Gradient Descent on Wide Neural Networks and Federated Learning

Gisele Stephanie Otiobo (gisele.otiobo@aims-cameroon.org)
African Institute for Mathematical Sciences (AIMS)
Cameroon

Supervised by: Prof. Dr. Enrique Zuazua
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

20 May 2023

Submitted in Partial Fulfillment of a Structured Masters Degree at AIMS-Cameroon

Abstract

Deep neural networks are very complex and difficult to understand due to their intricate structure and a vast number of parameters. The relationship between parameters and predictors is entangled by the sequential composition of nonlinear functions and the non-convex objective presents further complications. Despite being too complex for classical learning theory, neural networks can successfully generalize in practice. This project focuses on a neural network with a homogeneous activation function and a wide hidden layer. We explored qualitative convergence, particularly global convergence guarantees that can be derived. The aim is to gain a good understanding of the associated gradient flow and connect it with the new paradigm of Federated Learning. Rather than examining finite-width neural networks, the study focuses on networks in the infinite width limit, where a distribution over weights and biases characterizes predictors. This enables a strong connection between these overparametrized networks and gradient flow. As we observed, numerical study shows that the more the number of neurons increases the more flow is observed and the better the convergence of the model. This technique would therefore revolutionize the way models are now trained and those also for Federated Learning approaches in order to make local training more efficient for a more efficient global model to be trained by collaboration. The study also analyzes the Federated Learning technique on the MNIST dataset and examined both Independent and Identically Distributed (IID) and non-Independent and Identically Distributed $(\textsc{non}\text{-}\mathrm{IID})$ data. We have therefore understood the functioning of the federated learning approach from the theoretical concept to its experimentation for a neural network with finite width, during this experimentation we obtained 94% performance for two models and 98% for four other models.

Keywords: Deep learning, Neural ODE, Wide Neural Networks, Gradient Descent method, Gradient Flow, Federated Learning, Global Convergence.

Declaration

I, the undersigned, hereby declare that the work contained in this essay is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

Gisele Stephanie Otiobo, 21 May 2023.

Contents

1. Introduction

The fields of machine learning and deep learning are advancing rapidly, with fresh algorithms and techniques being developed every day. Two such techniques that have gained significant attention in recent times are Gradient Descent (GD) and Federated Learning (FL). While GD is an optimizing procedure used to minimize loss functions in ML models, FL (subfield of ML), is a distributed learning framework that enables the training of models without the need for data centralization. In this essay, we will explore these two techniques in detail and discuss their implementation in FL.

Deep neural networks (DNN) are architectures that use linear and nonlinear operations to process input data samples represented as $\{\vec{x}_i\}_{i=1}^N \subset \mathcal{X} \subset \mathbb{R}^d$. As described in [29], Residual Neural Networks $(\textsc{ResNets})$ are specific types of neural networks which can be simplified into a common pattern

$$\begin{cases} z_i^{j+1} & = z_i^j + \boldsymbol{\sigma}\left(w^j z_i^j + \beta^j\right) \qquad \text{for all} \quad j \in \{0, \dots, N-1\} \\ z_i^0 & = \vec{x}_i \in \mathbb{R}^d, \end{cases} \tag{1.0.1}$$

for all $i \in \{1, \dots, N\}$, where $N \geqslant 1$ refers to the depth of the Neural network. The unknown variables are call states $z_i^j \in \mathbb{R}^d$ for any $i \in \{1, \cdots, N\}$. The activation function $\boldsymbol{\sigma}$ is a Lipschitz continuous nonlinear function defined component-wise in (1.0.1). The parameters $\left\{w^j, \beta^j\right\}_{j=0}^{N-1}$ are optimizable and consist of weights $w^j \in \mathbb{R}^{d\times d}$ and biases $\beta^j \in \mathbb{R}^d$.

During the training process, the goal is to find the best parameters that can steer all of the network outputs, represented by z_i^N, as close as possible to their corresponding labels, denoted by $\vec{y}_i$. To ensure reliable performance on new data, the training process involves finding the best parameters that will make the outputs z_i^N close to their corresponding labels $\vec{y}_i$. To achieve this, one needs to solve the following equation:

$$\min_{\{w^j, b^j\}_{j=0}^{N-1}} \frac{1}{N} \sum_{i=1}^{N} \ell\left(\Phi z_i^N, \vec{y}_i\right), \tag{1.0.2}$$

where the function $\ell(\cdot,\cdot)$ varies according to the task and is continuous and non-negative. For example, in regression tasks, $\ell(x,y) := \|x-y\|_{\ell^2}^2$ is commonly used, whereas in binary classification tasks where the logistic function $\ell(x,y) = \log(1+\exp(-y\,x))$, with output vector $\vec{y}_i \in \{-1,1\}$ could be employed. Additionally, $\Phi : \mathbb{R}^d \to \mathbb{R}^m$ is an affine map.

Recent studies have focused on analyzing the continuous-time formulation of $\textsc{ResNets}$. This trend began with the works in [54, 31]. The motivation behind this approach is the observation that equation (1.0.1) is essentially the Euler scheme for the ordinary differential equation, for any $i \in \{1, \cdots, N\}$, $T > 0$ and for all $\tau \in (0,T)$,

$$\begin{cases} \dot{z}_i(\tau) & = \boldsymbol{\sigma}(w(\tau) z_i(\tau) + b\beta(\tau)) \\ z_i(0) & = \vec{x}_i \in \mathbb{R}^d. \end{cases} \tag{1.0.3}$$

The use of continuous-time neural ordinary differential equations in deep learning has proven to be highly effective in applications [41, 47]. It is important to emphasize that this approach is related to the adjoint method used in optimal control through back-propagation and GD method [59].

As a starting point for understanding the foundations of FL and its applications in relation to data privacy, one may figure out what are Gradient method and its importance.

GD is an iterative optimization algorithm used in ML models to minimize loss functions as in (1.0.2) by adjusting the parameters. The algorithm works by computing the gradient of the loss function with respect to weights and biases and moving to the direction of the steepest descent. There are various forms of GD, including batch, mini-batch, and SGD. The choice of the GD algorithm depends on the dataset and the computational resources available. GD has become the most common way to optimize neural networks. It has proven to be an effective way to train ML models and is widely used in the industry.

Recent studies have shown that the use of GD in overparametrized neural networks can result in remarkably low training errors across various datasets [63, 44]. The authors of [15, 5, 14] analyze overparametrization on hidden neural networks. In their study, the authors analyze one hidden neural network with ReLU activation and introduced a Mean-field, Wasserstein Gradient Flow approach to finding the best parameters of the network. They also demonstrate that these parameters eventually lead to global minimums of the cost function, which can be described as a max-margin classifier. To learn more about related works in this direction, we suggest referring to the following sources [16, 46, 37].

FL enables collaborative training of ML models without centralized datas [25]. With FL, the data used for training is stored locally on devices, and only the updates to the model are transmitted to the central server. This approach offers significant benefits, such as increased data privacy, reduced communication costs, and greater scalability.

Implementing FL with GD has become a popular approach in ML. The combination of these two techniques allows the collaborative training of ML models while ensuring data privacy and reducing communication overheads. Several researchers have proposed various approaches to implement FL with GD [21, 62]. The implementation of FL with GD has shown promising results in several applications such as image classification, natural language processing, and speech recognition.

The unique properties of infinitely wide neural networks have caught the attention of professionals in the field of ML [15, 16, 5]. The use of infinitely wide neural networks has been shown to perform well in various ML models [5]. Although neural networks have been widely used, there is still a limited understanding of their theoretical properties and behavior. In this essay, we will examine how GD behaves on infinitely wide neural networks and explore the theoretical and empirical implications of this analysis.

The theoretical analysis of GD on infinitely wide neural networks has shown that these networks have unique properties that differ from finite neural networks. In particular, the convergence properties of infinitely wide neural networks are different from those of finite neural networks. For example, the convergence rate of infinitely wide neural networks is faster than that of finite neural networks [5].

This thesis suggests modifications to optimize infinitely wide linear neural networks to make FL practical. It also discusses how dedicated compression methods can be used. By utilizing differential privacy techniques and the GD method, significant information about individuals can be prevented from leaking when sending weight updates. Additionally, strategies for locally personalizing models can also be proposed.

Main scientific objectives.

In view of the above discussions, one is expected to break down and understand in a clear manner the transition from the GD method gradient flow on infinitely wide linear neural networks and further applied it to FL. The departure of this powerful method may be found in the framework of [5, 25, 15, 16, 46, 37] and references therein. To successfully deliver on this vision, the overreaching goal of the proposed thesis has three major objectives:

(O1) Explore the convergence guarantees of infinitely wide hidden layer neural network with ReLU activation function. We shall, first of all, focus on the main tools developed in [5, 25, 15, 16, 4] and later perform some numerical experiments.

(O2) Understand the construction of a general theory of FL and analyze it from a theoretical and numerical point of view. Two methods are introduced that will serve as the foundation for conducting numerical tests.

(O3) Finally, examine the possible extension of the question of FL on infinitely wide linear neural networks.

Through my work, we aim to achieve groundbreaking techniques for analyzing FL in infinitely wide linear neural networks. These techniques will be different from traditional deep learning, optimization, and optimal control-based approaches and can significantly enhance performance and robustness across various industries.

Roadmap of the thesis.

Chapter 2: Explain the key ideas behind the mathematical concepts of neural networks and provide clear definitions for related terms. Using these concepts, One can demonstrate that any continuous function is accurately approximated through neural networks. We further provide specific limitations on the number of parameters needed. This knowledge is crucial for understanding the universal flow approximation theorem and the overall global convergence for wide neural networks with one hidden layer.

Chapter 3: We will be explaining the usefulness of overparameterization through theoretical results. When width increases infinitely, the associated Gradient Flow (GF) will converge to the global optimum of the given cost function. The proof mainly relies on the homogeneity of the ReLU activation function. Our discussion focuses on the transition from GD optimization to GF, which is the most important part of the concept. However, we should note that the main weakness of this result is that it is only qualitative in nature. We can't be sure how large must be to approach the infinite latitude limit, or how quickly GF converges to the global optimum.

Chapter 4: We will explain FL and its mathematical principles using finite-width neural networks. We will also present a numerical analysis of six different model architectures. To demonstrate the effectiveness of FL, we apply the FedAvg algorithm to the task of handwriting recognition using the MNIST database containing handwritten digits. Our training set included $60,000$ images and we used $10,000$ images for model validation.

Chapter 5: We provide a conclusion and suggest some directions for future research. Chapter 3 highlights the unique properties of GD on infinitely wide neural networks, that differ from their finite counterparts. Although the use of infinitely wide neural networks can improve performance on various tasks, the computational cost may be a limiting factor. This chapter proposes further research to fully understand the behavior of GD on infinitely wide neural networks and explore their potential applications in FL. Our future work aims to combine the concepts from Chapter 3 and Chapter 4.

2. Preliminaries to neural network theory

In this chapter, you will find important information such as notations, definitions, and assumptions that will be useful in the upcoming chapters. These concepts have been sourced primarily from [1, 2, 20, 42].

2.1 Notations and assumptions

To simplify notation, we define the derivative $\dot{z}(\tau)$ of z with respect to τ at time τ and $[n]$ to be the set of integers from 1 to n. If h is a vector in $\mathbb{R}^n$, we write $h^\top$ to denote its transpose.

When referring to a vector $h \in \mathbb{R}^n$, we use the standard Euclidean norm denoted by $\|h\|$. However, when $h \in \mathbb{R}^{n\times m}$ is a matrix, we use the entry-wise Euclidean norm and also denote it by $\|h\|$.

$$\|h\| := \left(\sum_{i=1}^{n}\sum_{j=1}^{m} |h_{i,j}|^2\right)^{1/2}.$$

We use $\mathrm{Lip}(\mathbb{R})$ to denote the set of functions $g : \mathbb{R} \longrightarrow \mathbb{R}$, which are globally Lipschitz continuous. We recall from [2] that the bounded variational function space $\mathrm{BV}(0,T;\mathbb{R}^n)$ is defined as a space of integrable functions whose weak derivatives are finite Radon measures:

$$\mathrm{BV}(0,T;\mathbb{R}^n) := \Big\{ g \in L^1(0,T;\mathbb{R}^n) \colon \exists\, \mathrm{D}g \in \mathfrak{M}(0,T;\mathbb{R}^n) \text{ such that } \sum_{i=1}^{n}\int_0^T g_i(\tau)\psi_i'(\tau)\,\mathrm{d}\tau = -\sum_{i=1}^{n}\int_0^T \varphi_i(\tau)\,\mathrm{d}\mathrm{D}g_i(\tau), \quad \forall \varphi \in C_c^1(0,T;\mathbb{R}^n)\Big\}.$$

Here $\mathfrak{M}(0,T;\mathbb{R}^n)$ represents the set of finite radon measures on $(0,T)$ with values in $\mathbb{R}^n$. The space $\mathrm{BV}(0,T;\mathbb{R}^n)$ is equipped with a norm

$$\|g\|_{\mathrm{BV}(0,T;\ \mathbb{R}^n)} := \|g\|_{L^1(0,T;\ \mathbb{R}^n)} + \sum_{i=1}^{n} |\mathrm{D}g_i|_{(0,T)}.$$

2.1.1 Assumption. From this point forward, let us assume that we have been provided with a dataset for training purposes.

$$\{\vec{x}_i, \vec{y}_i\}_{i\in\{1,\dots,N\}} \subset \mathcal{X} \times \mathcal{Y},$$

where we have a set of points called $\mathcal{X} \subset \mathbb{R}^d$ where each point, represented by $\vec{x}_i \neq \vec{x}_j$ for $i \neq j$. For regression tasks, $\mathcal{Y}$ is a non-empty subset of $\mathbb{R}^m$, for binary classification, $\mathcal{Y} = \{-1, 1\}$, and for multi-label classification, $\mathcal{Y}$ is a subset of integers from 1 to m denoted by $[N]$.

To simplify things, we will use the notation: we have input data $\{\vec{x}_i\}_{i\in\{1,\dots,N\}}$ where $\vec{x}_i \in \mathbb{R}^d$. We define the vector $z^0 \in \mathbb{R}^{N\times d}$ using this input data

$$z^0 := [\vec{x}_1 \cdots \vec{x}_N]^\top \qquad \text{along with} \qquad z_i^0 = \vec{x}_i \quad \text{for } i \in \{1,\dots,N\}. \tag{2.1.1}$$

We will more often use (2.1.1) to other vectors generally denoted by $z \in \mathbb{R}^{N\times d}$ and defined the sub-vectors $\{z_i\}_{i\in\{1,\dots,N\}}$ with $z_i \in \mathbb{R}^d$: $z := [z_1 \cdots z_N]^\top$ When we use the notation z_i where i is in the range of $\{1,\dots,N\}$, it implies that z_i belongs to the vector space $\mathbb{R}^d$.This differs from treating only row i of the matrix $z \in \mathbb{R}^{N\times d}$ as a scalar.

2.1.2 Some definition and fundamental properties of neural networks.

2.1.3 Definition (Neural network)**.** Let $d, m, N \in \mathbb{N}$. A neural network of depth N, input dimension d and output dimension m is a tuple

$$\Theta^j := [w^j, \beta^j] \in \mathbb{R}^{d\times(d+1)}.$$

Matrix-vector pair, where $w^j \in \mathbb{R}^{d_{j+1}\times d_j}$ and $\beta^j \in \mathbb{R}^{d_j}$. If $N = 2$ we call this network flat, otherwise deep. We denote the entries of the matrices and vectors as the weights Θ of the network.

To make things easier, we define a vector-valued version of this function as $\boldsymbol{\sigma} : \mathbb{R}^d \longrightarrow \mathbb{R}^d$. In this regards, each component is defined as $\boldsymbol{\sigma}(z)_i := \boldsymbol{\sigma}(z_i)$ for $i \in [d]$. Each matrix-vector pair $[w^j, \beta^j] \in \mathbb{R}^{d\times(d+1)}$ induces an affine linear transformation, we denote

$$z_i^N = (\boldsymbol{\sigma} \circ \Lambda^k \circ \ldots \circ \boldsymbol{\sigma} \circ \Lambda^0)(\vec{x}_i), \quad \text{with} \quad \Lambda^j \vec{x} := w^j \vec{x} + \beta^j \quad \text{for all} \quad j \in \{0, \ldots, N\}. \tag{2.1.2}$$

An example of a neural network with two hidden layers is shown in Figure 2.1.

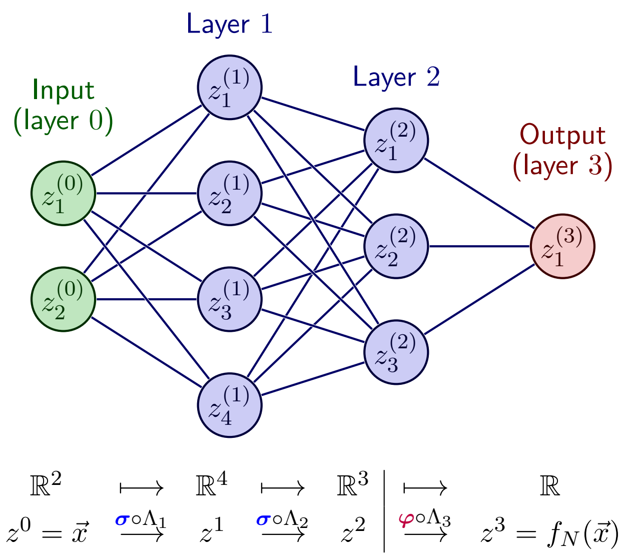

Layer 1
Layer 2
Input (layer 0)
Output (layer 3)
$z_1^{(0)}$
$z_2^{(0)}$
$z_1^{(1)}$
$z_2^{(1)}$
$z_3^{(1)}$
$z_4^{(1)}$
$z_1^{(2)}$
$z_2^{(2)}$
$z_3^{(2)}$
$z_1^{(3)}$
$\mathbb{R}^2 \longmapsto \mathbb{R}^4 \longmapsto \mathbb{R}^3 \longmapsto \mathbb{R}$
$z^0 = \vec{x} \xrightarrow{\boldsymbol{\sigma}\circ\Lambda_1} z^1 \xrightarrow{\boldsymbol{\sigma}\circ\Lambda_2} z^2 \xrightarrow{\varphi\circ\Lambda_3} z^3 = f_N(\vec{x})$

Figure 2.1: Neural network with two hidden layers

2.1.4 Definition (ReLU)**.** The Rectified Linear Unit (ReLU) also known as a special activation function is defined via $\boldsymbol{\sigma} \in \mathrm{Lip}(\mathbb{R})$ and $\boldsymbol{\sigma}(0) = 0$.

The most commonly used activation functions, such as sigmoids and rectifiers like $\boldsymbol{\sigma}(x) = \tanh(x)$, satisfy the assumption $\boldsymbol{\sigma}(x) = \max\{\lambda x, x\}$ For $\lambda \in [0, 1)$. Unless otherwise stated, we create activation functions $\boldsymbol{\sigma}$ that satisfy the following conditions:

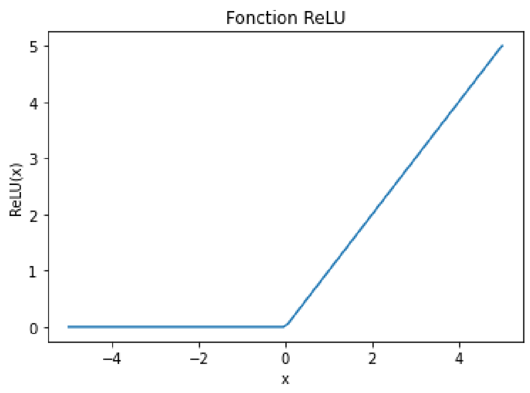


Figure 2.2: ReLU activation function.

2.2 Neural ordinary differential equations for learning

2.2.1 Forward neural networks. A multilayer perceptron (MLP) is a common example of a feed-forward neural network. It typically has a specific structure.

$$\begin{cases} z_i^{j+1} & = z_i^j + \sigma\left(w^j z_i^j + \beta^j\right) \quad \text{for all} \quad j \in \{0, \dots, N-1\} \\ z_i^0 & = \vec{x}_i \in \mathbb{R}^d, \end{cases} \tag{2.2.1}$$

for all $i \in \{1, \dots, N\}$; The equation (2.2.1) has a depth of at least one layer $(N \geqslant 1)$. In the neural network, every layer is identified by a variable called j and has a specific width known as d_j. The state of each layer is shown through the vector $z_i^j \in \mathbb{R}^{d_j}$. The weight and bias parameters that can be optimized in (2.2.1) are represented by $w^j \in \mathbb{R}^{d_{j+1} \times d_j}$ and $\beta^j \in \mathbb{R}^{d_j}$.

We use a fixed nonlinear activation function, which can be seen in Figure 2.2. This function is denoted as $\boldsymbol{\sigma}$ and belongs to the set of Lipschitz functions on the real numbers. One may notice that equation (2.2.1) aligns with the traditional approach to neural networks. This entails composing parametric affine operators and nonlinearities defined in (2.1.2).

It is worth emphasizing that by iteratively rearranging the parametric affine maps and the nonlinearity $\boldsymbol{\sigma}$ in (2.2.1), we can consider a system that is almost equivalent.

$$\begin{cases} z_i^{j+1} & = w^j \boldsymbol{\sigma}\left(z^j\right) + \beta^j \quad \text{for all } j \in \{0, \dots, N-1\} \\ z_i^0 & = \vec{x}_i \in \mathbb{R}^d. \end{cases} \tag{2.2.2}$$

Moving forward, we will focus on ResNets. Unlike the architecture of MLP defined as in (2.2.1) – (2.2.2), ResNets require a fixed width of $d_j = d$ for every layer j. In the context of fixed width, generally in the form of residual neural network is given as

$$\begin{cases} z_i^{j+1} & = z_i^j + f\left(\Theta^j, z_i^j\right) \quad \text{for all } j \in \{0, \dots, N-1\} \\ z_i^0 & = \vec{x}_i \in \mathbb{R}^d, \end{cases} \tag{2.2.3}$$

for $i \in \{0, \dots, N\}$, where $z_i^j \in \mathbb{R}^d$ for all i, j, $\Theta^j := [w^j, \beta^j] \in \mathbb{R}^{d \times (d+1)}$.

2.3 Universal approximation

The very first work in this direction is the one of [18], which proves that wide neural networks with increasing width, may approximate any continuous function on compact sets. Indeed, given g in some function class $\mathcal{K}$ and a parameter $\epsilon > 0$, there exists a neural network $g_{N(\epsilon)}$ with $N(\epsilon) > 0$ hidden layers, each layer consisting of $N_j(\epsilon)$ components, as well as parameters $\Theta(\epsilon)$, in a way that

$$\|g - g_\Theta((\epsilon))\|_{\mathcal{K}} < \epsilon.$$

The main theorem reads as

2.3.1 Theorem ([18])**.** *Let $\boldsymbol{\sigma} : \mathbb{R} \to \mathbb{R}$ be a nonconstant, bounded, and continuous function. Let $d \geq 1$ and $N = 1$. For any $\epsilon > 0$ and any $g \in C^0\left([0,1]^d\right)$, there exists $N_1 \in \mathbb{N}$, $w^0 \in \mathbb{R}^{N_1 \times d}$, $w^1 \in \mathbb{R}^{1 \times N_1}$ and $\beta^0 \in \mathbb{R}^{N_1}$ such that*

$$g_N(\vec{x}) = w^1 \boldsymbol{\sigma}\left(w^0 \vec{x} + \beta^0\right)$$

satisfies

$$\sup_{\vec{x} \in [0,1]^d} |g(\vec{x}) - g_N(\vec{x})| < \epsilon$$

In general, you can specify $N > 0$. Then more components are needed in the hidden layer to approximate g. It is worth noting that the sentence does not say what those parameters are or how to find them. It can basically be seen as a consequence of the existence of neural networks.

In simpler terms, the Universal approximation theorem states that a neural network can learn to represent any function, no matter how complex it may be [51]. This theorem is the theoretical foundation of why neural networks work, as it suggests that a neural network is capable of learning complex patterns and relationships in data as long as certain conditions are met. We also mention [26, 40] for an extension to other neural networks. The result of [20] is a bit of a dual with the one of [18]. To improve approximation accuracy, the width is allowed to increase, while we fix the width and allow the depth to increase. We refer to the results of the paper [28] and a comprehensive overview of universal approximation results for ResNets, and the recent work [24] for neural networks. In fact, we also have the following improved results.

The Universal Approximation Theorem provides a theoretical justification for the use of neural networks in these and other applications, and its implications continue to be studied and applied in the field of machine learning.

2.4 Some preliminaries on measure theory

In this thesis, we have used concepts and ideas from measure theory and optimal [1, 42]. Throughout my thesis, when we mention measure, we mean finite signed measures on $\mathbb{R}^d$ and their Borel σ algebras. For measurable set $\mathcal{X} \subset \mathbb{R}^d$, we adopt the notation $\mathcal{M}(\mathcal{X})$. To help with the proofs, we will now provide some of the concepts and facts from the measure theory that we have used.

The support and concentration set is a term used to describe a measure, denoted by $\operatorname{spt}\nu$, in $\mathcal{M}_+\left(\mathbb{R}^d\right)$. It refers to the points in $\mathbb{R}^d$ whose neighborhoods have a positive measure, or in other words, the largest open set with measure 0 excluded. In case the complement of a set Ω is included in a measurable set of measure zero, then we say that the measure ν is concentrated on Ω. In particular, ν is concentrated on $\operatorname{spt} \nu$. Let us refer to the definition of a pushforward function outlined in [17].

2.4.1 Definition (Pushforward)**.** Assume we have measurable subsets $\mathcal{X}$ and $\mathcal{Y}$ of $\mathbb{R}^d$, and a measurable map $\Pi : \mathcal{X} \to \mathcal{Y}$, then for any measure $\nu \in \mathcal{M}(\mathcal{X})$, there is a corresponding $T_{\#}\nu \in \mathcal{M}(\mathcal{Y})$, called the pushforward of ν by Π. It is defined as $\Pi_{\#}\nu(\mathcal{A}) = \nu\left(\Pi^{-1}(\mathcal{A})\right)$ for all measurable sets $\mathcal{A} \subset \mathcal{Y}$. It tells you about the distribution of the mass of ν after the map Π has been applied. It satisfies

$$\int_{\mathcal{Y}} \psi \mathrm{d}\left(\Pi_{\#}\nu\right) = \int_{\mathcal{X}} \psi \circ \Pi \, \mathrm{d}\nu,$$

for ψ which maps from $\mathcal{Y}$ to $\mathbb{R}$ and ψ composed with Π is ν-integrable.

Notion of weak convergence The work in [1] and references therein are well-known for being the first to explore this area. If a sequence of measures $\nu_n \in \mathcal{M}\left(\mathbb{R}^d\right)$ is weakly converging to ν, then it means that for all bounded function $\psi : \mathbb{R}^d \to \mathbb{R}$, the equation $\int \psi \mathrm{d}\nu_n \to \int \psi \mathrm{d}\nu$ holds true. This is defined, for $\nu \in \mathcal{M}\left(\mathbb{R}^d\right)$, as

$$\|\nu\|_{\mathrm{BL}} := \sup\left\{\int \psi \mathrm{d}\mu ; \psi : \mathbb{R}^d \to \mathbb{R}, \quad \operatorname{Lip}(\psi) \leq 1, \|\psi\|_{\infty} \leq 1\right\}.$$

Here, $\operatorname{Lip}(\psi)$ refers to its smallest Lipschitz constant.

The **Wasserstein metric** also known as the q-Wasserstein distance measure the difference between two probability measures, ν, and μ, both belonging to $\mathcal{P}\left(\mathbb{R}^d\right)$ and is given by

$$W_q(\nu, \mu) := \left(\min \int |y - z|^q \mathrm{d}\xi(z, y)\right)^{1/q}, \tag{2.4.1}$$

where $\xi \in \mathcal{P}\left(\mathbb{R}^d \times \mathbb{R}^d\right)$. The set of probability measures with finite second moments endowed with the metric W_2 is a complete metric space that we denote $\mathcal{P}_2\left(\mathbb{R}^d\right)$. A sequence $\left(\nu_k\right)_k$ converges in $\mathcal{P}_2\left(\mathbb{R}^d\right)$ if and only if all continuous function $\psi : \mathbb{R}^d \to \mathbb{R}$ with at most quadratic growth it holds $\int \psi \mathrm{d}\nu_k \to \int \psi \mathrm{d}\nu$.

3. Gradient Descent on Wide Neural Networks

This chapter discusses the optimization problems associated with wide neural networks. Linear prediction models present convex problems, with mathematical guarantees readily available. Nonlinear models like neural networks create non-convex optimization problems, which are more challenging to guarantee. In particular, this review focuses on two-layer neural networks with homogeneous activation functions, where the number of hidden neurons tends to infinity. We examine the qualitative guarantees of convergence that can be drawn from this case. Our main sources for this chapter are [5, 7, 13].

3.1 Introduction

Infinitely wide neural networks have recently gained attention in the field of ML due to their unique properties. These networks have an infinite number of neurons in each layer, which allows them to approximate any function with arbitrary accuracy [5, 7]. The use of infinitely wide neural networks has been shown to improve the performance of various machine learning algorithms [5]. However, the theoretical properties and behavior of these networks are not yet fully understood. In this essay, we will analyze the behavior of GD on infinitely wide neural networks and its theoretical and empirical implications.

The theoretical analysis of GD on infinitely wide neural networks has shown that these networks have unique properties that differ from finite neural networks. In particular, the convergence properties of infinitely wide neural networks are different from those of finite neural networks. For example, the convergence rate of infinitely wide neural networks is faster than that of finite neural networks [5]. The theoretical analysis of GD on infinitely wide neural networks has also shown that the network can converge to a global minimum even with random initialization [5].

GD on infinitely wide neural networks has unique theoretical and empirical properties that differ from finite neural networks. Using infinitely wide neural networks has been shown to improve performance on a variety of tasks, but computational complexity can be a limiting factor. Further research is needed to fully understand the behavior of GD in infinitely wide neural networks and to explore their possible application in machine learning.

3.2 Supervised learning with ReLU activation function

Supervised learning methods vary, local averaging techniques such as nearest neighbors or decision trees are effective for low dimensional inputs [4, 12]. On the other hand, empirical risk minimization methods are commonly used in high-dimensional settings. The aim is to minimize training data risk by optimizing prediction functions in a set of pre-defined functions that may be regularized [4].

A related optimization problem is convex [27] if the set of functions is a convex subset of a vector space with a finite-dimensional representation and standard assumptions. Meaning that learning methods have strong theoretical guarantees and can be thoroughly understood in terms of their computational and statistical properties, including running time. This information is supported by studies from [10, 6].

3.2.1 One hidden-layer neural networks with ReLU activation function. The purpose of this section is to explain why supervised machine learning is effective for some models like a hidden-layer neural network (see e.g. Figure 3.1) with a ReLU activation function of the form:

$$\varphi(x,\vartheta)=\frac{1}{m}\vartheta_2^\top\sigma\left(\vartheta_1^\top x\right)=\frac{1}{m}\sum_{i=1}^{m}\vartheta_2(i)\cdot\sigma\left(\vartheta_1(:,i)^\top x\right). \tag{3.2.1}$$

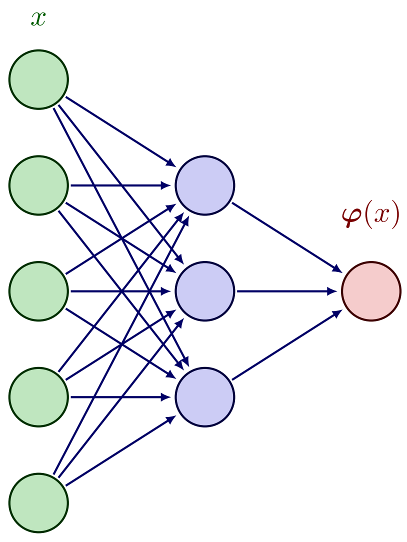


Figure 3.1: One hidden layer neural network of dimension $d=5$, with neuron or width $m=3$, and a single output.

Here in (3.2.1), the vector input x belongs to $\mathbb{R}^d$, and there are m hidden neurons. The output weights are denoted by $\vartheta_2\in\mathbb{R}$, while the input weights are represented by $\vartheta_1\in\mathbb{R}^d, i=1,\ldots,m$. The activation function used is the ReLU defined in Chapter 2, for which results in we develop here heavily rely on its positive homogeneity property.

It is worthwhile noticing that the above deep learning setup is simplified and idealized. Although it only has one hidden layer with no convolutions and pooling, it is still challenging to comprehend, but it highlights some of the main issues related to non-convexity. One may wonder why analyzing it is difficult. Two main challenges contribute to the difficulty:

- **Nonlinearity:** Due to the presence of an activation function, the dependence on the input weights $\vartheta_1(:,i)$' is non-linear, often resulting in non-convex optimization problems..
- **Overparameterization:** When the width, denoted by m, is very high, optimizing and generalizing to new data becomes challenging. The parameters $m(d+1)$ may exceed the number of observations, further complicating the issue.

In this chapter, Our approach will employ overparameterization and tending to infinity(without depending on the number of observations) thereby enabling us to derive theoretical results. This approach will leverage two key properties of the problem:

- **Separability of the model** in $z_i = [\vartheta_2(i), \vartheta_1(\cdot, i)] \in \mathbb{R}^{d+1}$, for a given n pairs of observation $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \ldots, n$ with $\mathcal{Y} = \{-1, 1\}$, the set of weights associated with the hidden neuron i refers to how it is parameterized. The prediction function defined in (3.2.1) is made up of terms that are independently parameterized. The function is calculated as the sum of these terms, where each term corresponds to a different parameter value. The function is defined by

$$\boldsymbol{\varphi}(x; z_1, \cdots, z_m) = \frac{1}{m} \sum_{i=1}^{m} \Phi(z_i), \tag{3.2.2}$$

where $\Phi : \mathbb{R}^p \to \mathcal{F}$, where $p = d + 1$ and $\mathcal{F}$ is a space of functions. The function $\Phi(z)(x)$ which is also called a "feature function" is defined as

$$\Phi(z) : x \mapsto z(1) \cdot \sigma \left[x^\top z(2, \ldots, p) \right] := \vartheta_2 \max \left\{ \vartheta_1^\top x, 0 \right\} \tag{3.2.3}$$

To clarify, the hidden neurons do not share any parameters. However, this only applies to a single hidden layer and not multiple layers. Additionally, the ReLU activation is positively homogeneous, meaning that the function in (3.2.3) is positively 2-homogeneous when considering the variable $z = (\vartheta_2, \vartheta_1) \in \mathbb{R} \times \mathbb{R}^d$. This indicates that $\Phi(\alpha z) = \alpha^2 \Phi(z)$ for $\alpha > 0$.

Recent developments in optimization and machine learning theory have shown that optimization and statistics must be combined. In line with this trend, we focus on gradient flows on expected risks. Our discussion begins with a review of optimization, highlighting how global convergence can be achieved for 2-homogeneous models through over-parameterization [14]. To achieve this, we utilize concepts from optimal transport [38], which we will briefly discuss.

3.3 Infinitely wide limit and probability measures

We are looking at prediction functions that depend on a vector ϑ and are represented by $x \mapsto \boldsymbol{\varphi}(x, \vartheta) \in \mathbb{R}$. Following [5], one of the aims is to minimize with respect to the prediction function $\boldsymbol{\varphi}$ the empirical risk $\mathcal{G}$ defined as

$$\mathcal{G}(\boldsymbol{\varphi}) = \mathbb{E}\left[\ell(y, \boldsymbol{\varphi}(x))\right],$$

which is convex in $\boldsymbol{\varphi}$ for convex loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$. Indeed, this loss function is most often convex in its second arguments, such as for least-squares or logistic loss

$$\ell\left(y_i, \boldsymbol{\varphi}(x_i, \vartheta)\right) = \log\left(1 + \exp(-y_i \boldsymbol{\varphi}(x_i, \vartheta))\right),$$

for a given n pairs of observation $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \ldots, n$ with $\mathcal{Y} = \{-1, 1\}$. Thus, we will assume that $\mathcal{G}$ is convex. In this framework, the expectation can be considered in two scenarios [5, 13, 14]:

(a) **Empirical risk:** This refers to a scenario where we gather data from a joint distribution on $(x, y) \in \mathbb{R}^d \times \mathbb{R}$, denoted by observations (x_i, y_i), $i = 1, \ldots, n$. Simply minimizing $\mathcal{G}$ may not assure unseen data unless we incorporate explicit or implicit regularization.

(b) **Generalization performance:** We take the expectation of unseen data, which means we cannot compute its value or gradient. However, we can use any training observation (x_i, y_i) to get an unbiased estimate. If we use a single-pass stochastic gradient, we can guarantee the expected risk.

We define the set of probability measures on $\mathcal{Z} = \mathbb{R}^p$ as $\mathcal{P}(\mathcal{Z})$. Using this definition, we can rewrite (3.2.2) as

$$\varphi = \int_{\mathbb{R}^p} \Phi(z) \mathrm{d}\mu(z), \quad \text{with} \quad \mu = \frac{1}{m}\sum_{i=1}^{m} \delta_{z_i},$$

where μ is an average of Dirac measures $\delta\left(z_i\right)$ at at each z_i. We refer to each z_i as a particle, following a physics analogy [4]. As the number of particles m increases, the empirical measure μ can converge in distribution to a probability measure with a density, which is commonly referred to as a mean field limit [5]. Therefore, our primary reformulation involves considering an optimization problem over probability measures. The main idea is to consider the minimization of

$$T(Z) = T\left(z_1, \ldots, z_m\right) = \mathcal{G}\left(\frac{1}{m}\sum_{i=1}^{m} \Phi\left(z_i\right)\right) \tag{3.3.1}$$

and see it as the minimization of

$$F(\mu) = \mathcal{G}\left(\int_{\mathbb{R}^p} \Phi(z) \mathrm{d}\mu(z)\right) \tag{3.3.2}$$

with respect to a probability measure μ.

It is worth noticing that if the width is large, we can use expectations of continuous and bounded functions that can be approximated to represent any measure [8]. This makes our problem a convex optimization problem with a convex objective in ν over a convex set. Observed that many fields, including machine learning, and signal processing [9], have already benefited from using the space of all measures instead of discrete measures in various contexts. However, it is still an infinite-dimensional space that requires dedicated finite-dimensional algorithms. In the next section, we focus on GD on z, which corresponds to back-propagation in neural networks [3].

3.3.1 From gradient descent to finite-dimensional gradient flow. Our general goal is to study the GD recursion on $Z = (z_1, \ldots, z_m) \in \mathcal{Z}^m$, defined as

$$Z_{i+1} = Z_i - \gamma m \nabla T(Z_i), \tag{3.3.3}$$

with $T(Z)$ defined in (3.3.1). The back-propagation algorithm is used. To ensure a well-defined limit when m approaches infinity, we add the term m to the step size. As the step-size γ approaches zero, we examine the limit. By linearly interpolating the values of a function $\Psi : \mathbb{R} \to \mathcal{Z}^m$ with values $\Psi(i\gamma) = Z_i$ at $t = \gamma i$, we obtain the Euler discretization of the ordinary differential equation [5, 49].

$$\dot{\Psi} = -m \nabla Z(\Psi). \tag{3.3.4}$$

The main focus of this paper is the GF. It is the limit of the gradient recursion in equation (3.3.3) for vanishing step-sizes γ, especially when additional regularity assumptions are taken into account. Additionally, SGD, which only gives an unbiased version of the gradient, can also lead to the same ordinary differential equation in the limit $\gamma \to 0$ under appropriate conditions. This has been noted in previous studies such as by [5, 11]. There are two relevant questions to consider: **(1)** what is the outcome of the GF dynamics in (3.3.4) when the width m approaches infinity? **(2)** Is it possible to obtain any assurance of global convergence for the resulting dynamics? In the next sections, we will focus on these two questions.

3.3.2 On mean-field limit and wasserstein gradient flow. As m approaches infinity, to use the measure representation, we must analyze the impact of performing the GF on $z_1, \ldots, z_m$ simultaneously on the measure μ. We need to determine if it is possible to take a limit as m approaches infinity. This process, known as the "mean-field" limit, is common in physics and has been explored in recent works such as [14, 36, 55]. To simplify the discussion, let us assume that the map Φ is differentiable enough, although this excludes the ReLU activation as per [14].

In this framework, we aim to minimize a function, denoted by F, that is defined on probability measures, as shown in various works on the infinite width limit of two-layer neural networks [14, 36, 45]. The method utilized in this framework converges to a well-defined mathematical concept known as a Wasserstein GF [1, 42]. This GF is derived from the Wasserstein metric defined in (2.4.1)

$$W_2(\mu, \nu)^2 = \inf_{\xi \in Q(\mu,\nu)} \int \|v - z\|_2^2 \mathrm{d}\xi(v, z),$$

Here, the set $Q(\mu, \nu)$ refers to the collection of probability measures on $\mathcal{Z} \times \mathcal{Z}$ with marginals μ and ν. The GF is defined as the limit when $\gamma \to 0$ of the extension of the following discrete-time dynamics [5]:

$$\mu(t + \xi) = \inf_{\nu \in \mathcal{P}(\mathcal{Z})} F(\nu) + \frac{1}{2\xi} W_2(\mu(t), \nu)^2.$$

In a Euclidean space using the Euclidean metric, applying this definition results in the familiar GF

$$\frac{\mathrm{d}}{\mathrm{d}t}\mu = -\nabla F(\mu).$$

However, with the Wasserstein metric, it defines a distinct flow on the set of measures. If the initial measure is a weighted sum of Diracs, this is equivalent to backpropagation as ξ approaches zero.

In addition, if the Diracs add up and converge to a measure, then the flow will converge to the solution of the PDE. To be more specific, we assume that $\Phi : \mathbb{R}^p \to \mathcal{F}$ is a Hilbert space, and $\nabla \mathcal{F}(\varphi) \in \mathcal{F}$. Next, we will examine the mean potential defined as

$$\mathcal{J}(z|\mu) = \left\langle \Phi(z), \nabla \mathcal{G}\left(\int_{\mathcal{Z}} \Psi(v)\mathrm{d}\mu(v)\right)\right\rangle, \tag{3.3.5}$$

which solves the classical continuity equation

$$\partial_t \mu_t(z) = \mathrm{div}(\mu_t(z)\nabla \mathcal{J}(z|\mu_t)). \tag{3.3.6}$$

The concept of distribution is involved here. To clarify this behavior, a result has been formalized in a study by Chizat and Bach in 2018 [14], which we will now present as a theorem

3.3.3 Theorem ([5]). *Assume that $\mathcal{G} : \mathcal{F} \to [0, +\infty)$ and $\Psi : \mathcal{Z} = \mathbb{R}^p \to \mathcal{F}$ is Lipschitz differential. Additionally, assume that $\mathcal{G}$ is Lipschitz on its sublevel sets. Let $(z_i(0))_{i \geq 1}$ be a sequence of initial weights contained in a compact subset of $\mathcal{Z}$, and let $\mu_{t,m} := \frac{1}{m}\sum_{i=1}^m z_i(t)$ where $(z_1(t), \ldots, z_m(t))$ solves the equation (3.3.4). If $\mu_{0,m}$ weakly converges to some $\mu_0 \in \mathcal{P}(\mathcal{Z})$, then $\mu_{t,m}$ weakly converges to μ_t. Here, $(\mu_t)_{t \geq 0}$ is the unique weakly continuous solution to (3.3.6) initialized with μ_0.*

In the next section, we will look at the solution to the PDE called the Wasserstein GF. This is seen as the gradient flow's limit in equation (3.3.4) when the number of neurons m gets closer to infinity.

3.4 On the global convergence of the gradient flow

The main finding of [5] can be summarized as follows: If the assumptions described below hold and the Wasserstein GF converges to a measure, that measure must be the global minimum of the function F defined above. In addition to technical regulations, there are two important broad assumptions that we need to consider:

- To ensure that the Wasserstein GF converges to a global minimizer, we require a condition for the homogeneity of the function $\Phi : \mathbb{R}^p \to \mathcal{F}$. This is because if $\mathcal{G}$ is a linear map, thus $F(\mu)$ becomes $F(\mu) = \int_{\mathbb{R}^p} T(z)\mathrm{d}\mu(z)$ with $T(z) = \mathcal{G}(\Phi(z))$. However, the weighted sum of Diracs at local minimizers of T may not be a global minimizer.
- To begin, there is a positive mass distributed evenly in all directions. This means that the z_i values are Gaussian.

To obtain an accurate outcome, we will convert the flow on probability measures in $\mathcal{W} = \mathbb{R}^p$ into a flow on measures on the unit sphere, denoted by $\mathcal{S}^d = \{z \in \mathbb{R}^p,\ \|z\|_2 = 1\}$. This conversion is feasible if the function Ψ is positively 2-homogeneous on $\mathcal{W} = \mathbb{R}^p$, which implies that $\Psi(kz) = k^2\Psi(z)$ for $k > 0$. By parameterizing each particle z_i in polar coordinates, i.e., $z_i = \lambda_i\psi_i$ with $\lambda_i \in \mathbb{R}$ and $\psi_i \in \mathcal{S}^d$, we can exploit homogeneity.

Using homogenetity, we have a prediction function:

$$\varphi = \frac{1}{m}\sum_{i=1}^{m}\Psi(z_i) = \frac{1}{m}\sum_{i=1}^{m}\lambda_i^2\Psi(\psi_i).$$

Moreover the function $\mathcal{J}$ defined in (3.3.5) is also 2-homogeneous, and its gradient then 1-homogeneous. The flow from (3.3.4), can be written

$$\dot{z}_i = -\nabla\mathcal{J}(z_i|\mu) \quad \text{with} \quad \mu = \frac{1}{m}\sum_{i=1}^{m}\delta_{z_i}.$$

After some computations we can shows that the flow

$$\left\{\begin{array}{ll} \dot{\lambda}_i & = -2\lambda_i\mathcal{J}(\psi_i|\varrho) \\ \dot{\psi}_i & = -(I - \psi_i\psi_i^\top)\nabla\mathcal{J}(\psi_i|\varrho) \end{array}\right. \quad \text{with} \quad \varrho = \frac{1}{m}\sum_{i=1}^{m}\lambda_i^2\delta_{\psi_i}, \tag{3.4.1}$$

leads to exactly the same dynamics. Indeed, by homogeneity of Ψ, the two definitions of μ and ϱ (through the z_i's, or the ψ_i's and λ_i's) lead to the same functions $\mathcal{J}(\cdot|\mu)$ and $\mathcal{J}(\cdot|\varrho)$, and we get

$$\begin{aligned} \dot{z}_i &= \dot{\lambda}_i\psi_i + \lambda_i\dot{\psi}_i = -2r_i\mathcal{J}(\psi_i|\varrho)\psi_i - \lambda_i(I - \psi_i\psi_i^\top)\nabla\mathcal{J}(\psi_i|\varrho) \\ &= -\lambda_i\nabla\mathcal{J}(\psi_i|\varrho) - \lambda_i\left(2\mathcal{J}(\psi_i|\varrho) - \psi_i^\top\nabla\mathcal{J}(\psi_i|\varrho)\right)\psi_i \\ &= -\nabla\mathcal{J}(z_i|\mu), \end{aligned}$$

because $z \mapsto \nabla\mathcal{J}(z|\varrho)$ is 1-homogeneous and by the Euler identity for the 2-homogeneous function $z \mapsto \mathcal{J}(z|\varrho) = \mathcal{J}(z|\mu)$. Moreover, the flow defined in (3.4.1) is such that ψ_i remains on the sphere $\mathcal{S}^d$. We will study this flow under the assumption that the function Ψ is sufficiently regular. The ReLU neural networks are an exception to this, as they simplify the proof [13]. We first derive a PDE analogous to (3.3.6). We consider a smooth test function $\chi : \mathcal{S}^d \to \mathbb{R}$, and we set

$$\theta = \int_{\mathcal{S}^d} \chi(\psi) \mathrm{d}\varrho(\psi) = \frac{1}{m} \sum_{i=1}^{m} \lambda_i^2 \chi(\psi_i).$$

We have

$$\begin{aligned}\dot{\theta} &= \frac{1}{m} \sum_{i=1}^{m} 2\lambda_i \dot{\lambda}_i \chi(\psi_i) + \frac{1}{m} \sum_{i=1}^{m} \lambda_i^2 \nabla\chi(\psi_i)^\top \dot{\psi}_i \\ &= -\frac{1}{m} \sum_{i=1}^{m} 4\lambda_i^2 \mathcal{J}(\psi_i|\varrho)\chi(\psi_i) - \frac{1}{m} \sum_{i=1}^{m} \lambda_i^2 \nabla\chi(\psi_i)^\top (I - \psi_i\psi_i^\top)\nabla\mathcal{J}(\psi_i|\varrho) \\ &= -4 \int_{\mathcal{S}^d} \chi(\psi)\mathcal{J}(\psi|\varrho)\mathrm{d}\varrho(\psi) - \int_{\mathcal{S}^d} \nabla\chi(\psi|\varrho)^\top (I - \psi\psi^\top)\nabla\mathcal{J}(\psi|\varrho)\mathrm{d}\varrho(\psi).\end{aligned} \tag{3.4.2}$$

This demonstrates that we possess the PDE for the density ϱ_t during a given time frame t.

$$\partial_t \varrho_t(\psi) = -4\mathcal{J}(\psi|\varrho_t) + \mathrm{div}(\varrho_t(\psi)\nabla\mathcal{J}(\psi|\varrho_t)) \tag{3.4.3}$$

satisfied in the sense of distributions. We have the following result.

3.4.1 Theorem ([5]). *Assume the function $\Psi : \mathcal{S}^d \to \mathcal{F}$ is continuously differentiable d times and ϱ_0 is a nonnegative measure on the sphere $\mathcal{S}^d$ with a finite mass and full support, then the flow defined in (3.4.3) is well defined for all values of $t \geqslant 0$. Foremost, if ϱ_t converges weakly to some limit ϱ_∞, then ϱ_∞ is a global minimum of the function $\varrho \mapsto F(\varrho) = \mathcal{G}\left(\int_{\mathcal{S}^d} \Psi(\psi)\mathrm{d}\varrho(\psi)\right)$ over the set of nonnegative measures.*

Proof. To minimize the convex functional F, the global optimality conditions require that $\mathcal{J}(\psi|\varrho_\infty) = 0$ on the support of ϱ_∞, and $\mathcal{J}(\psi|\varrho_\infty) \geqslant 0$ on the entire sphere. The proof for this, which has been adapted from [13], is as follows:

Step 1: The flow $(\varrho_t)_{t\geq 0}$ can be proven to exist and be unique by using the equivalence with a Wasserstein GF $(\mu_t)_{t\geq 0}$ in $\mathcal{P}(\mathbb{R}^p)$, as well as the theory of Wasserstein [1]. $(\varrho_t)_{t\geq 0}$ is a GF for a metric between nonnegative measures. This metric is the inf-convolution of the Wasserstein and the Hellinger metric, as discussed in [13].

Step 2: The flow ϱ_t is always fully supported at all times t. This conclusion can be drawn from the way the solutions to (3.4.3) are represented as:

$$\varrho_t = X(t, \cdot)_\# \left(\varrho_0 \exp\left(-4 \int_0^t \mathcal{J}(X(\xi, \cdot)|\varrho_\xi)\mathrm{d}\xi\right)\right),$$

The function $X : [0, +\infty) \times \mathcal{S}^d \to \mathcal{S}^d$ maps points in space over time, and is connected to a vector field that changes over time. This vector field is represented by the negative gradient of $\mathcal{J}$, i.e. $-\nabla\mathcal{J}(\cdot \mid \varrho_t)$ with respect to its input, conditioned on a variable ϱ_t for all $\psi \in \mathcal{S}^d$. Based on our regularity assumptions, the stability results for ordinary differential equations ensure that at any point in time t, $X(t, \cdot)$ is a diffeomorphism of the sphere. This means that ϱ_t is the image measure, and it is a result of a diffeomorphism of a measure with the form $\varrho_0 \exp(\dots)$ that has full support. Thus, ϱ_t also has full support.

Step 3: We assume that the flow will eventually reach a certain measure called ϱ_∞, which may not be a regular measure. The equation (3.4.1) shows that $\mathcal{J}(\psi|\varrho_\infty)$ will be equal to zero on the range of ϱ_∞ due to its stability. However, we cannot assume anything outside this range of ϱ_∞. Additionally, we require that $\mathcal{J}(\psi|\varrho_\infty)$ should always be non-negative for all $\psi \in \mathcal{S}^d$.

To prove that $\min_{\psi\in\mathcal{S}^d} \mathcal{J}(\psi|\varrho_\infty)$ is positive, we will assume that it is actually negative and then show that this assumption leads to a contradiction. To begin, we need to find a value of v that meets two conditions. Firstly, it must be less than zero, and secondly, it must be greater than the minimum value of $\mathcal{J}(\psi|\varrho_\infty)$ for all ψ in $\mathcal{S}^d$. Additionally, we need to ensure that the gradient $\nabla\mathcal{J}(\psi|\varrho_\infty)$ does not disappear on the v-level-set $\{\psi \in \mathcal{S}^d, \mathcal{J}(\psi|\varrho_\infty) = v\}$ of $\mathcal{J}(\cdot|\varrho_\infty)$. We know that such a value of v exists due to the Morse-Sard lemma, which applies because our assumptions ensure that $\mathcal{J}(\cdot|\varrho)$ is continuously differentiable for any finite nonnegative measure ϱ.

We then consider the set $\mathcal{E} = \left\{\psi \in \mathcal{S}^d,\ \mathcal{J}(\psi|\varrho_\infty) \leqslant v\right\}$, which has some boundary $\partial\mathcal{E}$, such that the $\nabla\mathcal{J}(\psi|\varrho_\infty)$ has strictly positive dot-product with an outward normal vector to the level set at $\psi \in \partial\mathcal{E}$.

There is a value, ϱ_∞, to which ϱ_t weakly converges. After a certain point, $t_\star > 0$, for any $t \geqslant t_\star$, $\sup_{\psi\in\mathcal{E}} \mathcal{J}(\psi|\varrho_t) < v/2$, while on the boundary, the dot-product of $\nabla\mathcal{J}(\psi|\varrho_\infty)$ and an outward normal vector is non-negative. This implies that if θ_t is the value of $\varrho_t(\mathcal{E})$ and we apply (3.4.2) to the indicator function of $\mathcal{E}$, then for all t greater than $t_\star$, the conditions are satisfied.

$$\theta'(t) \geqslant -4 \sup_{\psi\in\mathcal{E}} \mathcal{J}(\psi|\varrho_t)\theta(t).$$

Based on the preceding statement, if $\theta(t_\star) > 0$ is greater than zero, then according to the generalized Grönwall's lemma, $\theta(t)$ will diverge. This contradicts the convergence of ϱ_t to ϱ_∞. □

3.5 Numerical experiment

We demonstrate our findings of the global convergence result by considering a supervised learning problem on $\mathbb{R}^2$, with a Gaussian input data x, and output data y given as:

$$y = \sum_{i=1}^{m_\star} \vartheta_2(i) \cdot \max\{x^\top \vartheta_1(:, i), 0\} \tag{3.5.1}$$

We are given a "teacher" neural network with finite weight values represented by ϑ_1 and ϑ_2, as well as a finite value for $m_\star$. Our goal is to evaluate the expected square loss of $\mathcal{G}(\varphi)$ using SGD and new samples (x_i, y_i) with a small step size. We will test different values of hidden neurons, denoted by m, to determine when the original neurons can be retrieved.

The diagram in Figure 3.2 represents a neural network comprising of $m_\star = 9$ neurons that produce output values of y, as defined in equation (3.5.1), from the input value, x. Upon applying the GF method, the model attains zero loss when m is greater than or equal to 10, which is the optimal outcome. However, the GF may become stuck at a local minimum, resulting in failure to reach the global minimum. Our theory suggests that increasing the number of neurons to a larger value can help us reach the original neurons, as demonstrated below. Surprisingly, we have discovered that we do not need to increase m significantly beyond $m_\star$ to achieve global convergence in practical scenarios, although the reason for this remains unexplained.

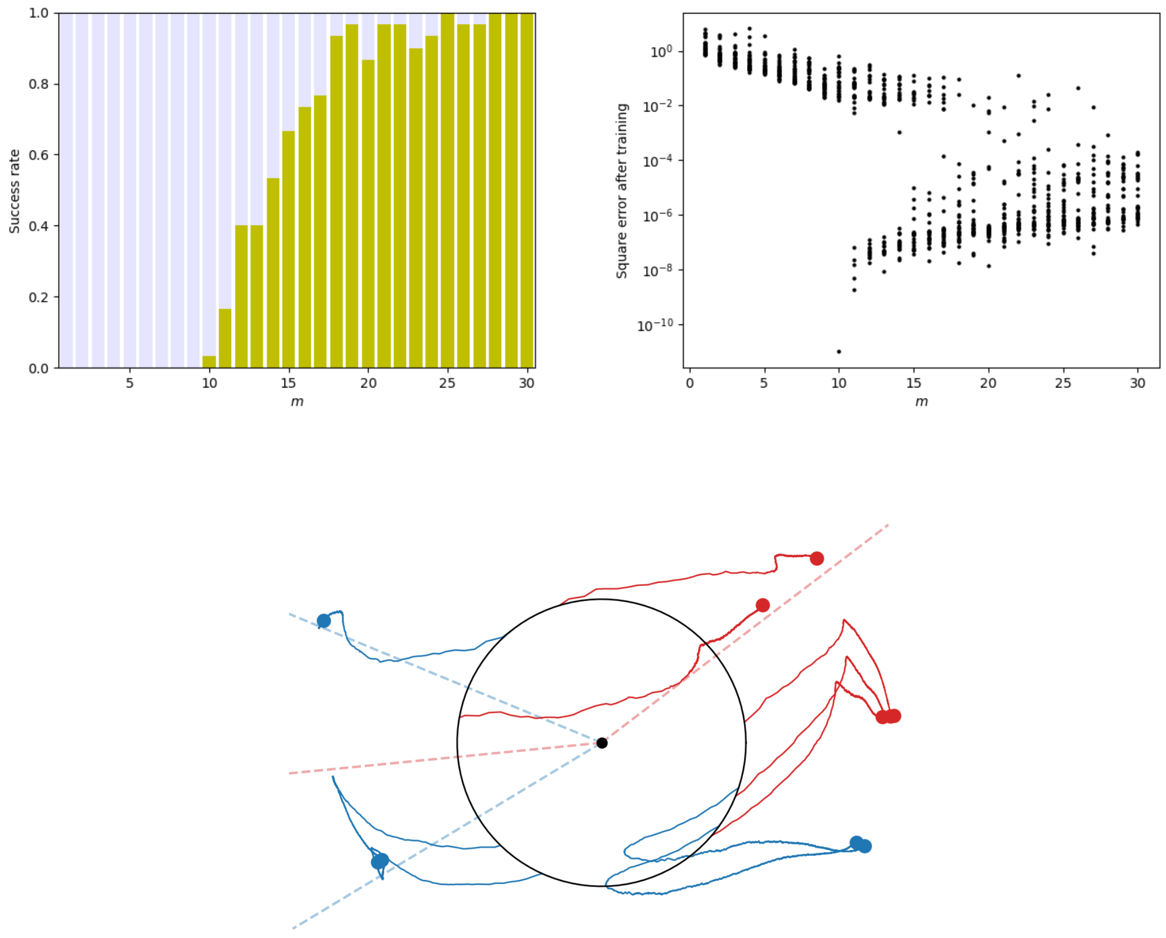


Figure 3.2: This is a visual representation of a GF on a two-layer neural network with $m = 10$, using the ReLU activation function. The position of the particles on the graph is determined by multiplying the absolute value of the second layer's values (represented by $\vartheta_2(i)$) with the first layer's values (represented by $\vartheta_1(\cdot, i)$) as $|\vartheta_2(i)| \cdot \vartheta_1(\cdot, i)$. The color of each particle depends on whether $\vartheta_2(i)$ is positive or negative. The dashed lines show the neurons responsible for generating the data distribution, with a total of $m_\star = 9$. In order to improve the clarity of the information presented, a black unit circle is used to initialize particles, and the radial axis is scaled using $\tanh$. Two images are displayed above, depicting an SGD on the square loss in the "teacher-student" scenario, with 10^4 iterations, a batch size of 100, a learning rate of 0.005, and $d = 100$. The image on the right shows the risk (expected square loss) after training as a function of m over 30 random repetitions, while the image on the left shows the success rate as a function of m over 30 repetitions.

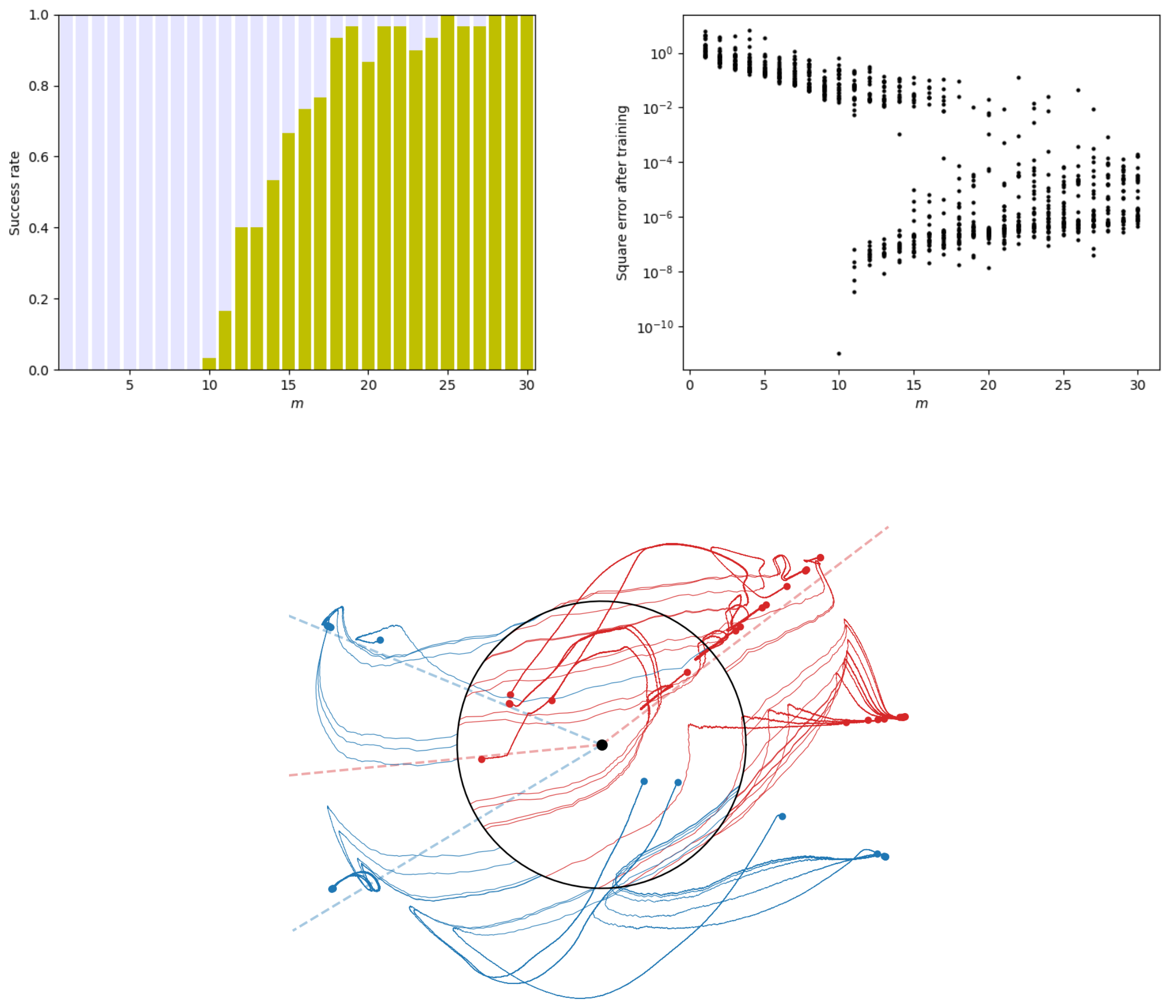
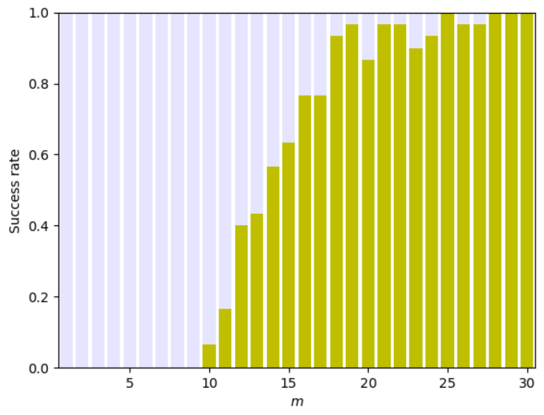
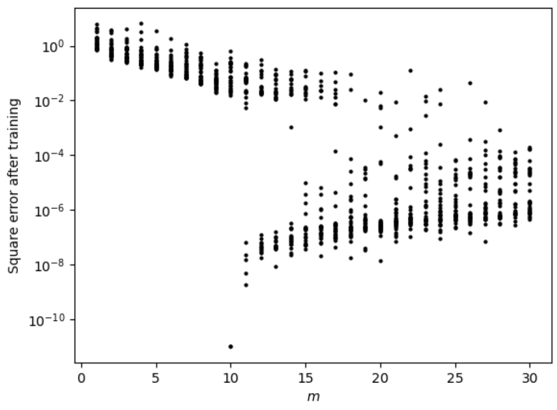


Figure 3.3: Visual representation of a GF on a two-layer neural network with $m = 50$, using the ReLU activation function. The position of the particles on the graph is determined by multiplying the absolute value of the second layer's values (represented by $\vartheta_2(i)$) with the first layer's values (represented by $\vartheta_1(\cdot, i)$) as $|\vartheta_2(i)| \cdot \vartheta_1(\cdot, i)$. The color of each particle depends on whether $\vartheta_2(i)$ is positive or negative. The dashed lines show the neurons responsible for generating the data distribution, with a total of $m_\star = 40$. In order to improve the clarity of the information presented, a black unit circle is used to initialize particles, and the radial axis is scaled using $\tanh$. Two images are displayed above, depicting an SGD on the square loss in the "teacher-student" scenario, with 10^4 iterations, a batch size of 100, a learning rate of 0.005, and $d = 100$. The image on the right shows the risk (expected square loss) after training as a function of m over 30 random repetitions, while the image on the left shows the success rate as a function of m over 30 repetitions.





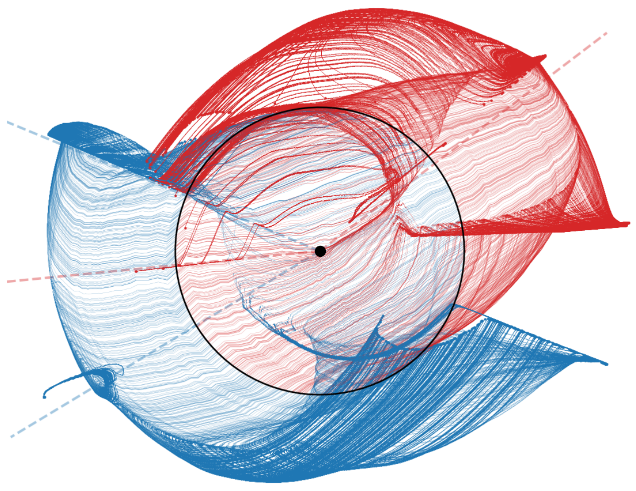


Figure 3.4: Visual representation of a GF on a two-layer neural network with $m = 2000$, using the ReLU activation function. The position of the particles on the graph is determined by multiplying the absolute value of the second layer's values (represented by $\vartheta_2(i)$) with the first layer's values (represented by $\vartheta_1(\cdot, i)$) as $|\vartheta_2(i)| \cdot \vartheta_1(\cdot, i)$. The color of each particle depends on whether $\vartheta_2(i)$ is positive or negative. The dashed lines show the neurons responsible for generating the data distribution, with a total of $m_\star = 1900$. In order to improve the clarity of the information presented, a black unit circle is used to initialize particles, and the radial axis is scaled using $\tanh$. Two images are displayed above, depicting an SGD on the square loss in the "teacher-student" scenario, with 10^4 iterations, a batch size of 100, a learning rate of 0.005, and $d = 100$. The image on the right shows the risk (expected square loss) after training as a function of m over 30 random repetitions, while the image on the left shows the success rate as a function of m over 30 repetitions.

4. Federated Learning with Finite Wide Neural Networks

Federated learning refers to a machine learning scenario wherein several entities (clients) work together to address a machine learning problem, with a central server or service provider overseeing the coordination. client data is stored locally and not exchanged or transferred. Instead, focused updates are utilized for immediate aggregation, with the goal of achieving the learning objective without compromising the privacy of the raw data. [30, 23].This is done by collaboratively training a neural network on a server and having each user send parameter updates based on their local data [23]. McMahan's algorithm is commonly used for this process and involves using algorithms such as Federated SGD and FedAvg to optimize the objective function. Federated SGD does not update the weights on the client, instead obtaining the weights at the server, while FedAvg uses a combination of Federated SGD and mini-batches to update the models directly on the client [50]. In both cases, the server obtains the parameters of the global model by averaging the parameters collected from each client [50]. However, sending parameter gradients in federated learning may not be secure, as it is possible to reconstruct images at high resolution from the knowledge of their parameter gradients [23]. To tackle these challenges, researchers are exploring methods to minimize data collection, enhance the security of collected data, improve the resilience to network instability, and decrease the number of training parameters in neural networks. [35, 39, 50]. Continuous optimization is an effective method for dealing with the dynamic and heterogeneous nature of federated learning, and can be utilized to achieve an optimal solution in a distributed environment. [35, 39, 50]. By applying gradient flow to federated learning, a new ANN model can be proposed to enhance neural network efficiency, which is in line with learning through many repetitions of machine learning [35, 39, 50]. Finally, the proposed algorithm aggregates the trained model on the server by sharing the score value, with the client with the best score providing the trained model to the server as the final result [35, 39, 50].

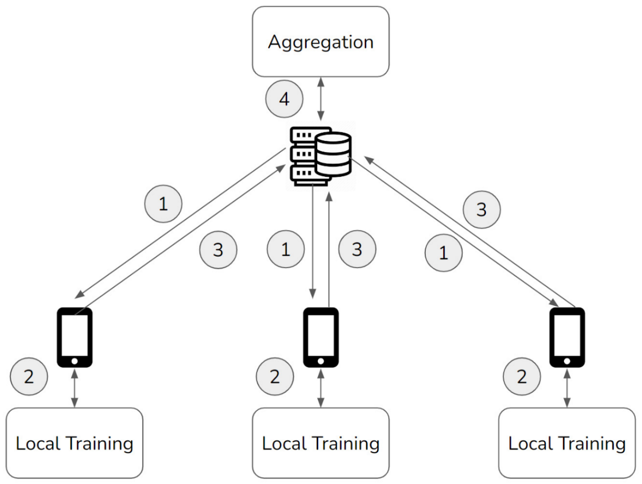


Figure 4.1: This figure borrowed from [52] provides an overview of how federated learning works.

In this chapter, we will briefly cover the theoretical analysis of the convergence of current federated learning algorithms in the classical scenario. We will start by presenting some characteristics and restrictions of federated learning in Section 4.0.1. Next, in Section 4.1 we shall introduce some theoretical tools of Federated Learning, and for the convergence analysis in Section 4.2 by offering a simple proof for the FedAvg. We will then discuss the impact of local updates and data heterogeneity and compare them to well-known baselines in optimization theory. Finally, in Section 4.3, we will provide some numerical results.

4.0.1 Characteristics and restrictions of federated learning. As a relatively new term, federated learning has been defined differently in various existing literature, which can lead to misinterpretation of terms and concepts. To ensure consistency and clarity in this report, this section presents a set of characteristics and limitations that define the federated learning setting in the context of this study. These characteristics are based on the definitions in [**?**] and [32].

Data locality. Involves the generation of data locally by clients, with the data remaining decentralized and never shared with other clients or the central server. Moreover, the non-IID nature of the data across clients means that it is not independently or identically distributed.

Central orchestration. Involves clients communicating solely with a central orchestration server, which coordinates the training process. However, the server does not have access to the raw data, ensuring that the privacy of the data is maintained.

Communication. Due to the potentially large physical distance between clients in federated learning, communication can be costly and subject to volatility. Hence, all advanced methods and techniques for federated learning aim to minimize the communication between clients and the server.

Privacy guarantees While focused updates in federated learning represent a step towards protecting the data generated at each client, they do not provide complete protection against the potential revelation of sensitive information, such as gradient information[22, 43]. To ensure that sensitive information is protected in federated learning, all developed methods and techniques should be designed to be compatible with privacy-enhancing technologies such as differential privacy. [19].

Systems heterogeneity. it is crucial to take into account the varying data storage technologies and computing resources of each client's system when working with multiple clients. This is because differences in data schemes and processing times can lead to significant disparities in system performance. To ensure optimal performance, any developed methods or techniques for federated learning must account for these variations.

4.1 Federated optimization theory

As seen in Chapter 2 and in Chapter 3, in Machine Learning, the goal is to find a model for the training data that minimizes a loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$ that defines how our learned model distribution differs from the empirical distribution. For example images ($\mathcal{X}$ is then the set of all possible images), with a set of labels ($\mathcal{Y}$ is then a finite set, which we will assume to be a subset of $\mathbb{R}$ for simplicity).

In this framework, we are given n pairs of observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \ldots, n$, and we consider prediction functions that are parameterized by a vector θ and denoted as $x \mapsto \varphi(x, \theta) \in \mathbb{R}$. We estimate the value of θ by minimizing the regularized empirical risk. This is done by solving the minimization problem

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, \boldsymbol{\varphi}(x_i, \theta)\right). \tag{4.1.1}$$

In the setup of Federated Learning the characteristics of data distribution from which our training examples (x_i, y_i) will be drawn, are the following:

(a) *Massively Distributed*. Data points are distributed across a large number of K nodes, where the number of nodes can be significantly greater than the average number of training examples stored on each node (n/K).

(b) NON-IID. Data available on each node may be drawn from a distinct distribution, which implies that the locally available data points may not be an accurate representation of the overall distribution.

(c) *Unbalanced*. Different nodes may vary by order of magnitude in the number of training examples they hold.

If we adapt the objective function (4.1.1) to these characteristics, our problem can be defined as introduced in the following: we have K nodes and n data points, a set of indices $\mathcal{V}_j$ $(j \in \{1, \ldots, K\})$ of data stored at node j, and $n_j = |\mathcal{V}_j|$ is the number of data points at $\mathcal{V}_j$. We assume that $\mathcal{V}_j \cap \mathcal{V}_k = \emptyset$ whenever $k \neq K$, thus $\sum_{j=1}^{K} n_j = n$. We can then define the local loss for the node as

$$F_j(v) \overset{\text{def}}{=} \frac{1}{n_j} \sum_{i \in \mathcal{V}_K} \ell\left(y_i, \boldsymbol{\varphi}(x_i, v)\right).$$

Thus the problem to be minimized will become:

$$\min_{\theta \in \mathbb{R}^d} \ \ell\left(y_i, \boldsymbol{\varphi}(x_i, \theta)\right) = \min_{\theta \in \mathbb{R}^d} \sum_{j=1}^{K} \frac{n_j}{n} F_j(\theta). \tag{4.1.2}$$

To solve the problem in (4.1.2) the simplest algorithm is FEDERATED SGD introduced in [35, 39], which is equivalent to minibatch gradient descent overall data, and it is a simple application of distributed SGD for the described setup.

Algorithm 1: FEDERATED SGD

```
 1: Initialize global model parameters $\theta$
 2: Broadcast $\theta$ to all devices **for** $t = 1, 2, 3, \ldots$ **do**
 3:
        **end for**
        Randomly select a subset of devices $S_t$ with size $m$ **for** *each device* $j \in S_t$ **do**
 4:
        **end for**
        Receive current global model parameters $\theta_t$ from server
 5: Sample a mini-batch of size $B$ from the device's local data: $(x_i, y_i)i = 1^B$
 6: Calculate the gradient on the mini-batch: $\mathcal{H}j, t \leftarrow \nabla \frac{1}{B} \sum_{i=1}^{B} \ell(x_i, y_i, \theta_t)$
 7: Update local model parameters: $\theta_{j,t+1} \leftarrow \theta_t - \eta \mathcal{H}_{j,t}$
 8: Send updated local model parameters $\theta_{j,t+1}$ to server
 9:
10: Aggregate the gradients of the devices in $S_t$: $\mathcal{H}_t \leftarrow \frac{1}{m} \sum_{j \in S_t} \mathcal{H}_{j,t}$
11: Global model update: $\theta_{t+1} \leftarrow \theta_t - \eta \mathcal{H}_t$
12: Broadcast updated global model parameters $\theta_{t+1}$ to all devices
13:
```

When optimizing a neural network (NN), the loss functions often have a non-convex shape. As a result, the most common methods for optimizing network parameters involve using gradient-based techniques, specifically the various versions of stochastic gradient descent (SGD). These methods involve taking the derivatives of the loss function with respect to the model's parameters and then adjusting the parameter values accordingly in the direction opposite to the gradient.

From the pure form of SGD samples (see [35, 39] and references therein) a random function (e.g a random training data point) $i(j) \in \{1, \ldots, n\}$ taken uniformly at random, the number of iterations is at most $\frac{R^2}{\lambda} \frac{1}{\varepsilon}$ and performs the update:

$$\theta_{j+1} = \theta_j - \gamma \nabla \ell\big(y_{i(j-1)}, \varphi(x_{i(j-1)}, \theta_j)\big), \tag{4.1.3}$$

where γ denotes the learning rate, which is, in the base case, decaying during the learning to enforce convergence.

4.2 Convergence analysis for federated optimization algorithms

We present a demonstration for the vanilla FEDAVG [35, 39], also referred to as Local SGD or parallel SGD for homogeneous data in the literature. This serves as an example of how theoretical results can be achieved in federated optimization, highlighting the structure, tools, and techniques employed. Initially, we examine a simplistic scenario of the generalized FEDAVG framework (Algorithm 2), and then we provide references to the literature for variations in different situations. The approaches and insights discussed in this section can be found in numerous prior works, including [48, 53, 57, 56].

4.2.1 Assumptions and Preliminaries. For our analysis, we have made several assumptions. The first four assumptions (a-d) pertain to the algorithmic choices in FEDAVG, while the fifth and sixth (e and f) pertain to the characteristics of local objectives. The seventh assumption (g) pertains to the similarity or heterogeneity of data across clients.

(a) In each round, every client performs a total of τ local SGD steps with a constant learning rate η, namely $z_i^{(\tau,j+1)} = z_i^{(\tau,j)} - \gamma\mathcal{H}_i(z_i^{(\tau,j)})$, where $j \in [0,\tau)$.

(b) The ServerOpt takes a unit descent step with server learning rate 1.0, namely $z^{(\tau+1)} = z^{(\tau)} + \Delta^{(\tau)}$.

(c) There are a finite number of clients (K), and each client contributes equally to the global objective. Each client is identified by a unique label ranging from $\{1,2,\ldots,K\}$. Formally, we have $\mathcal{V}(z) = \frac{1}{K}\sum_{i=1}^{K}\mathcal{V}_i(z)$.

(d) Each client participates *every round*, namely $S^{(\tau)} \equiv \{1,2,\ldots,K\}$.

(e) Each local objective $\mathcal{V}_i(z)$ is *convex* and *L-smooth*.

(f) Each client can request an unbiased stochastic gradient with a uniformly bounded variance of σ^2 in the ℓ_2 norm. The expected value of the stochastic gradient is equal to the gradient of the client's objective function, and the variance of the difference between the stochastic gradient and the objective function gradient is uniformly bounded by σ^2.

$$\mathbb{E}[\mathcal{H}_i(z_i^{(\tau,j)})|z_i^{(\tau,j)}] = \nabla\mathcal{V}_i(z_i^{(\tau,j)}), \quad \mathbb{E}[\|\mathcal{H}_i(z_i^{(\tau,j)}) - \nabla\mathcal{V}_i(z_i^{(\tau,j)})\|^2|z_i^{(t,j)}] \leq \sigma^2. \tag{4.2.1}$$

(g) The difference of local gradient $\nabla\mathcal{V}_i(z)$ and the global gradient $\nabla\mathcal{V}(z)$ is ζ-uniformly bounded in ℓ_2 norm, namely

$$\max_i \sup_z \left\|\nabla\mathcal{V}_i(z_i^{(\tau,j)}) - \nabla\mathcal{V}(z_i^{(\tau,j)})\right\| \leq \zeta. \tag{4.2.2}$$

To prove convergence, we aim to demonstrate that our iterates, denoted as $z^{(\tau,j)}$, are moving closer to the optimal value $\mathcal{V}(z^\star)$ of the objective function F. In the context of federated learning, we have multiple local iterates on clients and it is important to ensure that all of them are approaching the minimizer. To address this, we use the concept of a shadow sequence, which is commonly used in decentralized optimization literature (as described in [33]). The shadow sequence is defined as $\overline{z}^{(\tau,j)} := \frac{1}{K}\sum_{i=1}^{K} z_i^{(\tau,j)}$, where K is the number of clients. Using this notation, we can analyze the convergence of the federated learning process.

$$\overline{z}^{(\tau,j+1)} = \overline{z}^{(\tau,j)} - \frac{\eta}{K}\sum_{i=1}^{K} g_i(z_i^{(\tau,j)}). \tag{4.2.3}$$

In equation (4.2.3), it can be seen that the averaged iterates use a perturbed stochastic gradient descent approach. The gradients are computed using the client models $z_i^{(\tau,j)}$ instead of $\overline{z}^{(\tau,j)}$. Our goal is to demonstrate that:

$$\mathbb{E}\left[\frac{1}{\kappa T}\sum_{\tau=0}^{T-1}\sum_{j=1}^{T}\mathcal{V}(\overline{z}^{(\tau,j)}) - \mathcal{V}(z^\star)\right] \leq \text{an upper bound decreasing with } T. \tag{4.2.4}$$

How fast the above quantity decreases with T is called the *rate of convergence*. It is also worth noting that at the end of each round, we have $\overline{z}^{(\tau,\tau)} = \overline{z}^{(\tau+1,0)} = z^{(\tau+1)}$. So the above quantity (4.2.4) also quantifies the convergence of the global model. The speed at which the quantity mentioned above decreases with respect to T is known as the *rate of convergence*. It is important to note that at the end of each round, the values of $\overline{z}^{(\tau,\tau)} = \overline{z}^{(\tau+1,0)} = z^{(\tau+1)}$. This means that the quantity described in (4.2.4) can also be used to measure the convergence of the global model.

4.2.2 Main Results. We introduce two lemmas to facilitate the proof.

1. To demonstrate progress in each round, we aim to keep $\mathbb{E}\left[\frac{1}{\tau}\sum_{j=1}^{\kappa}\mathcal{V}(\overline{z}^{(\tau,j)}) - \mathcal{V}(z^{\star})\right]$ within the bounds of a potential function evaluated at both τ and $\tau+1$. This function measures the optimization's progress, with an added small error term.
2. Demonstrate that the client iterations will stay close to the global average/shadow sequence. This means that the expectation of $\|z_i^{(\tau,j)} - \overline{z}^{(\tau,j)}\|^2$ will remain within a certain limit.

The initial lemma follows a familiar approach to analyzing centralized optimization algorithms, but now we must also guarantee that no client is straying from the expected path. With these two lemmas, we can easily demonstrate significant advancement towards achieving the optimal value by telescoping over τ throughout T rounds. Below, we present the two lemmas formally and provide a brief outline of their proofs.

4.2.3 Lemma. Let us assume that the client's learning rate satisfies $\eta \leq \frac{1}{4L}$, then

$$\mathbb{E}\left[\frac{1}{\kappa}\sum_{j=1}^{\kappa}\mathcal{V}(\overline{z}^{(\tau,j)}) - \mathcal{V}(z^{\star})\middle|\mathcal{F}^{(\tau,0)}\right] \leq \frac{1}{2\eta\kappa}\left(\left\|\overline{z}^{(\tau,0)} - z^{\star}\right\|^2 - \mathbb{E}\left[\left\|\overline{z}^{(\tau,\kappa)} - z^{\star}\right\|^2\middle|\mathcal{F}^{(\tau,0)}\right]\right) + \frac{\eta\sigma^2}{K} + \frac{L}{K\kappa}\sum_{i=1}^{K}\sum_{j=0}^{\kappa-1}\mathbb{E}\left[\left\|z_i^{(\tau,j)} - \overline{z}^{(\tau,j)}\right\|^2\middle|\mathcal{F}^{(\tau,0)}\right],$$

The symbol $\mathcal{F}^{(\tau,0)}$ refers to the collection of all information from previous rounds up until the start of the τ-th round, known as the σ-field.

4.2.4 Lemma (Boundedness of client drift)**.** Assume that the client learning rate satisfies $\gamma \leq \frac{1}{4L}$, then we have

$$\mathbb{E}\left[\left\|z_i^{(\tau,j)} - \overline{z}^{(\tau,j)}\right\|^2\middle|\mathcal{F}^{(\tau,0)}\right] \leq 18\kappa^2\gamma^2\zeta^2 + 4\kappa\gamma^2\sigma^2,$$

where $\mathcal{F}^{(\tau,0)}$ is the σ-field representing all the historical information up to the start of the τ-th round.

Now, we are in a position to state our main theorem: Combine Lemma 4.2.3 and Lemma 4.2.4 and telescope τ from 0 to $T-1$ to obtain the main theorem as follows.

4.2.5 Theorem (Convergence rate)**.** *Under the aforementioned assumptions (a)-(g), if the client learning rate satisfies $\gamma \leq \frac{1}{4L}$, then one has*

$$\mathbb{E}\left[\frac{1}{\tau T}\sum_{\tau=0}^{T-1}\sum_{j=1}^{\kappa}\mathcal{V}(\overline{z}^{(\tau,j)}) - \mathcal{V}(z^{\star})\right] \leq \frac{\Lambda^2}{2\gamma\tau T} + \frac{\gamma\sigma^2}{K} + 4\tau\gamma^2L\sigma^2 + 18\kappa^2\gamma^2L\zeta^2, \tag{4.2.5}$$

where we set $\Lambda := \|z^{(0,0)} - z^{\star}\|$. Foremost, when the client learning rate is chosen as

$$\gamma = \min\left\{\frac{1}{4L}, \frac{K^{\frac{1}{2}}\Lambda}{\kappa^{\frac{1}{2}}T^{\frac{1}{2}}\sigma}, \frac{\Lambda^{\frac{2}{3}}}{\kappa^{\frac{2}{3}}T^{\frac{1}{3}}L^{\frac{1}{3}}\sigma^{\frac{2}{3}}}, \frac{\Lambda^{\frac{2}{3}}}{\kappa T^{\frac{1}{3}}L^{\frac{1}{3}}\zeta^{\frac{2}{3}}}\right\}, \tag{4.2.6}$$

we have

$$\mathbb{E}\left[\frac{1}{\kappa T}\sum_{\tau=0}^{T-1}\sum_{j=1}^{\kappa}\mathcal{V}(\overline{z}^{(\tau,j)}) - \mathcal{V}(z^{\star})\right] \leq \frac{2L\Lambda^2}{\kappa T} + \frac{2\sigma\Lambda}{\sqrt{K\kappa T}} + \frac{5L^{\frac{1}{3}}\sigma^{\frac{2}{3}}\Lambda^{\frac{4}{3}}}{\kappa^{\frac{1}{3}}T^{\frac{2}{3}}} + \frac{19L^{\frac{1}{3}}\zeta^{\frac{2}{3}}\Lambda^{\frac{4}{3}}}{T^{\frac{2}{3}}}. \tag{4.2.7}$$

We can gain useful insights for FedAvg or local-update algorithms by referring to Theorem 4.2.5. In the following discussion, we will delve into the implications of Theorem 4.2.5 in detail. These findings are consistent with previous research [53, 57, 56].

4.2.6 Discussion. Insights for FedAvg or local-update algorithms can be gained by utilizing Theorem 4.2.5. To provide a more detailed analysis, we will explore the implications of Theorem 4.2.5. Previous literature [48, 53, 61, 57, 56] also supports similar conclusions.

Effects of local steps. To accurately compare performance, it's important to consider synchronous SGD, which synchronizes local models at each local iteration (in FedAvg, this means setting the number of local steps τ to 1). In synchronous SGD, the last two terms in equations (4.2.5) and (4.2.7) are not present, and the rate of convergence is $\mathcal{O}(1/\tau T + \sigma/\sqrt{K\tau T})$. However, if clients perform multiple local steps, there will be an additional error in the upper bound (the last two terms in equations (4.2.5) and (4.2.7)). Fortunately, this additional error is proportional to γ^2 and can decay at a rate of $\mathcal{O}(1/T^{\frac{2}{3}})$ (assuming the learning rate is small enough based on Theorem 4.2.6). As long as the total communication rounds T are sufficiently large and there is non-zero stochastic noise ($\sigma^2 \neq 0$), performing local steps will not hinder the rate of convergence.

Savings in communication rounds. Implementing local steps can reduce the total number of communication rounds in comparison to synchronous SGD. This means that if the local steps τ are adjusted and the total number of gradient evaluations/computations across all clients ($M = K\tau T$) is kept constant, it can result in significant improvement as

$$\tau \leq \min\left\{\frac{\sigma}{DL}\frac{M^{\frac{1}{2}}}{K^2}, \frac{\sigma}{\zeta}\sqrt{\frac{\sigma}{DL}\frac{M^{\frac{1}{2}}}{K^2}}\right\}, \tag{4.2.8}$$

which is the upper bound error (4.2.7). This is mainly influenced by the second term $\mathcal{O}(1/\sqrt{M})$, which is equivalent to synchronous SGD. However, the local-update algorithm requires only T communication rounds in M parallel SGD iterations, whereas synchronous SGD requires τT communication rounds. As the upper bound of local steps (4.2.8) increases, there will be greater communication savings. Previous works have reported that (4.2.8) represents the largest savings in communication rounds.

Effects of data heterogeneity. According to Theorem 4.2.5, data heterogeneity worsens the negative effects of local updates on convergence. It's important to note that the additional error terms in Theorem 4.2.5 are of the order $\mathcal{O}(\tau^2)$. However, if all local objectives are the same ($\zeta = 0$), then the additional error terms will only be linear to the number of local steps. Additionally, from (4.2.8), we can see that when there is high data heterogeneity ($\zeta \gtrsim \sigma$), the maximum number of local steps is $\kappa = \mathcal{O}(K^{\frac{1}{4}}M^{-1})$, which is much smaller than in the case of an IID data distribution ($\kappa = \mathcal{O}(K^{\frac{1}{2}}M^{-2})$.

4.2.7 Remark. Although the discussions regarding Theorem 4.2.5 mostly apply to the stochastic setting ($\sigma \neq 0$), it is important to note that the conclusions may differ in the deterministic setting. This setting involves clients using full-batch gradients to perform local updates, and $\sigma = 0$. In this case, it should be noted that the rate of convergence under heterogeneous data sets will be significantly slowed down to $\mathcal{O}(1/T^{\frac{2}{3}})$ from $\mathcal{O}(1/\tau T)$.

4.3 Experimental setup

4.3.1 Training process. To evaluate optimizer algorithm performance, we created a simulated environment that trains multiple local neural network models. These models are then combined into a single model using the designated algorithms. Federated Learning (FL) relies heavily on the aggregation algorithm, specifically FedAvg which is used to calculate the average of local model updates. This iterative process involves multiple steps that work together to create a seamless and productive learning experience.

- **Initialization:** At the start of the training, a central server initializes a global model and distributes it to all the participating clients.
- **Local training:** The client receives the model which is then trained using their local data, resulting in updated parameters.
- **Model aggregation:** The updates are transmitted to the central server, where they are averaged.
- The global model is updated by taking the average of local model updates. This updated global model is then sent back to each client for the next training cycle.
- **Iteration:** The process is repeated a specific number of times until the overall model reaches a state of convergence.

The information presented here pertains to the FedAvg algorithm, which can be further explored along with other relevant techniques (e.g., [35, 39]).

Algorithm 2: FedAvg algorithm

```
Input  : $K$: number of clients
         $E$: number of local training iterations per client
         $B$: batch size for each client
         $C$: proportion of clients selected at each global communication iteration
         $\eta$: learning rate
         $\alpha$: aggregation coefficient for global update
         $w_0$: initialization of model weights
         $N$: number of rounds
Output: $w$: trained global model
1  Initialize global model: $w \leftarrow w_0$;
2  for $round \leftarrow 1$ to $N$ do
3      $S \leftarrow$ (fraction of $C$ clients chosen randomly);
4      for $k$ in $S$ do
5          $w_k \leftarrow w$;
6          for $epoch \leftarrow 1$ to $E$ do
7              $B_k \leftarrow$ (a batch of data of size $B$ selected randomly);
8              $w_k \leftarrow w_k - \eta \nabla f_i(w_k, B_k)$;
9          end for
10         $\Delta w_k \leftarrow w_k - w$;
11     end for
12     $w \leftarrow w + \alpha \frac{1}{|S|} \sum_{k \in S} \Delta w_k$;
13 end for
14 return $w$;
```

4.3.2 Datasets. For the experiment, we have utilized the FedAvg technique on the MNIST dataset [58], examined both IID (Independent and Identically Distributed) and non-IID data in the Federated Learning scenario. Specifically, we created local datasets that were highly skewed, consisting mostly of instances from the same class. We did not consider the unbalanced nature of the data and assigned each node the same amount of data. Our aim was to test if methods that work in this simplified setting could be effective in real-world use cases.

We employed three different architectures: ANN, CNN, and ResNet. Our models are described in detail and include two variations for each architecture introduced in Chapter 2. The training process utilized 85.71% of the available data, which equates to 60000 images out of 70000. The dataset consists of black and white digit images ranging from 0 to 9, which are handwritten and normalized. Our models were trained across 100 clients.

4.3.3 Remark. When data is IID, each image represents a unique manuscript digit, and the probability distribution of each image is the same. However, if the data is divided into subgroups based on authors or writing styles, it becomes non-ID data. This means that the probability distribution may differ between the subgroups, which is more reflective of real-world scenarios.

4.3.4 Roadmap for Implementation. We utilized Pytorch to execute the FedAvg algorithm and tested it on a server equipped with four V100 GPUs. To facilitate distributed computing, we employed MPI as the communication backend. Our client training optimizer of choice was Adam and SGD, with a fixed learning rate of $\eta = 0.01$ and a weight decay of $1e - 4$. We conducted 300 rounds of global communication on our dataset, but the illustration presented here is based on the 200 round data. Additionally, we set the learning epoch to a fixed value of 5.

4.3.5 The architecture of our models. We evaluate six models with finite neural network architectures to determine the configuration that can best predict unseen data while also having minimal computation time.

- FedAvg-MLP1. The neural network has a single hidden layer that consists of 200 units, and it uses ReLU activation functions.

- FedAvg-MLP2. The neural network has two hidden layers of 200 units each and use of ReLU activation functions.

- FedAvg-CNN1. The model consists of two convolution layers, both with dimensions of $5 \times 55 \times 55 \times 5$ and channel sizes of 5 and 10. Each layer is then followed by a max pooling layer with dimensions of $2 \times 22 \times 22 \times 2$, a fully connected layer with 50 units, ReLU activation functions, and a final softmax output layer.

- FedAvg-CNN2. The architecture of the network includes two convolutional layers, each with dimensions of $5 \times 55 \times 55 \times 5$ and containing 32 and 64 channels, respectively. After each convolutional layer, there is a max pooling layer with dimensions of $2 \times 22 \times 22 \times 2$. Furthermore, there is a fully connected layer with 512 units and a ReLU activation function, followed by a final softmax output layer.

- FedAvg-ResNet1. There are two residual blocks, where each block consists of two convolution layers having the same number of channels and kernel size as in FedAvg-CNN1.

- FedAvg-ResNet2. Two residual blocks, wherein each block consists of two convolution layers with the same channel number and kernel size as those present in FedAvg-CNN2.

The evaluations conducted in this project pertain to two aspects: a global test and individual client tests to analyze the behavior of each client on their respective data.

4.4 Results overview

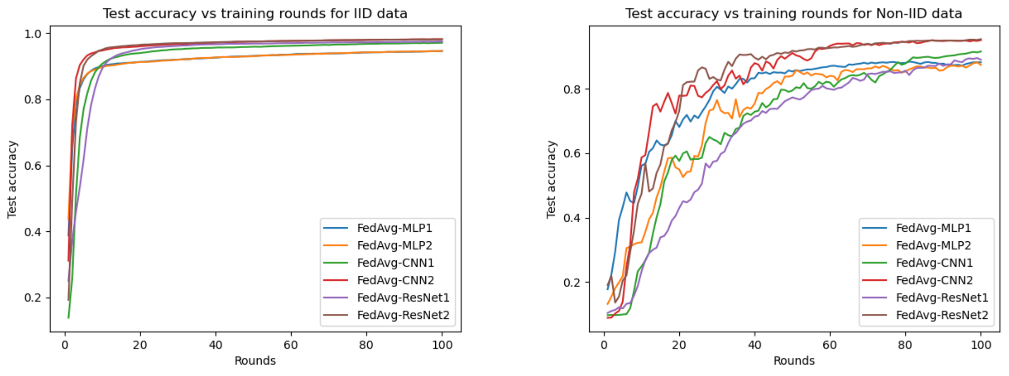


Figure 4.2: These different figures evaluate the performance of the models on IID and NON-IID data, where the figures on the left represent respectively the test accuracy of the IID data and on the right the NON-IID data.

Through an overall assessment of these models, we can determine the level of contribution of each client to the collaborative training process that leads to the development of a successful overall model. Based on the six models presented above we obtain the following results: The parameters used for our implementation:

- $K = 100$: clients
- $M = 10$: Number of clients chosen at random for each round of training.
- $E = 5$: Number of steps for performing gradient descent during each client update:
- $\eta = 0.01$: learning rate

We would like to bring to your attention that the technique of aggregation FEDAVG-MLP1 and its averaging algorithm are derived from a source mentioned in the citation ([34]). In the initial rounds, all the models exhibit similar behavior with a loss that fluctuates around 2.2 and an accuracy of 0.48. However, as the number of rounds progresses, the loss reduces and the accuracy improves, indicating that the models learn more effectively from their partners.

These are the findings from the 100th round. During this round, the MLP1 and MLP2 models incurred a loss of around 0.5 while maintaining a 94% accuracy. On the other hand, the CNN and RESNET models, which performed almost equally well, experienced a loss of 0.033 and achieved a 98% accuracy. Although these models deliver excellent results, they require significant computation time.

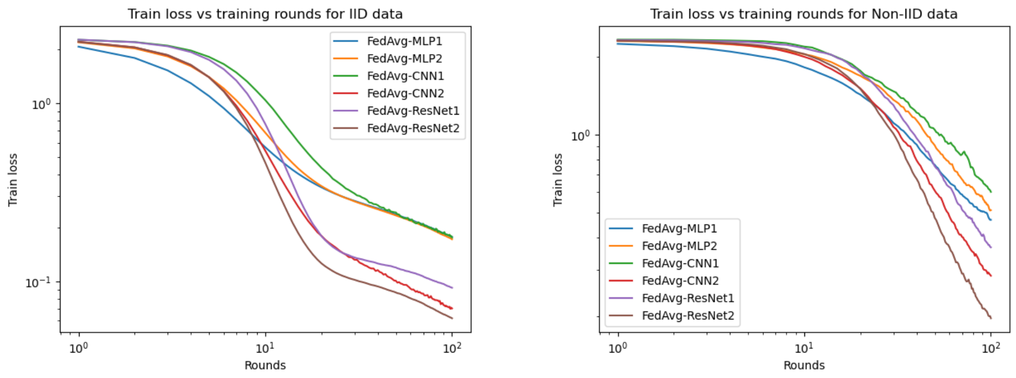


Figure 4.3: These different figures evaluate the performance of the models on IID and NON-IID data, with the figures on the left representing train loss respectively for IID data and those on the right for non-ID data.

The training loss evaluates the performance of our models on both types of data (IID and NON-IID) i.e. how well the model adjusted to the data it saw during training. Where we notice that the models adjusted faster to IID data during training than to NON-IID data.

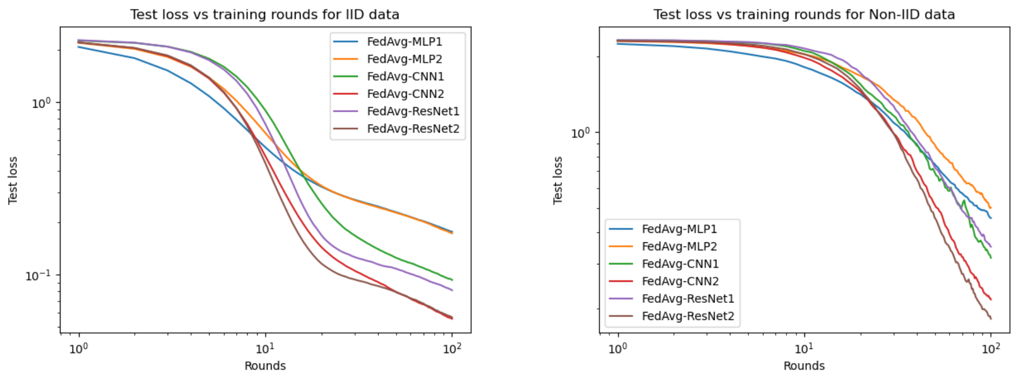


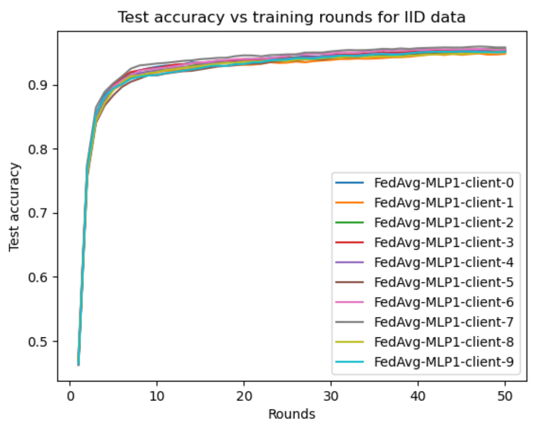
Figure 4.4: These different figures evaluate the performance of the models on IID and NON-IID data, with the figures on the left representing test loss respectively for IID data and those on the right for non-ID data.

The test loss, on the other hand, measures the performance of our models on a separate test set of data, which assesses the ability of the model to generalize to new data that it did not see during training.

When the difference between train loss and test loss is large it means that the model has overfitted the training data, based on the train loss and test loss figures of both types of data we observe that this difference is not large. So our models will predict with the above-mentioned percentage of accuracy on new data that they have not seen during training.

Comparison IID and non-IID Data Sets: It has been observed that when dealing with IID data (which is identically distributed), most models achieve good accuracy faster at around the 15th round. However, NON-IID data (which is not identically distributed) presents a different scenario. In this case, certain data points may take longer to learn, resulting in an increase in the number of training rounds required to achieve good accuracy (typically around 60 rounds). The loss train and loss test exhibit a similar trend, where they decrease and tend towards a minimum value of around 100 rounds for IID data. However, for NON-IID data, the cost function does not tend towards its minimum even at 100 rounds. This is because customers tend to focus on the minima of their local objectives, and the aggregate global model may not be the optimum of the global objective [60].

In order to illustrate this phenomenon as well as possible we represent on a particular model $(\mathrm{MLP1})$ the behavior of an example of 10 clients during their training those made for 50 rounds, the following figures confirm in fact the character of distribution i.e. for IID and for NON-IID data stated earlier.



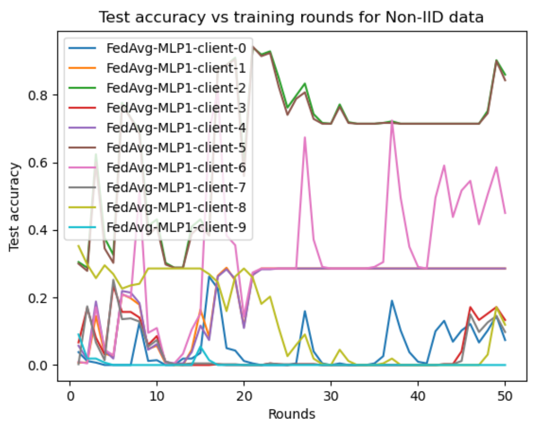


Figure 4.5: These two figures illustrate the behavior of the data during training, the one on the (left) for IID data and the one on the (right) for NON-IID data.

5. Conclusion and future outlooks

The focus of this thesis was on "Gradient Descent on Infinitely Wide Neural Network and Federated Learning". The aim was to propose a method that leads to faster convergence towards the global optimum. Later, this method will be used to investigate the behavior of the Federated learning approach in light of this innovation.

To start, we needed to comprehend the two approaches independently. It is important to note that the federated learning found in literature is created using limited neuron-width models, like most models in machine learning. As a result, convergence is quick, but reaching a global optimum is challenging. This is why methods like Shadow sequence convergence exist for homogeneous data, but not for heterogeneous data, which slows down convergence considerably.

In Chapter 3, we explained how having more hidden neurons, denoted by m, can lead to better results. When m approaches infinity, the gradient flow will reach the global minimum of the cost function. This is proved through the homogeneity properties of the ReLU activation.

Ensuring a more precise convergence towards the global optimum is crucial for better performance. The literature on infinitely wide neural networks outlines properties that can guarantee this convergence for random initializations. The gradient behavior is of particular interest in this study. Commonly, the descending gradient method is used in various types of neural networks. As the number of neurons on the hidden layer approaches infinity, they behave like a continuum. Consequently, the optimization problem becomes a probability measurement problem, and the SGD method is replaced by the gradient flow.

However, it is important to note that we cannot determine exactly how large m needs to be to approach the infinite width limit, nor can we determine the speed at which the gradient flow will reach the global optimum. These questions are still being researched.

We experimented to demonstrate a principle. We trained a neural network with a hidden layer using a teacher-student approach, and we experimented with three different values of the number of neurons m: 10, 50, and 2000. Our findings indicate that as the number of neurons in the cache layer (represented by m) increases, the flow improves, and this leads to better convergence.

In Chapter 4, we explored the issue of Federated Learning on various neural networks. We delved into the theoretical optimization problem behind it, including the gradient descent methods involved. To better understand Federated Learning, we used homogeneous data (MNIST) and experimented with six different models, testing them on both IID and $\textsc{non}$-IID distributions. Overall, our results were satisfactory, with the MLP models achieving a 94% accuracy rate and the CNN and $\textsc{ResNets}$ models achieving a 98% accuracy rate.

We are interested in exploring ways to ensure better convergence in convolutional and deep networks, particularly in the context of Federated Learning with heterogeneous data. Our goal is to combine these two approaches (namely, gradient descent on infinitely wide neural networks and federated learning) to help local models converge toward their global optimum and observe their behavior when aggregated.

Acknowledgements

I am filled with gratitude to God for the countless blessings He has bestowed upon me during this program. His grace, good health, and unwavering perseverance have been an incredible source of strength for me.

I wish to express my sincere appreciation to everyone who played a role in making my thesis project a reality. First and foremost, I would like to thank my esteemed supervisor, Prof. Dr. Enrique Zuazua, for sharing his invaluable expertise and guiding me in my research. I am also grateful to the members of the Dynamics, Control, and Numerics FAU chair, particularly Dr. Jean-Daniel Djida and Ziqi Wang for their unwavering support throughout the project.

My heartfelt thanks go to the President of AIMS Cameroon, Prof. Dr. Mama Foupouagnigni; the Academic Director, Dr. Daniel Duviol Tcheutia; Cooperative Education Manager, Mr. Honore Youfegnuy, the lecturers, tutors, and all those who made my learning experience possible. I cherish the memories I have made with my classmates and the staff, and I am grateful to the AIMS family. Lastly, I would like to acknowledge the support of the MasterCard Foundation for making this program possible.

I express my sincere appreciation for the consistent support and direction that my mentor, Dr. Vitalis K. Lagat, Sylvain Wagoum Nodem, and Junior Momo Ziazet provided me with throughout the project's completion. Their unwavering encouragement and aid have been of immeasurable value.

I would like to extend a special thank you to all those who have been a part of my journey.

References

[1] L. Ambrosio. Gradient Flows in Metric Spaces and in the Spaces of Probability Measures, and Applications to Fokker-Planck Equations with Respect to Log-Concave Measures. *Bollettino dell'Unione Matematica Italiana*, 1(1):223–240, 2 2008.

[2] L. Ambrosio, N. Fusco, and D. Pallara. *Functions of bounded variation and free discontinuity problems*. Courier Corporation, 2000.

[3] F. Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(1):629â681, 2017.

[4] F. Bach. Gradient descent for wide two-layer neural networks - I : Global convergence. https://francisbach.com/gradient-descent-neural-networks-global-convergence/, 2020. Accessed: May 16, 2023.

[5] F. Bach and L. Chizat. Gradient Descent on Infinitely Wide Neural Networks: Global Convergence and Generalization. Technical report, ArXiv: 2110.08084v1, 2021.

[6] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[7] Francis Bach and Le̓naïc Chizat. Gradient descent on infinitely wide neural networks: Global convergence and generalization. *arXiv preprint arXiv:2110.08084*, 2021.

[8] A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[9] Y. Bengio, N. Roux, P. Vincent, O. Delalleau, and P. Marcotte. Convex neural networks. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

[10] L. Bottou, Frank E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *SIAM Review*, 60(2):223–311, 2018.

[11] A. C. Brooms. Stochastic Approximation and Recursive Algorithms with Applications, 2nd Edn by H. J. Kushner and G. G. Yin. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 169(3):654–654, 06 2006.

[12] S. Bubeck. Convex Optimization: Algorithms and Complexity. *Found. Trends Mach. Learn.*, 8(3â4):231â357, nov 2015.

[13] L. Chizat. Sparse optimization on measures with over-parameterized gradient descent. *Math. Program.*, 194:487â532, 2022.

[14] L Chizat and F. Bach. On the Global Convergence of Gradient Descent for Over-Parameterized Models Using Optimal Transport. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 3040–3050, Red Hook, NY, USA, 2018. Curran Associates Inc.

[15] L. Chizat and F. Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. *arXiv preprint arXiv:2002.04486*, 2020.

[16] L. Chizat, E. Oyallon, and F. Bach. On lazy training in differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018.

[17] L. Cohn. volume 165. Springer, 1980.

[18] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, page 303â314, 1989.

[19] Cynthia Dwork and Aaron Roth et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[20] C. Esteve, B. Geshkovski, D. Pighin, and E. Zuazua. Large-time asymptotics in deep learning. *ArXiv*, abs/2008.02491, 2020.

[21] C. Fang, Y. Guo, Y. Hu, B. Ma, L. Feng, and A. Yin. Privacy-preserving and communication-efficient federated learning in Internet of Things. *Computers Security*, 103:102199, 2021.

[22] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. CCS '15, pages 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[23] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients - how easy is it to break privacy in federated learning? *CoRR*, abs/2003.14053, 2020.

[24] Z. Han, G. Xi, U. Jacob, and A. Tom. Approximation capabilities of neural ODEs and invertible residual networks. In *International Conference on Machine Learning*, pages 11086–11095. PMLR, 2020.

[25] L. Heiko and N. Baracaldo. *Federated Learning*, volume 96. Springer Cham, 1 edition, 2022.

[26] K. Hornik, M. Stinchcombe, and W. Halbert. Multilayer feedforward networks are universal approximators. *Neural networks*, pages 359–366, 1989.

[27] V. Jeyakumar and H. Wolkowicz. Generalizations of Slater's constraint qualification for infinite convex programs. *Mathematical Programming*, 57:85–101, 1992.

[28] M. Johannes. Universal flow approximation with deep residual networks. *arXiv preprint arXiv:1910.09599*, 2019.

[29] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] Peter Kairouz, H. Brendan McMahan, Brendan Avent, AurÃ©lien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawit, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, AdriÃ GascÃ$^3n, BadihGhazi, Phillip B.Gibbons, MarcoGruteser, ZaidHarchaoui, ChaoyangHe, LieHe, Z$

[31] R. Lars and H. Eldad. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pages 1–13, 2019.

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

[33] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[34] 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.

[35] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282, 2017. Initial version posted on arXiv in February 2016.

[36] S. Mei, A. Montanari, and P. Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[37] Atsushi Nitanda and Taiji Suzuki. Stochastic particle gradient descent for infinite ensembles. *arXiv preprint arXiv:1712.05438*, 2017.

[38] G. Peyré and M. Cuturi. 2019.

[39] Kiyoshi Nakayama Phd and George Jeno. *Federated Learning with Python: Design and implement a federated learning system and develop applications using existing frameworks*. Packt Publishing, 2022.

[40] A. Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numer.*, 8(1):143â195, 1999.

[41] C. Tian Qi, R. Yulia, B. Jesse, and D. David. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

[42] F. Santambrogio. *Optimal transport for applied mathematicians*. Springer, 2015.

[43] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, Los Alamitos, CA, USA, 2017. IEEE Computer Society.

[44] D. Simon, L. Jason, L. Haochuan, Wang. Liwei, and Z. Xiyu. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

[45] J. Sirignano and K. Spiliopoulos. Mean Field Analysis of Neural Networks: A Law of Large Numbers. *SIAM Journal on Applied Mathematics*, 80(2):725–752, 2020.

[46] M. Song, M. Andrea, and N. Phan-Minh. A mean field view of the landscape of two-layer neural networks. *Proc. Natl. Acad. Sci. U.S.A.*, 115(33):E7665–E7671, 2018.

[47] G. Stefanie, R. Lars, S. Jacob B, C. Eric. C, and G. Nicolas. R. Layer-parallel training of deep residual neural networks. *SIAM J. Math. Data Sci.*, 2(1):1–23, 2020.

[48] Sebastian U Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations (ICLR)*, 2019.

[49] E. Süli and F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

[50] P. Sunghwan, S. Yeryoung, and L. Jaewoo. Fedpso: Federated learning using particle swarm optimization to reduce communication costs. *Sensors*, 21(2), 2021.

[51] P. Tabuada and B. Gharesifard. Universal approximation power of deep residual neural networks via nonlinear control theory. In *International Conference on Learning Representations*, 2021.

[52] Muhammad Habib ur Rehman and Mohamed Medhat Gaber. *Federated learning systems: Towards next-generation AI*, volume 965. Springer Nature, 2021.

[53] Jianyu Wang and Gauri Joshi. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

[54] E. Weinan. A proposal on machine learning via dynamical systems. *Commun. Math. Stat.*, 5(1):1–11, 2017.

[55] S. Wojtowytsch. On the convergence of gradient descent training for two-layer relu-networks in the mean field regime. *CoRR*, abs/2005.13530, 2020.

[56] Blake Woodworth, Kumar Kshitij Patel, and Nathan Srebro. Minibatch vs local SGD for heterogeneous distributed learning. *arXiv preprint arXiv:2006.04735*, 2020.

[57] Blake Woodworth, Kumar Kshitij Patel, Sebastian U. Stich, Zhen Dai, Brian Bullins, H. Brendan McMahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? *arXiv preprint arXiv:2002.07839*, 2020.

[58] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017. cite arxiv:1708.07747Comment: Dataset is freely available at https://github.com/zalandoresearch/fashion-mnist Benchmark is available at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/.

[59] L. Yann, Touresky. D, Hinton. G, and Sejnowski. T. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

[60] Dezhong Yao, Wanning Pan, Yutong Dai, Yao Wan, Xiaofeng Ding, Hai Jin, Zheng Xu, and Lichao Sun. Local-global knowledge distillation in heterogeneous federated learning with non-iid data. *arXiv preprint arXiv:2107.00051*, 2021.

[61] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5693–5700, 2019.

[62] K. Yu, X. Zhang, Z. Ye, Gong-De. Guo, and S. Lin. Quantum federated learning based on gradient descent. Technical report, Arxiv:2212.12913, 2023.

[63] A-Z. Zeyuan, L. Yuanzhi, and S. Zhao. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.