning the parameters to the server, performing the **Local Model Upload**.

Returning to ServerTraining, at the end of each round, after receiving all client updates ω_{t+1} , the server performs the **Aggregation** as defined in line 8 generating a *New Global Model*. If t reaches m the FL process ends otherwise a new round starts.

Algorithm 1 FedWS: Federated Learning Weight Standardzation

```

1: function SERVERTRAINING :
2:   initialize  $\omega_0$ 
3:   for  $t = 1$  to  $m$  do
4:      $S_t \leftarrow \text{clientsrandomset}$ 
5:     for  $k \in S_t$  do
6:        $\omega_{t+1}^k \leftarrow \text{ClientUpdate}(k, \omega_t)$  ▷ Run in parallel
7:     end for
8:      $\omega_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} \omega_{t+1}^k$ 
9:   end for
10: end function
11: function CLIENTUPDATE( $k, \omega$ ) : ▷ Run on client  $k$ 
12:    $\beta \leftarrow n_k$  ▷ split  $n_k$  local dataset in batches with size  $B$ 
13:   for  $i = 1$  to  $E$  do ▷ local epochs (i) ranging from 1 to  $E$ 
14:     for  $b \in \beta$  do
15:       if current layer is convolutional then
16:          $\omega \leftarrow \frac{\omega - \mu^\omega}{\sigma^\omega}$  ▷ apply WS on convolutional layers output
17:       end if
18:        $\omega \leftarrow \omega - \eta \nabla l(\omega; b)$ 
19:     end for
20:   end for
21:   return  $\omega$  to server
22: end function

```

Considering Remark 1 and 2 we can develop what the main effects expected from the application of FedWS will be. In Remark 1, as we have no round before the initial round, the divergence between weights is mainly influenced by EMD and gradient ($g_{\max}$). In Remark 2, in subsequent rounds the biggest impact, in the case where the gradients have very small values, comes from the divergence obtained in the previous round.

The objective of FedWS is to smooth the values of the gradients so that in the first round of FL, its value multiplied by EMD, as seen in (3.1), reduces the impact of EMD in the first round, and the weight divergence generated. Therefore, higher EMD values should produce little effect in the first round of training, generating a smaller divergence for the following rounds. Furthermore, as seen in Remark 2 in subsequent rounds, smaller gradients will reduce the influence of EMD so that the greatest impact comes from the previous round.

Table 3.1 exemplifies the expected effect of FedWS on divergence by reducing gradients in FL. Considering round 0 as the first round, the gradients directly influence the divergence generated $\|\omega_0^{(f)} - \omega_0^{(c)}\|$ that comes from the product of $\text{EMD} \times g_{\max}$. In the next round, the divergence generated will come from $\|\omega_0^{(f)} - \omega_0^{(c)}\| + (\text{EMD} \times g_{\max})$ and in this case it will be influenced both by the reduction of the previous divergence and by the reduction of the values of $g_{\max}$.

This effect becomes cumulative until the final round m which concludes the FL training.

Table 3.1: Expected influence of FedWS on weight divergence during FL rounds, considering the impact on η , EMD and $g_{\max}$ components, considering fixed values of η and EMD.

FL Rounds	Weight Divergence Main Components	Generated Weight Divergence
0	$\eta \times \text{EMD} \times \Downarrow g_{\max}$	$\Downarrow \ \omega_0^{(f)} - \omega_0^{(c)}\ $
1	$\Downarrow \ \omega_0^{(f)} - \omega_0^{(c)}\ + (\eta \times \text{EMD} \times \Downarrow g_{\max})$	$\Downarrow \ \omega_1^{(f)} - \omega_1^{(c)}\ $
2	$\Downarrow \ \omega_1^{(f)} - \omega_1^{(c)}\ + (\eta \times \text{EMD} \times \Downarrow g_{\max})$	$\Downarrow \ \omega_2^{(f)} - \omega_2^{(c)}\ $
m	$\Downarrow \ \omega_{m-1}^{(f)} - \omega_{m-1}^{(c)}\ + (\eta \times \text{EMD} \times \Downarrow g_{\max})$	$\Downarrow \ \omega_m^{(f)} - \omega_m^{(c)}\ $

In this way, FedWS will produce effects in local training resulting from the use of standardization, but it will also produce effects in federated training, reducing the impact of EMD and divergences in the first and subsequent rounds, and mitigating the effect that heterogeneous distributions produce in FL.

In the next chapter, research questions and experiments will be developed to validate the proposal according to its objective and expected effects.

4. Experiments

Based on the proposed solution, it is important to understand its effectiveness and behavior through experiments. Considering that the proposal aims to act on the convolutional layers of a CNN, it must be verified how heterogeneous distributions affect this type of layer and how the solution behaves in this situation. Furthermore, it is important to identify how the proposed solution behaves in relation to weight divergence, losses, and gradients since its objective is to use a technique to reduce the impact of heterogeneous distributions on these components. Another point to be evaluated is how much FedWS can compare to the results of centralized training aimed at potential use. Finally, as the use of normalization techniques increases the convergence of centralized training, we must measure how much the increase in local rounds produces benefits in situations of heterogeneous distributions when using FEdWS.

Considering this information, complementary research questions are presented below to develop what has been exposed.

4.1 Research Questions

In addition to the research questions RQ1 and RQ2 defined in Section 1.2, it is important to investigate how the proposed solution manages to improve the results. Therefore, new research questions are defined below:

- **RQ3 - How heterogeneous data distribution affect the layer responsible for extracting the features of a CNN and how do the normalization techniques perform in this case ?**

This research question aims to show how the convolutional layers of a CNN are affected by the heterogeneous distribution and how normalization techniques can improve this scenario.

- **RQ4 - Does the proposed method manage to reduce the impact of the divergence of the weights of the heterogeneous distributions?**

This research question aims to assess whether the proposed method reduces the impact of weight divergence, identified in Section 2.5 as the most significant problem caused by heterogeneous data distribution.

- **RQ5 - Does the proposed method manage to improve the behavior of losses and gradients?**

The objective of this research question is to evaluate the behavior of losses and gradients of the observed techniques, considering the influence that these factors have on the understanding of the results of RQ1.

- **RQ6 - How does the proposed method compare against the centralized scenario?**

Considering a possible improvement in results with heterogeneous data, this research question seeks to understand the performance of the proposed method compared to a centralized machine learning scenario.

- **RQ7 - How the local training epochs can impact the results of FL on heterogeneous data distributions ?**

This research question aims to identify whether the increase in local epochs of local training carried out by clients can positively affect FL results in situations of heterogeneity and how this increase is reflected in communication and computational costs.

Based on the research questions defined in Section 4.1, in this chapter we present the experiments to evaluate the performance of our proposal and techniques of the state of the art techniques. A summary of these experiments is described in Table 4.1.

Table 4.1: Summary of the definition of experiments

Experiment	Research question	Datasets	Observed techniques	Metrics
I (in Section 4.3)	RQ1, RQ2	MINST, CIFAR10 and EuroSAT	FedBN(BN), Fed2(GN), FedNorm (LN) and FedWS	Test Accuracy
II (in Section 4.4)	RQ1, RQ2	MINST, CIFAR10 and EuroSAT	FedBN(BN), Fed2(GN), FedNorm (LN) and FedWS	Communcation costs
III (in Section 4.5)	RQ3	MINST, CIFAR10 and EuroSAT	FedBN(BN), Fed2(GN), FedNorm (LN) and FedWS	-
IV (in Section 4.6)	RQ4	MINST, CIFAR10 and EuroSAT	FedBN(BN), Fed2(GN), FedNorm (LN) and FedWS	Weight Divergence
V (in Section 4.7)	RQ5	MINST, CIFAR10 and EuroSAT	FedBN(BN), Fed2(GN), FedNorm (LN) and FedWS	Losses and gradients
VI (in Section 4.8)	RQ6	MINST, CIFAR10 and EuroSAT	Centralized Training and FedWS	Test accuracy and gradients
VII (in Section 4.9)	RQ7	EuroSAT	FedBN(BN), Fed2(GN), FedNorm (LN) and FedWS	Test Accuracy, Communca-tion and Computa-tional costs

4.2 Setup and Datasets

The machine used for federated training was a notebook with a Core i5 processor, 16GB RAM, and Nvidia GPU graphics card with 4GB memory and CUDA support. The experiments were carried out using a framework¹, which was changed to allow the experiments, using the Python 3.8 and Pytorch 1.19 environment.

According to [33], the dataset most used in studies on heterogeneous data on FL is MNIST [26] and CIFAR10 [21]. So we used these two datasets, and the EuroSAT [6] composed of 27,000 images captured via satellite, with 64×64 pixels, and divided into ten classes representing different types of land (permanent and temporary plantations, roads, rivers, forests, pastures, lakes, diverse vegetation, industrial and residential areas) as detailed on Table 4.2.

Table 4.2: Datasets used on experiments.

Datasets	Number of Images	Image Dimension	Number of Classes	Classes
EuroSAT	27,000	64×64 pixels	10	permanent and temporary plantations, roads, rivers, forests, pastures, lakes, diverse vegetation, industrial and residential areas
CIFAR	60,000	32×32 pixels	10	airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks
MNIST	60,000	28×28 pixels	10	0,1,2,3,4,5,6,7,8,9

All data sets were set in a fixed proportion of 80% for training and 20% for tests. Most studies on the subject use the strategy below [33] to represent data heterogeneity in experiment datasets:

- **Class1 non-IID:** Studies that process data so that clients have a divergent Class 1 distribution.
- **Class2 non-IID:** Studies that process the data so that customers have a divergent distribution of 2 classes.
- **IID:** Homogeneous distribution of data among clients.

In our experiments, we chose a strategy that allows for a comparison with more degrees of heterogeneity to understand how the techniques behave in this

¹<https://github.com/c-gabri/Federated-Learning-PyTorch>

scenario. We generated sample distribution for local client data with five arrangements with EMD values of 0, 0.37, 0.71, 1.08, and 1.41, to represent sets with an increasing class imbalance on client local data, with 0 meaning a homogeneous distribution, 0.37 a slight imbalance on data distribution, 0.71 and 1.08 intermediate imbalance and 1.41 the distribution with the highest degree of heterogeneity. Three data samples were generated for each EMD value, allowing each experiment to be executed three times and making it possible to use the mean and standard deviation for the metrics.

The federated training was set up with 200 rounds ($m = 200$), a learning rate (η) of 0.01, a local training batch size of 50 ($B = 50$), five local training epochs, and ten clients ($K = 10$). Two architectures of CNNs will be used in the experiments, as described in Table 4.3.

Table 4.3: CNN Model Architectures

Architecture	Layers	No. of parameters
LeNet5 [38]	7	1,065,354
ResNet18 [17]	18	11,689,512

The scenarios considered included "No Normalization" (NN), where no normalization technique was applied to the model, and the other represent the techniques used by related works: FedBN(BN) [30], Fed2(GN) [60], and FedNorm (LN) [9], and finally the scenario using FedWS.

To perform FL using heterogeneous distributions, the Datasets were prepared by the framework that read the data, generated subsets considering a division by 10 clients, and organized them to reach the heterogeneity defined by the chosen EMD. In the end, these subsets were assigned to clients to be used in local FL training. It is important to emphasize that this process only occurs to prepare the experiments. In real situations, clients would already have access to local data and FL does not transmit data between federation entities.

The training strategy was defined as follows:

1. Perform FL according to hyperparameters and the defined CNN architecture for client local training.
2. Write the results to disk for further analysis.
3. Stop when you reach the desired number of rounds (m).

4.3 Experiment I: Classifying Images in Heterogeneous Data

This experiment aims to investigate the Research Questions RQ1 and RQ2, regarding the comparison of techniques observed in a FL task for image classification. In this experiment, the impact of the degree of heterogeneity in the techniques will also be evaluated.

Here, the results of image classification using the MNIST, CIFAR10, and EuroSAT datasets will be discussed to compare the behavior of the techniques at different levels of heterogeneity. First, in Table 4.4 the results using the LeNet5 architecture are presented and the best results are highlighted in bold.

Table 4.4: Federated Test Accuracy(%) Results on Different Datasets, Techniques and Levels of EMD

EMD	0	0.37	0.71	1.08	1.41
NN	98.8 (0.1)	98.8 (0.0)	98.6 (0.1)	98.2 (0.3)	97.8 (0.2)
BN	99.0 (0.1)	99.1 (0.0)	99.0 (0.1)	98.9 (0.1)	98.7 (0.1)
GN	99.1 (0.0)	99.1 (0.1)	99.0 (0.1)	98.8 (0.2)	98.6 (0.2)
LN	99.0 (0.0)	99.1 (0.1)	99.0 (0.1)	98.8 (0.2)	98.6 (0.2)
FedWS	99.1 (0.1)	99.1 (0.1)	99.1 (0.1)	98.9 (0.1)	98.7 (0.1)

Standard deviation in parentheses

(a) Results in MNIST

EMD	0	0.37	0.71	1.08	1.41
NN	71.6 (0.2)	71.1 (0.4)	68.6 (0.4)	62.4 (1.2)	53.2 (2.0)
BN	76.9 (0.2)	76.5 (0.3)	74.6 (0.5)	68.8 (1.2)	61.8 (1.2)
GN	75.5 (0.1)	75.3 (0.2)	72.7 (0.4)	66.9 (0.6)	59.4 (1.5)
LN	75.4 (0.2)	75.2 (0.4)	72.7 (0.6)	66.7 (0.8)	60.0 (0.6)
FedWS	79.7 (0.2)	79.1 (0.4)	77.5 (0.6)	72.5 (0.4)	66.0 (1.3)

Standard deviation in parentheses

(b) Results in CIFAR10

EMD	0	0.37	0.71	1.08	1.41
NN	77.1 (0.7)	76.2 (0.5)	74.8 (1.7)	71.9 (0.1)	64.8 (0.7)
BN	84.7 (0.3)	82.4 (0.1)	82.1 (1.1)	77.8 (0.3)	72.5 (3.8)
GN	82.0 (0.9)	80.9 (0.3)	80.1 (1.4)	75.0 (0.6)	71.3 (1.2)
LN	81.0 (0.1)	80.9 (0.2)	79.7 (1.9)	74.4 (0.2)	70.4 (0.5)
FedWS	84.4 (0.6)	84.5 (0.6)	84.3 (0.8)	80.3 (0.3)	77.0 (0.2)

Standard deviation in parentheses

(c) Results in EuroSAT

In the experiment with the MNIST dataset, all normalization techniques observed obtained a very similar result, confirming that this dataset is less affected by this combination of architecture and corroborating the result observed by [62]. In the results with the CIFAR10 dataset, FedWS presented better results than the other techniques in all levels of EMD, with test accuracy superior to the second place ranging from 2.41 to 4.15%. In the case of the EuroSAT dataset, on EMD 0 FedWS and BN have similar results; however, as the data becomes heterogeneous, FedWS performs better with superiority ranging from 2.0 to 5.0%.

One point deserves to be highlighted in the results: the accuracy advantage between FedWS and the others grows according to the increase in the EMD. This indicates that the proposed technique has a more linear and better behavior than

the others in larger EMDs. The difference between the accuracy measured at the highest EMD value (1.41) and the lowest (0) indicates the impact of the EMD on the technique compared to the homogeneous scenario (EMD = 0). In this scenario, FedWS also stands out, with values of 13.8% and 8.6% for CIFAR10 and EuroSAT, while the other techniques have results ranging from 15.1 to 18.4% (CIFAR10) and 11.6 to 14.8% (EuroSAT).

When observing the results of Table 4.4 clearly, the technique closest to the FedWS results is BN. Considering these results, another analysis point is the behavior of FedWS in a CNN architecture that initially used BN. In this case, the Table shows the experiment result considering three levels of EMD (0, 0.71, and 1.41) between the original ResNet18 and the one modified for FedWS.

Table 4.5: ResNet18 Image Classification Test Accuracy(%) Results, using 200 rounds of FL, 10 clients, and 5 local epochs

EMD	0	0.71	1.41
ResNet18	92.79	90.37	77.34
ResNet18 + FedWS	94.15	91.94	81.82

We can see in Table 4.5 that FedWS again demonstrates that in addition to having the best result in the three EMD levels, it has the most linear behavior with better results even at higher levels of heterogeneity and higher accuracies ranging from 1.36, 1.57 and 4.48 compared to ResNet18.

Another aspect to be considered is the convergence of techniques observed in the learning process. Figure 4.1 shows the convergence of techniques using the LeNet5 architecture, with datasets in distributions with EMDs 0.71 and 1.41. In the case of MNIST, the result with EMD 0.71 in Table 4.1a shows that FedWS reached an accuracy of 99% in round 160, while the other techniques reached this close value in round 200.

Considering EMD 1.41, FedWS reached 98% accuracy in less than 80 rounds, compared to the rest that reached close to 80 rounds. For the CIFAR10 dataset, the results with EMD 0.71 and 1.41 indicate that FedWS reaches accuracy percentages of 70% and 65%, respectively, in 90 and 160 rounds. In comparison, the other techniques only reached these levels in 120 rounds for 70% EMD 0.71 and failed to reach 65% in EMD 1.41. In the EuroSAT Dataset with two samples with EMD values of 0.71 and 1.41, FedWS achieved a test accuracy of 75% in 50 training rounds, in the first case, while the other techniques only reached this level after 100 rounds. With EMD 1.41, FedWS achieved values above 60% in 40 rounds, while the other techniques reached this accuracy after 80 rounds.

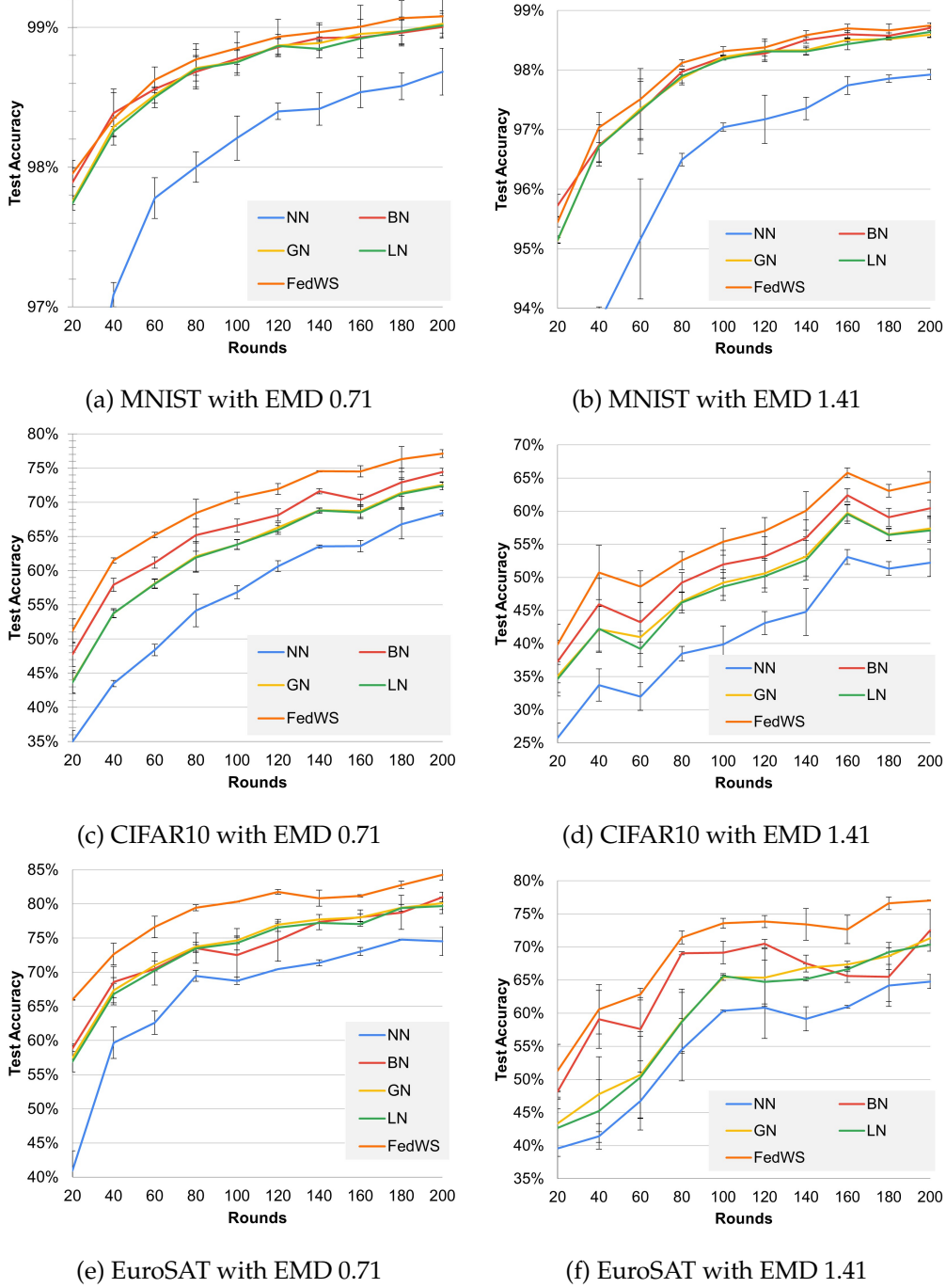
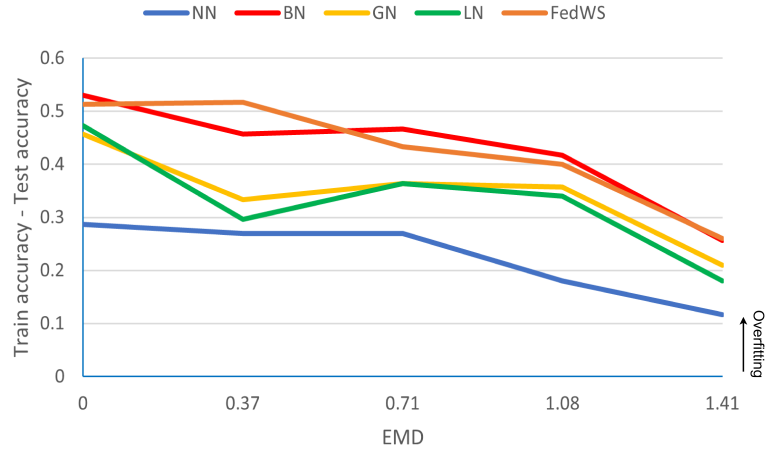
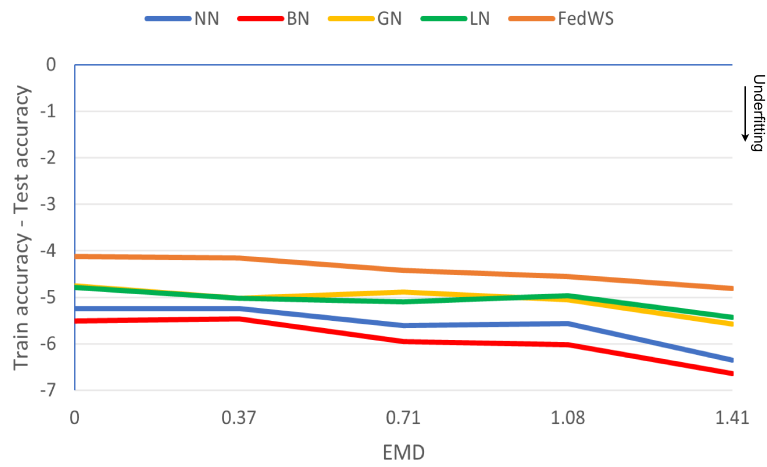


Figure 4.1: Convergence results of test accuracy in image classification task on MNIST, CIFAR10 and EuroSAT.

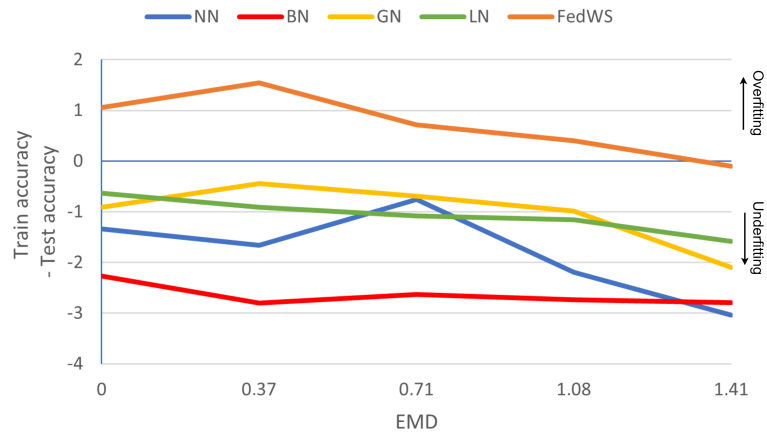
Another point to be analyzed is the comparison between training and testing results, which can indicate overfitting tendencies, when the neural network obtained a training result superior to the testing result, indicating that the generated model may not present a generalization aimed at using in real scenarios. On the contrary, underfitting occurs when the testing accuracy is higher than the training accuracy, it indicates that the neural network may not be learning enough from the training data.



(a) MNIST



(b) CIFAR10



(c) EuroSAT

Figure 4.2: Percentage difference between training and testing results of observed normalization techniques.

Figure 4.2 shows that the observed techniques generally presented an underfitting situation with a tendency to increase as the heterogeneity of the data samples increases. In MNIST, all techniques have a small overfitting behavior,

which reduces at higher levels of heterogeneity. However, we cannot credit this result to any improvement as the increase in heterogeneity provides a greater impact on the convolutional layer that extracts the features, harming this phase and impacting the training and testing accuracy. In CIFAR10, all techniques presented underfitting with values between -4% to 6.7%, with emphasis on FedWS which presented the best values. In EuroSAT, the difference values varied between -3% to 1.5%, indicating that in general, the difference was not so large as to cause any type of problem. But it is worth highlighting the behavior of some of the techniques: BN showed a more linear behavior, however with a higher underfitting value compared to the other techniques, LN showed a linear underfitting behavior that grew as the EMD value increased and FedWS showed overfitting in the samples less heterogeneous and becoming almost imperceptible underfitting in the sample with EMD 1.41.

4.4 Experiment II: Federated Learning Communication Cost

The objective of this experiment is to investigate Research Questions RQ1 and RQ2. In this experiment, the communication costs in FL will be compared considering the Dataset CIFAR10 in two EMD configurations (0.71 and 1.41) in the two CNNs (LeNet5 and ResNet18).

Based on the communication costs defined in (2.9), we can calculate the value using the number of training rounds or synchronizations (m), the number of clients (K), and the weights of the local models (ω). The size of the transferred weights in Megabytes (MB) varies according to the CNN architecture used for local training, usually increasing for more complex networks. Table 4.6 details the cost in MegaBytes for each network architecture in question for each round (m).

Table 4.6: Total Communication Cost per Round for CNN Architectures

Architecture	Parameters (in MB)	No. of clients	Total Cost per Round(T)
LeNet5	4.27	10	42.7 MB
ResNet18	46.76	10	467.6 MB

Thus, the total training cost is the product of the value defined in Table 4.6 of the architecture used by the number of rounds (m) necessary to reach a certain test accuracy, multiplied by 2 to consider the uplink and downlink transmissions.

The results of Table 4.7a show that in the scenario with LeNet5, FedWS had lower communication costs than the others, presenting reductions between 29 to 50% for EMD 0.71 and from 29 to 50% for EMD 1.41, compared to the best result of the other techniques. Considering the results using ResNet18 in Table 4.7b, FedWS presented a lower cost than ResNet18 between 25% to 50% in EMD 0.71 and from

20% to 100% in EMD 1.41. These results confirm that under more significant heterogeneity, the technique is less affected by data distribution presenting lower communication costs.

Table 4.7: FL Communication Costs on Different Target Test Accuracies, Techniques, and CNN Architectures.

EMD	Technique	Target Test Accuracy (%)			
		40	50	60	70
0.71	NN	3,416	6,832	10,248	-
	BN	1,708	3,416	5,124	11,956
	GN	1,708	3,416	6,832	15,372
	LN	1,708	3,416	6,832	15,372
	FedWS	1,708	1,708	3,416	8,540
1.41	NN	8,540	13,664	-	-
	BN	3,416	8,540	13,664	-
	GN	3,416	8,540	13,664	-
	LN	3,416	8,540	13,664	-
	FedWS	1,708	3,416	11,956	-

(a) LeNet5: Total Communication Cost.

EMD	Technique	Target Test Accuracy (%)					
		40	50	60	70	80	90
0.71	ResNet18	18,704	18,704	37,408	37,408	74,816	187,040
	ResNet18	18,704	18,704	18,704	37,408	56,112	130,928
	FedWS						
1.41	ResNet18	37,408	56,112	74,816	93,520	-	-
	ResNet18	18,704	37,408	56,112	74,816	130,928	-
	FedWS						

(b) ResNet18: Total Communication Cost.

4.5 Experiment III: Impact of Heterogeneous Distributions on Convolutional Layers of CNN

The objective of this experiment is to investigate Research Question RQ3. Considering the use of normalization techniques mentioned previously to mitigate the effects of heterogeneous distributions in FL and that these techniques are widely used in conjunction with CNN architectures, an important factor is to evaluate the impact of data heterogeneity on the weights of convolutional layers.

Table 4.8: Architecture of LeNet5.

No.	Layer	Kernel	Shape	Parameters	Activation
01	Input	-	$(3 \times 24 \times 24)$	-	-
02	Conv2D	(5×5)	$(64 \times 24 \times 24)$	4,864	ReLU
03	Normalization	-	-	-	ReLU
04	MaxPooling2D	-	$(64 \times 12 \times 12)$	-	-
05	Conv2D	(5×5)	$(64 \times 12 \times 12)$	102,464	ReLU
06	Normalization	-	-	-	ReLU
07	MaxPooling2D	-	$(64 \times 6 \times 6)$	-	-
08	Fully Connected	-	(384)	885,120	ReLU
09	Fully Connected	-	(192)	73,920	ReLU
10	Fully Connected	-	(10)	10	Softmax

Normalization layers are positioned in the 3 and 6 layers of the LeNet5. If normalization is not used these layers are ignored.

Figure 4.3 presents a histogram of the weight distributions of the last convolutional layer of a LeNet5 network [27] used as architecture in local training of FL clients and extracted after aggregation using FedAvg, represented by the number 05 in the Table 4.8. As we can see, the homogeneous distribution (EMD 0) has a histogram with a shape that resembles a normal distribution and when we observe the distribution with a higher EMD (1.41) the histogram shows an unbalanced weight distribution.

When using normalization techniques in conjunction with LeNet5 convolution layers, we observed in Figure 4.3 that even with values of EMD 1.41, the weights appear to be more homogeneous, more close to a normal form.

Considering the use of the proposed technique, it is important to evaluate how the outputs of the convolution layer behave when subjected to heterogeneous distributions. Figure 4.3 shows the histogram of the LeNet5 architecture used as local training applying WS to the outputs of the convolutional layers. We can see that the result obtained is a more uniform and normal distribution compared to other normalization techniques, demonstrating the potential of using the technique.

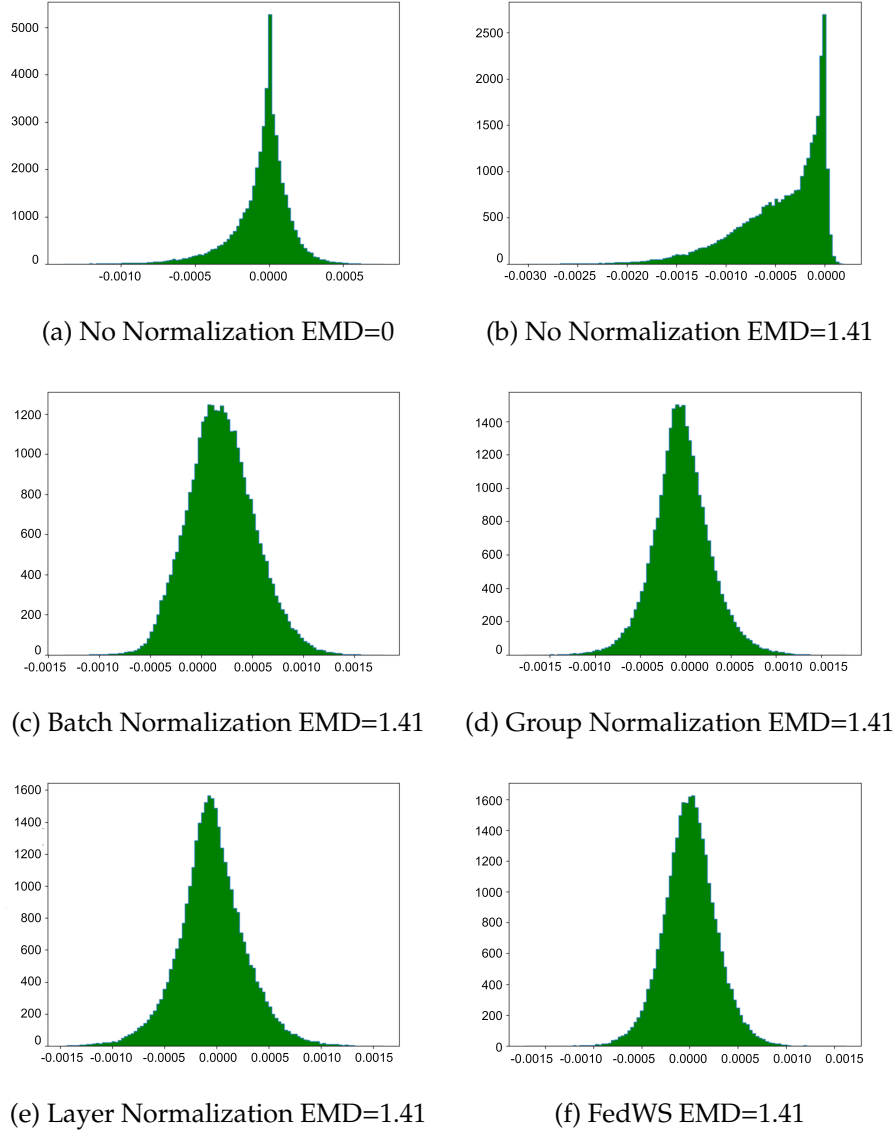


Figure 4.3: Histogram of Convolutional Layer on Round 1 of FL on EuroSAT Dataset

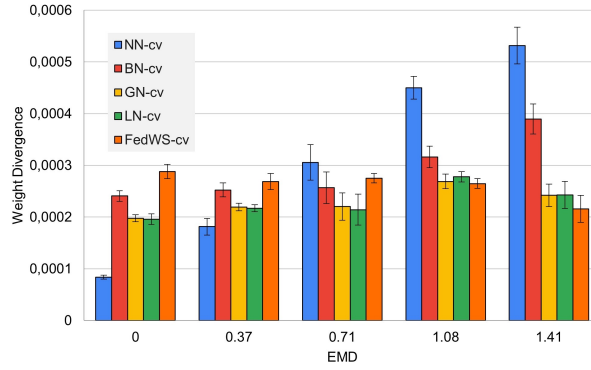
4.6 Experiment IV: Computing Weight Divergence

The weight divergence can be computed as the relative difference between the weights in FL training (ω^{FedAvg}) and the centralized SGD training (ω^{SGD}), as shown in:

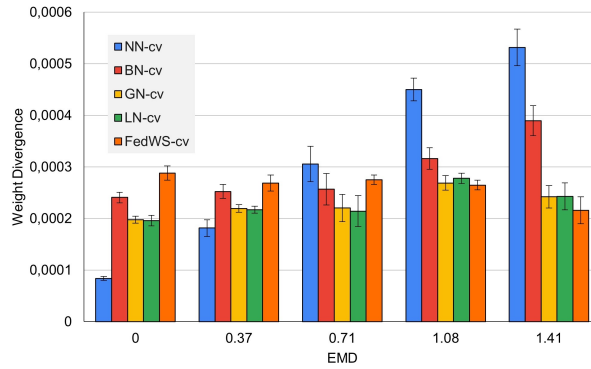
$$\text{weight divergence} = \frac{\|\omega^{\text{FedAvg}} - \omega^{\text{SGD}}\|}{\|\omega^{\text{SGD}}\|}. \quad (4.1)$$

The goal of this experiment is to investigate Research Question RQ4 comparing the weight divergence results from the defined scenarios against datasets MNIST, CIFAR10, and EuroSAT in data distributions with different EMD values. The values of weight divergence are computed on neural network convolutional layers in the first round of federated synchronization as defined by [62] on LeNet5 architecture. As this measurement is carried out in the first round, it does not

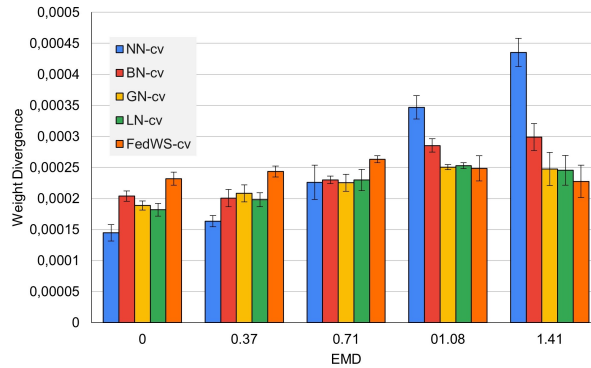
have a direct correlation with the results shown in Table 4.4 which take into account all rounds.



(a) MNIST Weight Divergence Results



(b) CIFAR10 Weight Divergence Results



(c) EuroSAT Weight Divergence Results

Figure 4.4: Weight Divergence Results on Different Datasets, Techniques, and Levels of EMD at round 1 of FL

In Figure 4.4, the difference in the divergence of the BN, GN, LN, and FedWS techniques compared to the result of NN indicates how they manage to mitigate the effect of EMD in the first round as shown on Remark 1.

Figures 4.4a and 4.4b show that both MNIST and CIFAR10 have very similar weight behavior in their convolutional layers. This result showed that NN has lower initial values; however, as the EMD values rise, the divergence rises, too. The normalization techniques (BN, GN, and LN) start with higher divergence

values but suffer less from increased EMD values, showing a smooth variation. This behavior is the same for EuroSAT in Figure 4.4c. In both scenarios, FedWS has a higher divergence on initial EMD values, with a lowering tendency as the EMD values rise.

These results showed that FedWS reduces the impact of increasing divergence on higher EMD values, as he does not tend to grow in such situations, performing better than the other normalization techniques. This result confirms the reduction in the impact of EMD on the divergence of weights, which has the greatest weight in the first round, as indicated by Remark 1. In some cases with lower EMD (0, 0.37, and 0.71) on which the divergence of FedWS is higher than the others, which indicates not necessarily reducing the initial divergence implies better accuracy results. As discussed in Section 3.1, other factors such as learning rate (η), the gradient descent (∇), and the loss of the local training step $l(w; b)$, can change the impact of EMD on accuracy [62]. In the following subsection, we can better examine this behavior by observing the results of gradient experiments.

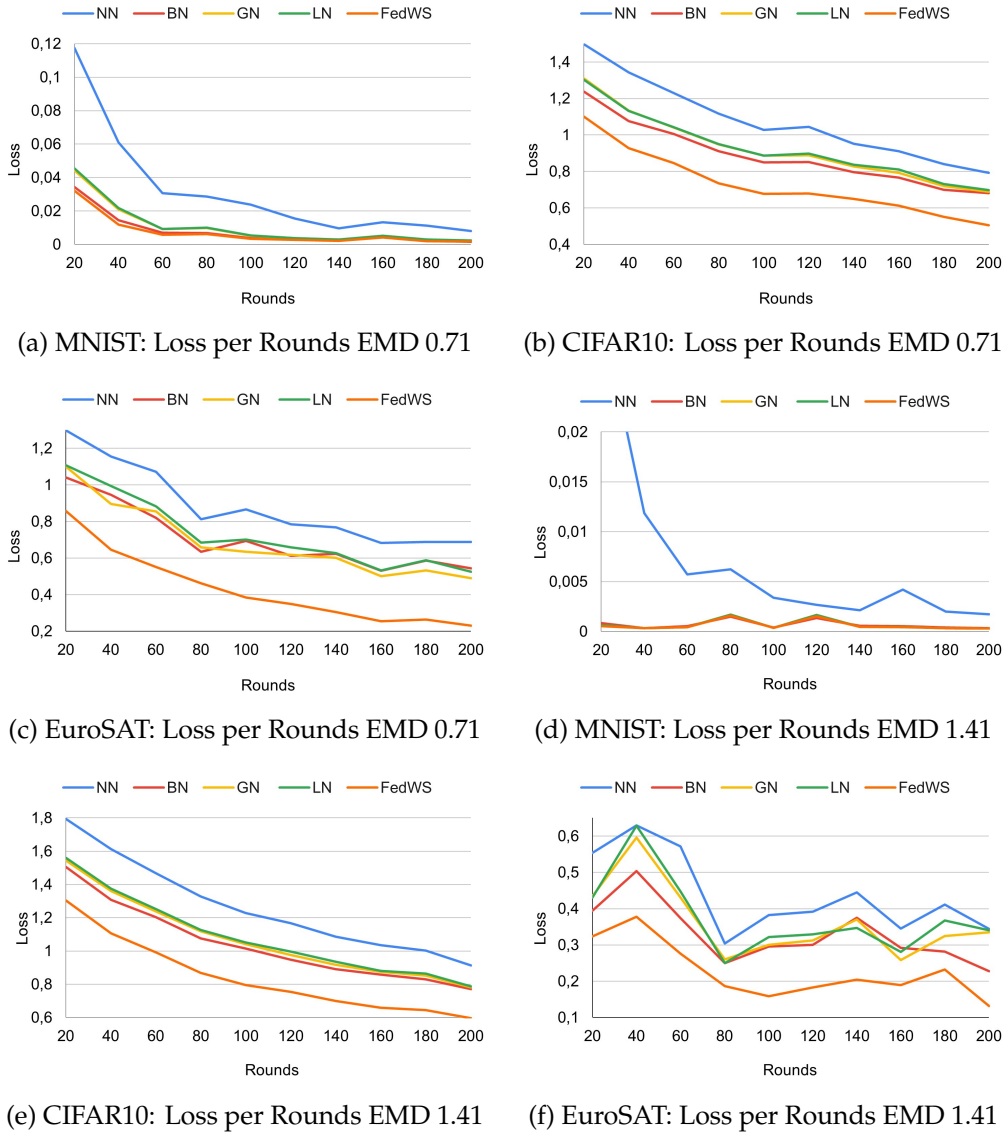


Figure 4.5: Loss on FL with MNIST, CIFAR10 and EuroSAT Dataset on two distributions

4.7 Experiment V: Losses and Gradient Behavior

The objective of this experiment is to investigate Research Question RQ5, considering the behavior of losses and gradients in the observed techniques.

One of the objectives of the FedWS is to smooth the FL training loss curve, and when we observe the results of Figure 4.5, it is clear that our proposal was confirmed, with FedWS having the smoothest loss curve and with smaller values in comparisons with other techniques taking as a sample the distributions with EMD 0.71 and 1.41. Even in the distribution with the highest level of heterogeneity, their values were smoother than the other participants.

Table 4.9: Gradient Variation on Different Techniques and Datasets using Samples with EMD 0.71 and 1.41 for FL and Lenet5 Architecture for Local Training

Technique	Mean	Standard Deviation	Technique	Mean	Standard Deviation
NN	-1.73×10^{-4}	6.5×10^{-4}	NN	-6.02×10^{-5}	4.9×10^{-3}
BN	-5.74×10^{-6}	2.8×10^{-4}	BN	-7.40×10^{-5}	6.2×10^{-3}
GN	-6.44×10^{-6}	2.9×10^{-4}	GN	-8.88×10^{-5}	3.2×10^{-3}
LN	-8.44×10^{-6}	3.0×10^{-4}	LN	-6.60×10^{-5}	3.2×10^{-3}
FedWS	-6.40×10^{-14}	2.0×10^{-11}	FedWS	-2.05×10^{-11}	9.2×10^{-10}

(a) MNIST Gradient Results (EMD 0.71)

(b) CIFAR10 Gradient Results (EMD 0.71)

Technique	Mean	Standard Deviation	Technique	Mean	Standard Deviation
NN	-7.91×10^{-4}	2.7×10^{-2}	NN	-1.14×10^{-4}	6.0×10^{-4}
BN	-2.83×10^{-5}	2.3×10^{-2}	BN	-6.08×10^{-8}	2.4×10^{-4}
GN	-6.49×10^{-5}	6.0×10^{-3}	GN	-3.42×10^{-6}	2.7×10^{-4}
LN	-3.59×10^{-5}	5.4×10^{-3}	LN	-4.67×10^{-6}	2.7×10^{-4}
FedWS	-1.48×10^{-12}	1.7×10^{-9}	FedWS	-5.67×10^{-13}	1.4×10^{-11}

(c) EuroSAT Gradient Results (EMD 0.71)

(d) MNIST Gradient Results (EMD 1.41)

Technique	Mean	Standard Deviation	Technique	Mean	Standard Deviation
NN	-3.35×10^{-5}	3.3×10^{-3}	NN	-1.11×10^{-4}	2.0×10^{-2}
BN	-3.40×10^{-4}	4.6×10^{-3}	BN	-2.91×10^{-4}	1.8×10^{-2}
GN	-8.59×10^{-5}	1.6×10^{-3}	GN	-9.01×10^{-5}	3.9×10^{-3}
LN	-9.01×10^{-5}	1.8×10^{-3}	LN	-1.10×10^{-4}	4.2×10^{-3}
FedWS	-9.51×10^{-12}	5.2×10^{-10}	FedWS	-1.00×10^{-11}	1.1×10^{-9}

(e) CIFAR10 Gradient Results (EMD 1.41)

(f) EuroSAT Gradient Results (EMD 1.41)

The smoother behavior of the losses indicates that the technique has a better result in heterogeneous situations. However, we must consider the results of

Table 4.9. Remark 2 states that any technique that produces smooth gradient values would reduce the impact of EMD in the divergence of the weights in the current round of FL and in this scenario, the most considerable part of the weight divergence comes from the divergence of the weights from the previous round. Table 4.9 shows us that the average of the gradients achieved by FedWS in all scenarios is much lower than the other techniques, with much smaller standard deviation variations. The differences range between 10^{-6} and 10^{-8} , reaching very small values. This result indicates that FedWS reduces the impact of the EMD in FL rounds after the initial round, which will have a greater impact on divergences arising from previous rounds.

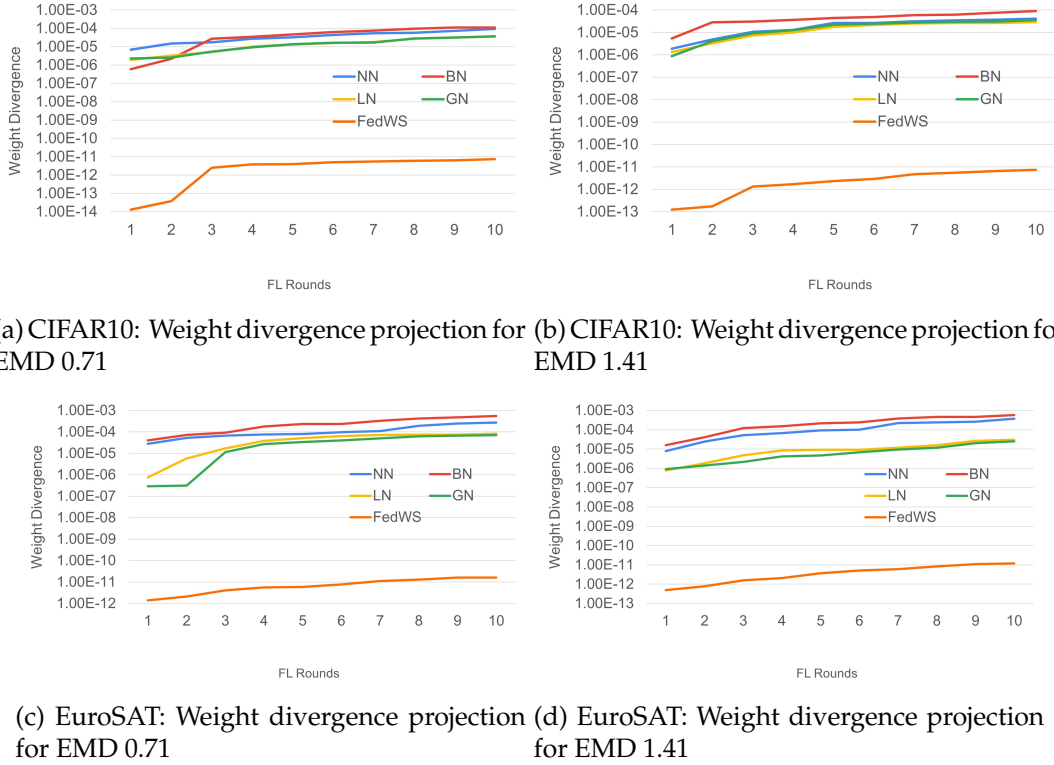


Figure 4.6: Projection of weight divergence considering the gradients of NN, BN, LN, GN and FedWS in 10 first FL Rounds, using Lenet5 CNN architecture a learning rate of 0.01 on datasets CIFAR10 and EuroSAT with an EMD of 0.71 and 1.41. Results are in logarithmic scale.

In this context, we can use the values of the gradients resulting from the experiment and apply them to Table 3.1 to have a projection of the impact of FedWS compared to NN, BN, LN, and GN on total weight divergence per round, considering all layers of a CNN. Furthermore, we can illustrate a projection of the impact of the behavior of gradients of FedWS in weight divergence as shown in Figure 4.6, considering the gradients on the first 10 rounds of FL of NN, BN, LN, GN and FedWS techniques with samples of CIFAR10 and Eurosat with EMD 0.71 and 1.41 using Lenet5 architecture for local training. Analysing the projection we observe that FedWs can produce less weight divergence than other techniques during FL training due to your smooth gradients.

Considering the results observed above and those shown in Section 4.6 we can revisit the Remarks 1 and 2. From **Remark 1** we can identify in the results

of Section 4.6 that FedWS managed to ensure that the increase in EMD does not necessarily increase initial divergence. From **Remark 2** we observed that the results of smooth and smaller gradients and losses result in the biggest impact for subsequent rounds being the divergence of previous rounds.

Therefore, we conclude that the proposed technique reduces the initial impact of the EMD, resulting in a more stable initial divergence. Furthermore, in subsequent rounds, the biggest impact becomes the initial divergence, which in this case is more stabilized. This corroborates the results observed in Section 4.3, which shows that FedWS receives less impact on its accuracy in more heterogeneous distributions.

4.8 Experiment VI: Comparison with the Centralized Scenario

The objective of this experiment is to investigate Research Question RQ6 and evaluate how the proposed solution behaves considering a centralized scenario.

Typically, FL has lower accuracy than centralized training (SGD) due to its decentralized nature. This experiment evaluated the difference in accuracy between FedWS and the original technique (GNWS) [44] in a centralized scenario. The baselines are conventional SGD learning without normalization (SGD NN) and FL without normalization (Fed NN).

The FL execution was carried out with five local epochs and 200 training rounds with batch size 50; for the case of SGD, the batch size was increased ten times since FedAvg performs an average of 10 clients at each synchronization[62]. For this experiment, MNIST, CIFAR10, and EuroSAT Datasets were used with their natural configuration corresponding to EMD with a value of 0, representing a homogeneous distribution.

Table 4.10: FedWS vs Centralized Training

Technique	MNIST Accuracy (%)	CIFAR10 Accuracy (%)	EuroSAT Accuracy (%)
SGD NN	99.13	81.22	83.41
Fed NN	98.79	71.64	77.06
SGD GNWS	99.28	86.47	88.50
FedWS	99.08	79.72	84.42

The results in Table 4.10 show that when comparing the difference in accuracy between SGD without normalization (SGD NN) and FL without normalization (Fed NN), the penalty for using the FL technique was approximately 0.34%, 9.58% and 6.35% for MNIST, CIFAR10, and EurSAT Datasets. When comparing SGD with GNWS and FedWS, these values change to 0.20%, 6.75% and 4.08%. These results indicate that the penalty for using FedWS instead of centralized training is smaller than the baseline used.

Table 4.11: Gradients Variation on Centralized Training

Dataset	Technique	Mean	Standard Deviation
MNIST	SGD NN	-5.26×10^{-5}	5.6×10^{-4}
	SGD GNWS	-9.39×10^{-13}	1.44×10^{-11}
CIFAR10	SGD NN	-1.74×10^{-4}	4.7×10^{-3}
	SGD GNWS	-2.20×10^{-11}	5.2×10^{-10}
EuroSAT	SGD NN	-5.84×10^{-4}	2.5×10^{-2}
	SGD GNWS	-1.22×10^{-13}	1.4×10^{-9}

Another important point is to verify the behavior of the gradients in the centralized scenario. Table 4.11 shows that the gradients of the centralized technique (SGD GNWS) have values of the order of magnitude compatible with the results obtained in Table 4.9, indicating that the use of the proposed technique does not suffer a considerable negative impact on the level of aggregation.

4.9 Experiment VII: Increasing Local Rounds in Federated Learning

The objective of this experiment is to investigate Research Question RQ7. According to [62] the number of local epochs influences the impact that EMD has on federated training in heterogeneous data distributions. Therefore, it is interesting to check the effect of increasing the number of local epochs (E) on test accuracy. Table 4.12 shows that all techniques showed an increase in accuracy, varying between 1.1% to 6.7%, considering that the local epochs were doubled, the gain in accuracy was not very significant.

Table 4.12: Test accuracy percentage in the EuroSAT Dataset using 10 local epochs, showing the comparative gain with 5 epochs

EMD	0	0.37	0.71	1.08	1.41
NN	81.9 (4.9)	80.9 (4.7)	80.0 (5.2)	75.2 (3.3)	69.7 (4.9)
BN	86.0 (1.3)	84.3 (1.9)	84.8 (2.7)	79.9 (2.1)	74.4 (1.9)
GN	85.0 (3.0)	83.7 (2.9)	83.3 (3.2)	77.9 (2.9)	75.4 (4.0)
LN	84.4 (3.3)	83.8 (2.9)	83.0 (3.3)	77.5 (3.1)	74.2 (3.8)
FedWS	87.7 (3.3)	87.1 (2.6)	86.7 (2.5)	83.5 (3.2)	79.6 (2.5)

Comparative with test accuracy between parentheses. Positive values indicate that the execution with 10 epochs obtained greater test accuracy

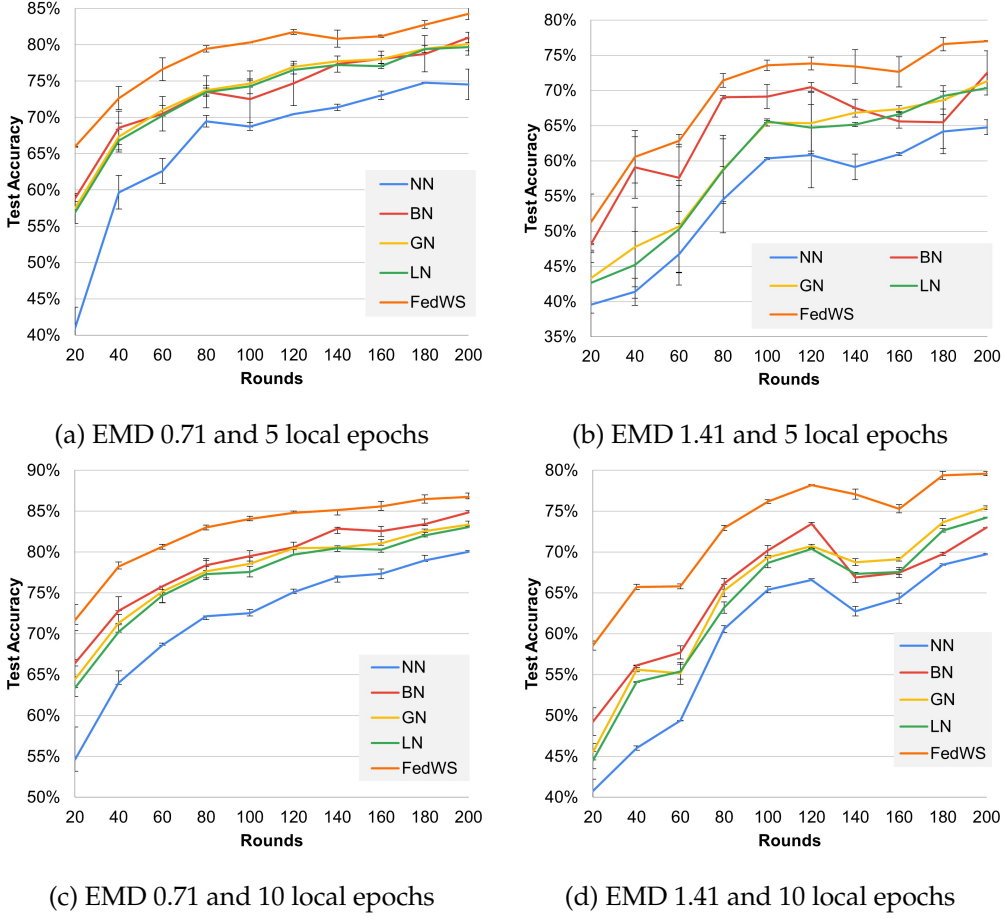


Figure 4.7: Convergence analysis in classification task on EuroSAT

The convergence of accuracy, which indicates how long the learning process took to reach a certain accuracy, is another characteristic that can be observed to evaluate the behavior of normalization techniques in heterogeneous data distributions. Figure 4.7 shows the test accuracy obtained by FL techniques on the EuroSAT Dataset with EMD 0, 0.71, and 1.41, considering 5 and 10 epochs of local training at clients. Considering the ability of the techniques to reach a level of 75% accuracy, the results for 5 epochs of local training show that FedWS managed to reach this accuracy using 50, 60, and 170 rounds for this, while the other techniques required 80 and 100 rounds and for EMD 1.41 were unable to reach this level. For 10 local rounds, we observed that the increase in local epochs increased the convergence of the techniques and reduced the number of rounds to reach the level of 75% at 30, 30, and 70 for FedWS and 50, 60, and 200 for the other techniques. These results indicate that the increase in local epochs, despite not greatly affecting the final result as observed in Table 4.12, has a more beneficial effect on the convergence of techniques.

Table 4.13 shows the results of the communication cost obtained by the techniques aiming to reach 75% in federated training. Using 5 local epochs, FedWS presented lower costs between 25% and 40% compared to the best results of other techniques. Furthermore, considering the increase in cost in relation to the homogeneous distribution scenario (EMD = 0), it presented the same cost for EMD = 0.71 and a cost three times higher in the scenario with EMD 1.41 in which the others were not even able to achieve the desired accuracy. Considering that EMD

0.71 has a medium unbalance of classes in clients (5 to 10) and EMD 1.41 has a more acute unbalance (1 to 2 classes), it shows that the technique's cost is little affected in lighter unbalances, and in more acute ones, despite achieving accuracies not achieved by others, it has a greater impact on cost.

Table 4.13: Communication Cost (MB) for Normalization Techniques to Achieve 75% Test Accuracy with EuroSAT Dataset

Local Epochs	EMD	NN	BN	GN	LN	FedWS
5	0	13,665	6,832	8,540	8,540	5,124
	0.71	15,372	10,248	8,540	8,540	5,124
	1.41	-	-	-	-	15,372
10	0	8,540	5,124	5,124	5,124	3,416
	0.71	10,248	5,124	5,124	5,124	3,416
	1.41	-	-	17,080	-	8,540

When analyzing the impact of increasing local epochs in training, from 5 to 10, we observed that previous results showed a relative gain in accuracy, as shown in Table 4.12, but a substantial gain in convergence as shown in Figure 4.7. In Table 4.13 we see that the increase in local epochs resulted in a reduction in communication cost between 25% and 50% for all techniques. Looking at these results alone, it may seem that increasing local epochs can be used as a solution to reduce communication costs, however, the processing cost must be taken into account.

As defined by Section 2.3 the processing cost is proportional to the product of the total number of rounds by the number of local training epochs. Using this definition, we analyzed the results shown in Table 4.14, which show the value of this product for each technique spent to achieve an accuracy of 75%. Observing the issue of techniques, FedWS presented a lower processing cost than other techniques of 25% in the homogeneous scenario (EMD=0) and 40% with an EMD value of 0.71.

Table 4.14: Computational Cost (number of synchronizations) to Achieve 75% of Test Accuracy on EuroSAT Dataset

Local Epochs	EMD	NN	BN	GN	LN	FedWS
5	0	6,000	4,000	5,000	5,000	3,000
	0.71	9,000	6,000	5,000	6,000	3,000
	1.41	-	-	-	-	9,000
10	0	10,000	6,000	6,000	6,000	4,000
	0.71	12,000	6,000	6,000	6,000	4,000
	1.41	-	-	20,000	-	10,000

Considering a scenario with 10 local epochs and an EMD of 0.71, normalization techniques increased the processing cost between 20% to 25%. Compared to the

reduction in communication cost observed in Table 4.13 between 25% to 50% in this context, the increase can be beneficial for most techniques aiming to obtain greater accuracy and reduce the cost of communication. However, the processing cost was measured considering the training process as a whole for all clients involved. If we consider each client individually, the increase in processing cost is 100% as the times increased from 5 to 10. In a scenario where clients have battery and processing capacity restrictions, this increase may not be feasible. In this case, FedWS can be an excellent alternative, as the reduction in communication costs obtained by other techniques when increasing local epochs was 20% to 50%, while FedWs using 5 local epochs achieved results of 25% and 40% the rest. This can be seen in Table 4.13 where the cost of FedWS for 5 local epochs was 5,124 MB for EMD 0 and 0.71 while the best technique using 10 local epochs reached the same cost. In other words, FedWs can be used in place of other techniques for use on devices that can use small local epochs due to capacity limitations.

Observing the results, we can identify that there is a trade-off in using more local epochs and reducing the communication cost versus increasing the computation cost. We can relate the computational cost to the number of local iterations (epochs) and the communication cost to the number of global iterations. Considering the results of the Tables and a target accuracy of 75%, the computational and communication costs have a competing relationship, as an increase in one leads to a decrease in the other. According to [19], there is an inverse relationship between local iterations and global iterations aiming to achieve global accuracy, considering the contribution of local training to FL considering the target accuracy, as represented in Figure 4.8.

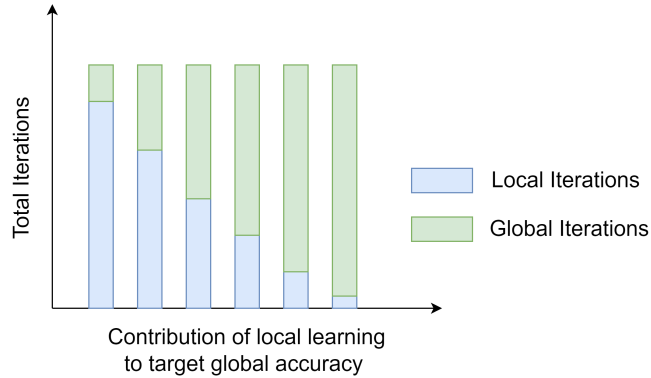


Figure 4.8: Representation of contribution of local and global iterations to achieve a target test accuracy.

In this way, the more significant the contribution of local learning, the less the need for more global iterations to achieve a determined global accuracy objective. Determining how worthwhile increasing epochs is must take into account how effective increasing local iterations will be. The number of local iterations necessary to achieve local accuracy is directly influenced by two factors: the technique applied by the client to solve the problem and the dataset [41].

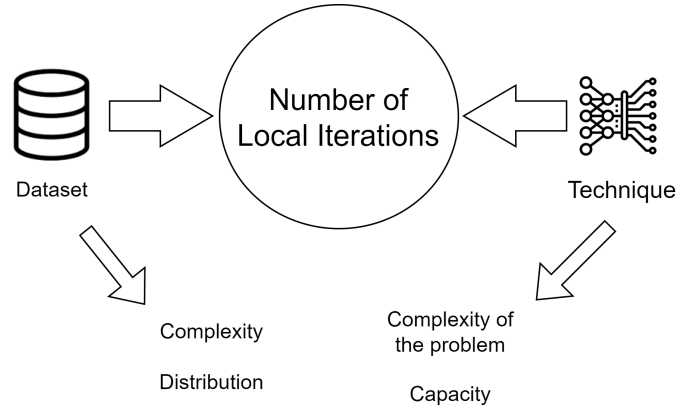


Figure 4.9: Factors that impact the number of local iterations in achieving a target accuracy.

Figure 4.9 highlights additional factors that impact how Datasets and techniques behave in FL. Therefore, more complex datasets tend to require more local epochs to increase convergence or use more complex neural networks (techniques). Furthermore, the heterogeneous data distribution may influence the need to increase epochs to maintain the same local accuracy. More complex problems may require more local epochs than simpler ones.

Table 4.15: Impact on computational and communication costs of rising local epochs from 5 to 10 on FL using FedWS on Dataset EuroSAT.

EMD	Impact on Communication Cost	Impact on Computational Cost
0	-33%	+33%
0.71	-33%	+33%
1.41	-44%	+11%

We can then evaluate the behavior of a specific scenario given a technique and data set used. Considering the execution of FL using the FedWS technique, using samples from the EuroSAT Dataset with EMDs 0, 0.71, and 1.41, we present the variation in communication and processing costs in Table 4.15 based on an increase of 5 to 10 local epochs.

Evaluating the results presented in this specific case, we identified that the increase in epochs from 5 to 10 allowed a reduction in the cost of communication between 33% and 44% with an increase in the cost of processing between 11% and 33%. If we only consider these results, the increase in processing cost was relatively low considering the reduction obtained in communication. However, the local processing cost for each client is considerably higher, considering the increase from 5 to 10 epochs. Therefore, how much local seasons clients can process can be a limiting factor in how much we can use this resource. Another limiting factor is the amount of data the client has at its disposal, as the increase in epochs in clients with a reduced amount of data can generate an overfitting effect in FL, producing a model with low generalization.

4.10 General Discussion

Considering the research questions and experiments, we present a general discussion of the results in this section.

Considering **RQ1** and **RQ2**, we observe that FedWS achieved test accuracy superior to the others between 2.41% to 5.0% and presented better behavior in higher EMD values. A similar behavior was observed when using the ResNet18 architecture, which natively uses BN; FedWS showed superiority between 1.36% and 4.48%. In terms of convergence, FedWS reached levels of accuracy in all datasets in fewer rounds of federated training than the BN, LN, and GN techniques, which in some cases were unable to reach the desired level. One point to highlight is that EMD is used in experiments only to generate a dataset distribution among clients considering a given distribution scenario. In real applications, quantifying the statistical heterogeneity of data in FL without full access to training data remains an open problem [52]. FedWS does not use any prior knowledge of data imbalance to work, and it has presented superior accuracy results with both homogeneous and heterogeneous data. This technique can be used in any of these scenarios.

Considering communication costs in response to **RQ1** and **RQ2** the experiments showed that FedWS performed with a lower cost, ranging from 20% to 100%, even with the increase in the neural networks' complexity. Much of this result comes from the high convergence observed in Section 4.3, where it achieved results in fewer rounds than other techniques.

In response to **RQ3**, the experiments demonstrated that the heterogeneous distributions led the weights and biases of the activation map after the convolution to present a distribution very far from normal, clearly showing a situation of neuron saturation. When used in this context, normalization techniques managed to improve this situation by bringing the activation maps closer to normal and with the potential for improvement in feature extraction and, consequently, in the results of FL training.

Observing questions **RQ4** about the divergence of weights in the initial round and **RQ5** about the behavior of gradients, we conclude that the proposed technique reduces the initial impact of the EMD, resulting in a more stable initial divergence. Furthermore, in subsequent rounds, the biggest impact becomes the initial divergence, which is more stabilized due to the behavior observed in Section 4.7. This corroborates the results observed in Section 4.3, which shows that FedWS receives less impact on its accuracy in more heterogeneous distributions. The experiments indicate that FedWS has results closer to the centralized scenario compared to the decentralized scenario, enhancing the use of FL for applications that require accuracy close to centralized but have a decentralized characteristic.

Considering how the techniques behaved with the increase in local training epochs, in **RQ7**, we observed that the increase in local epochs increased the final test accuracy slightly, influencing the convergence of techniques much more. Considering communication and computational costs, the results show a reduction in the communication cost between 20% to 50% for all techniques, but this increased the computational cost between 20% to 25%. In this context, the results of other techniques with ten local epochs are equivalent to those of FedWS with five local epochs, proving that the proposed technique achieves lower communication costs without increasing the computational cost related to the number of rounds/synchronizations in FL.

Based on the results of **RQ1**, **RQ3**, and **RQ7**, we can reach an extra understanding. According to the results of **RQ1**, the heterogeneous distribution of data

significantly affects the activation map resulting from the convolutional layers, which comprise the features extracted from the data by the CNN. The results of **RQ3** and **RQ7** show that using a neural network with a more robust architecture and greater capacity to extract features achieves a much more significant increase in accuracy than that provided by increasing local epochs in a network simpler.

5. Conclusion

In this work, after carrying out the experiments and discussion, we concluded that the BN, GN, and LN normalization techniques, when used in local FL training to reduce the impact of heterogeneous data distributions, presented high communication costs due to low convergence in training. In response to the first primary research question, normalization techniques did not show satisfactory results with communication costs. Considering this result, we propose a FedWS technique to reduce the communication cost while maintaining excellent and consistent test accuracy results. FedWS achieved superior results in test accuracy between 2.41% to 5.0% and presented better behavior in higher EMD values. A similar behavior was observed when using the ResNet18 architecture, which natively uses BN; FedWS showed superiority between 1.36% and 4.48%. Considering convergence and communication costs, the presented technique reached accuracies in fewer training rounds than the others, culminating in better communication cost results, in which it presented lower costs than the others, ranging from 20% to 100%. In that way, we demonstrated a technique that answers the second central research question, improving the state-of-the-art results.

In addition to answering these questions, other results were observed during the study, indicating that the heterogeneous distribution strongly impacts the convolution layer of a CNN, affecting the extraction of features performed by the neural network. Another point to highlight is how FedWS achieved the highlighted results. Considering the impact of weight divergence in FL aggravated by heterogeneous distributions, we were able to show that FedWS reduces the effect of divergence in the initial round of FL, suffering less impact from the variation measured by EMD when it increases. Furthermore, we identified that smaller gradients produce a beneficial result in subsequent steps, significantly reducing the final impact caused by heterogeneous distributions and making the technique more resilient to increased EMD. An important point to highlight is that the proposed technique does not require prior knowledge of data imbalance for it to work well. It obtains superior results in both homogeneous and heterogeneous distributions. Delving deeper into the issue of communication and computational costs, the results show that FedWS reduced the communication cost between 25 and 40 compared to other techniques using five local training epochs. When we increased the local training epochs to 10, there was a reduction in the communication cost between 20% to 50% for all techniques, but this increased the computational cost between 20% to 25%. When comparing the results, we observed that FedWS achieves communication costs that the others only achieved with an increase in local epochs and, consequently, in the computational cost. In this scenario, FedWS becomes appropriate for devices with processing and battery resource constraints and does not support the increase in local epochs.

5.1 Limitations

Below, we will discuss some limitations identified during the work. We highlight the following limitations, mainly involving points that could have been further explored.

Limitation of the effectiveness of the proposed technique. Considering the application of the technique on the extraction layers, we can identify a limitation that may impact the desired effects of the technique. The application proposed in (3.2) depends on the values of ω and σ to obtain the desired results, making the layer's output a normal distribution with mean 0 and standard deviation 1. However, as already observed by [44] in the WS study, situations in which the weights have a concentrated distribution around 0 can harm the effectiveness of the technique. Analyzing this consideration further, we would have two possibilities: the distribution of weights in question would already be normalized, or the distribution has many values close to zero and is not normalized. In this second case, applying the average proposed in (3.2) could lose the desired effect of smoothing the gradients, and the technique could have results below expectations.

Exploration of global and local training hyperparameters. The experiments were carried out with fixed values of the global and local FL hyperparameters. Varying some of these parameters could further enrich the discussion about how the methods behave. Considering the batch size specifically, larger values of this could improve the BN results, and we could check if this allows the results between FedWS and BN to be closer. However, testing high batch size values requires much memory to carry out the experiments and can limit its usability in FL clients with low capacity.

Variation of convolutional layers and neurons. The work did not explore how increased data heterogeneity affects CNNs considering different network sizes, whether using more or more extensive layers. This type of analysis could indicate a result that is more based on comparing the size of the extraction layers than a specific focus on architectures such as Lenet5 or ResNet18.

Variation in client status and mobility. The work did not consider the issue of client mobility and the limitation of their resource consumption. Considering that one of the study's objectives was to study communication and processing costs, situations where problems occur that could cause the interruption of the participation of FL customers were not considered, such as transmission interruption and low communication signal. In this way, exploring this type of scenario comes closer to reality.

Some of these points will be highlighted as opportunities for future work in the next section.

5.2 Future Works

In future work, we will indicate the above possible initiatives to explore new potential collaborations identified during the work.

Evolve the study of measuring data heterogeneity into problems other than data classification. Using EMD to measure how data is distributed is heavily based on the type of data classification problem. It is worth investigating how

to measure statistical heterogeneity in problems other than this one. This type of investigation can contribute to understanding the problems and enhance progress in working on other types of data distributions in other contexts.

Study the use of FedWs in other feature extraction layers. Another point that can be advanced is to study how this proposed technique can be used in another type of neural network layer to improve the neural network's feature extraction, thus increasing its generalization. As discussed in Section 4.10, data heterogeneity significantly affects the feature extraction capacity, influencing the CNN activation map. Observing how this type of problem affects other feature extraction methods in FL can lead to a better understanding of the problem and possible solutions.

Evaluate FedWS in a practical application. Another issue that can be explored is to evaluate the solution in a practical application with scenarios using mobile devices to observe other essential factors in FL for operation in this type of client, such as client participation criteria, client failures, and communication failures between the client and the central server.

Study a method of measuring data heterogeneity during FL training. As seen in Section 4.10, FedWS does not need knowledge of how and how much the data is imbalanced to work. Furthermore, we observed that using more complex CNNs has a better effect on test accuracy than increasing the number of local rounds. However, using more robust CNN architectures may be prohibitive for clients with lower computational capacity. A promising scenario would be for the technique to be able to choose different architectures, taking into account the degree of heterogeneity of client data and its processing power. In this context, a proposal for future study is how to measure the distribution of client data among themselves at FL execution time and how to use FedWS in conjunction with a technique such as Federated Ensemble Learning [34] that allows the execution of different architectures between clients on FL.

Evaluate the use of FedWS with other techniques with the potential to lower communication and computational costs. There are centralized learning techniques such as Early Exit Neural Network (EENN) [40], which proposes an approach that consists of identifying that the majority of input data used in a classification problem does not need to go through all the layers of a CNN to obtain an optimal result, proposing an early exit in this cases. In this way, training time and local processing costs are greatly reduced. Using this type of approach together with FedWS in local FL training can enhance FL convergence results and reduce communication and computational costs.

Bibliography

- [1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 7
- [2] Mohammed Aledhari, Rehma Razzak, Reza M Parizi, and Fahad Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725, 2020. 1
- [3] Muhammad Asad, Ahmed Moustafa, Takayuki Ito, and Muhammad Aslam. Evaluating the communication efficiency in federated learning algorithms. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 552–557. IEEE, 2021. 14
- [4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *stat*, 1050:21, 2016. 11, 21
- [5] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation—a survey. 12
- [6] Flavia Causa, Armando Franzone, and Giancarmine Fasano. Strategic and tactical path planning for urban air mobility: Overview and application to real-world use cases. *Drones*, 7(1):11, 2023. 29
- [7] Suresh Dara and Priyanka Tumma. Feature extraction by using deep learning: A survey. In *2018 Second international conference on electronics, communication and aerospace technology (ICECA)*, pages 1795–1801. IEEE, 2018. 5
- [8] Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021. 8
- [9] Zhixu Du, Jingwei Sun, Ang Li, Pin-Yu Chen, Jianyi Zhang, Hai” Helen” Li, and Yiran Chen. Rethinking normalization methods in federated learning. In *Proceedings of the 3rd International Workshop on Distributed Machine Learning*, pages 16–22, 2022. 21, 30
- [10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011. 20
- [11] Arun Kant Dwivedi, Arun Kumar Singh, and Dharmendra Singh. An object based image analysis of multispectral satellite and drone images for precision agriculture monitoring. In *IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium*, pages 4899–4902. IEEE, 2022. 1

- [12] Sara FTAIMI and Tomader Mazeri. The role of big data in a connected environment case study: smart city. In *2019 International Conference of Computer Science and Renewable Energies (ICCSRE)*, pages 1–7. IEEE, 2019. 1
- [13] Benyamin Ghojogh and Mark Crowley. The theory behind overfitting, cross validation, regularization, bagging, and boosting: Tutorial. *arXiv e-prints*, pages arXiv–1905, 2019. 10
- [14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning, 2016. 9
- [15] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pages 195–201. Springer, 1995. 6
- [16] Mohd Anul Haq, Gazi Rahaman, Prashant Baral, and Abhijit Ghosh. Deep learning based supervised image classification using uav images for forest areas classification. *Journal of the Indian Society of Remote Sensing*, 49:601–606, 2021. 1
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 30
- [18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 2, 10, 11
- [19] Latif U Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 23(3):1759–1799, 2021. 15, 47
- [20] D Kinga, Jimmy Ba Adam, et al. A method for stochastic optimization. In *International conference on learning representations (ICLR)*, volume 5, page 6. San Diego, California;, 2015. 20
- [21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 29
- [22] Jon Krohn, Grant Beyleveld, and Aglaé Bassens. *Deep learning illustrated: a visual, interactive guide to artificial intelligence*. Addison-Wesley Professional, 2019. 5, 6, 7
- [23] Christos Kyrkou and Theocharis Theocharides. Emergencynet: Efficient aerial image classification for drone-based emergency monitoring using atrous convolutional feature fusion. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:1687–1699, 2020. 1
- [24] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. 12

- [25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 8
- [26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 29
- [27] Yann LeCun et al. Lenet-5, convolutional neural networks. URL: <http://yann.lecun.com/exdb/lenet>, 20(5):14, 2015. 37
- [28] Minhyeok Lee. The geometry of feature space in deep learning models: A holistic perspective and comprehensive review. *Mathematics*, 11(10):2375, 2023. 11
- [29] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge university press, 2020. 8
- [30] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv e-prints*, pages arXiv–2102, 2021. 20, 30
- [31] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020. 13
- [32] Su Liu, Jiong Yu, Xiaoheng Deng, and Shaohua Wan. Fedcpf: An efficient-communication federated learning approach for vehicular edge computing in 6g communication networks. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1616–1629, 2021. 15
- [33] Xiaodong Ma, Jia Zhu, Zhihao Lin, Shanxuan Chen, and Yangjie Qin. A state-of-the-art survey on solving non-iid data in federated learning. *Future Generation Computer Systems*, 135:244–258, 2022. 15, 29
- [34] Alhassan Mabrouk, Rebeca P Díaz Redondo, Mohamed Abd Elaziz, and Mohammed Kayed. Ensemble federated learning: An approach for collaborative pneumonia diagnosis. *Applied Soft Computing*, 144:110500, 2023. 53
- [35] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386, 2020. 5
- [36] Zhengxiong Mao, Hui Li, Zuyuan Huang, Chuanxu Yang, Yanan Li, and Zihao Zhou. Communication-efficient federated learning for power load forecasting in electric iots. *IEEE Access*, 2023. 2
- [37] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. 5
- [38] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 1, 12, 15, 20, 30

- [39] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003. 11
- [40] Priyadarshini Panda, Abhronil Sengupta, and Kaushik Roy. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pages 475–480, 2016. 53
- [41] Shashi Raj Pandey, Nguyen H Tran, Mehdi Bennis, Yan Kyaw Tun, Aunus Manzoor, and Choong Seon Hong. A crowdsourcing framework for on-device federated learning. *IEEE Transactions on Wireless Communications*, 19(5):3241–3256, 2020. 47
- [42] Seyedamin Pouriyeh, Osama Shahid, Reza M Parizi, Quan Z Sheng, Gautam Srivastava, Liang Zhao, and Mohammad Nasajpour. Secure smart communication efficiency in federated learning: Achievements and challenges. *Applied Sciences*, 12(18):8980, 2022. 2
- [43] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018. 5
- [44] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization. *arXiv e-prints*, pages arXiv–1903, 2019. 3, 22, 43, 52
- [45] Frank Rosenblatt et al. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*, volume 55. Spartan books Washington, DC, 1962. 6
- [46] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 59–66. IEEE, 1998. 18
- [47] Shaeke Salman and Xiuwen Liu. Overfitting mechanism and avoidance in deep neural networks. *arXiv e-prints*, pages arXiv–1901, 2019. 10
- [48] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018. 11, 20, 21
- [49] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000. 10
- [50] Debaditya Shome, Omer Waqar, and Wali Ullah Khan. Federated learning and next generation wireless communications: A survey on bidirectional relationship. *Transactions on Emerging Telecommunications Technologies*, 33(7):e4458, 2022. 15
- [51] Jordan Suzuki, Saba F Lamah, and Yasaman Amannejad. Using transfer learning in building federated learning models on edge devices. In *2021*

- Second International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pages 105–113. IEEE, 2021. 1
- [52] Alysa Ziyang Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 49
 - [53] Flavio Vieira and Carlos Alberto V Campos. Fedws: Dealing with heterogeneous data on federated learning. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2023. 3
 - [54] Flávio Vieira and Carlos Alberto V Campos. Fedws: Uma nova abordagem para aprendizado federado usando dados heterogêneos. In *Anais do XXII Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, pages 1–12. SBC, 2023. 3
 - [55] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018. 11
 - [56] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2, 11
 - [57] Yang Xu, Yunming Liao, Hongli Xu, Zhenguo Ma, Lun Wang, and Jianchun Liu. Adaptive control of local updating and model compression for efficient federated learning. *IEEE Transactions on Mobile Computing*, 2022. 15
 - [58] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019. 1
 - [59] Seda Yeşilmen and Bahadır Tatar. Efficiency of convolutional neural networks (cnn) based image classification for monitoring construction related activities: A case study on aggregate mining for concrete production. *Case Studies in Construction Materials*, 17:e01372, 2022. 1
 - [60] Fuxun Yu, Weishan Zhang, Zhuwei Qin, Zirui Xu, Di Wang, Chenchen Liu, Zhi Tian, and Xiang Chen. Fed2: Feature-aligned federated learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 2066–2074, 2021. 20, 30
 - [61] Zhengming Zhang, Yaoqing Yang, Zhewei Yao, Yujun Yan, Joseph E Gonzalez, Kannan Ramchandran, and Michael W Mahoney. Improving semi-supervised federated learning by reducing the gradient diversity of models. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1214–1225. IEEE, 2021. 20
 - [62] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv e-prints*, pages arXiv–1806, 2018. 17, 20, 31, 38, 40, 43, 44
 - [63] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021. 2, 15, 16, 20, 22