ABSTRACT

Title of Dissertation: QUANTIZATION AND APPROXIMATION OF NEURAL NETWORKS

Sanghoon Na
Doctor of Philosophy, 2026

Dissertation Directed by: Professor Wojciech Czaja
Department of Mathematics

This thesis investigates several fundamental mathematical problems arising in machine learning, with a specific focus on the theoretical properties of neural networks. The first part of this work addresses neural network quantization. Despite the widespread use of quantization as a primary method for compressing deep neural networks, a majority of existing approaches lack provable guarantees. By leveraging tools from frame theory and signal processing, we establish a neural network quantization framework that provides rigorous error estimates. Furthermore, we explore the universal approximation theory of quantized networks, focusing on the most extreme case: one-bit neural networks.

The second focus of this thesis pertains to the theoretical analysis of neural network optimization, supported by the foundations of neural network approximation theory. We investigate the optimization of shallow neural networks to determine how the regularity of the target function influences the convergence rate. Finally, we analyze the neural collapse phenomenon under the unconstrained feature model setting, an analytical framework justified by approximation theory.

QUANTIZATION AND APPROXIMATION OF NEURAL NETWORKS

by

Sanghoon Na

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2026

Advisory Committee:
Professor Wojciech Czaja, Chair/Advisor
Professor Haizhao Yang
Professor Radu Balan
Professor Vince Lyzinski
Professor William Gasarch, Dean's Representative

Dedication

This thesis is dedicated to my parents, for their unwavering love and support.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Wojciech Czaja. My Ph.D. journey in College Park began during the challenging era of COVID-19, which marked my first time living in the United States. Starting my program online made adjusting to a new country and finding an advisor particularly difficult. I still vividly remember our first meeting in February 2022; since then, I have been profoundly grateful for his unwavering support. His guidance extended far beyond academic advising, providing me with invaluable mentorship for life.

I am also incredibly fortunate to have worked with Professor Haizhao Yang. I am deeply thankful for his extensive support, both as an academic mentor and a life guide, throughout my graduate career.

My sincere thanks also go to Professor Radu Balan. I gained diverse insights throught the RIT on Applied Harmonic Analysis and benefited greatly from his thoughtful advice. I would like to extend my gratitude to Professor Vince Lyzinksi for his invaluable contributions as a committee member, and to Professor William Gasarch for serving as the Dean's Representative.

I am grateful to Professor Leonid Koralov for his several support in my Ph.D. life and to Professor Larry Washington for his help during the postdoc application process. Additionally, I would like to thank the staff of the Mathematics Department, especially Jemma Natanson and Ruth Yun, for their assistance in various situations.

Finally, I am eternally grateful to my family and friends for their constant encouragement and love. This journey would not have been possible without them.

Table of Contents

Dedication ii

Acknowledgements iii

Table of Contents iv

List of Tables vi

List of Figures vii

List of Notations vii

Chapter 1: Introduction 1
- 1.1 Motivation and Scope . . . 1
- 1.2 Organization of the Dissertation . . . 3

Chapter 2: Preliminaries 5
- 2.1 Finite Frames . . . 5
- 2.2 Quantization for Finite Frames . . . 10
- 2.3 Manifolds . . . 14
- 2.4 Neural Networks . . . 16
- 2.5 Mean-field Parametrization of Neural Networks and Wasserstein Gradient Flows . . . 19
- 2.6 Barron Spaces . . . 20

Chapter 3: Frame Quantization of Neural Networks 22
- 3.1 Introduction . . . 22
- 3.2 Frame Quantization for Neural Networks . . . 24
- 3.3 Error Estimates . . . 26
  - 3.3.1 Feedforward Networks . . . 28
  - 3.3.2 Neural Networks with Residual Blocks . . . 32
- 3.4 Numerical Results . . . 37
  - 3.4.1 Feedforward Network with 3 Layers . . . 38
  - 3.4.2 Network with 2 Residual Blocks . . . 39
  - 3.4.3 1-Bit Quantization . . . 40
  - 3.4.4 ResNet-18 Quantization . . . 42
- 3.5 Conclusions and Future Work . . . 44

Chapter 4: One-bit quantized neural networks on manifolds 46
- 4.1 Introduction . . . 46
- 4.2 Notation and Setting . . . 47
- 4.3 Main Theorem . . . 50
- 4.4 Outline of the proof and supporting lemmas . . . 51

4.4.1 One-bit neural network expansion . . . 52
4.4.2 Approximating multiplication . . . 54
4.4.3 Composition of one-bit neural networks . . . 56
4.4.4 Regularity considerations . . . 59
4.4.5 Estimate under geometric constraints . . . 61
4.5 Proof of the Main Theorem . . . 62
4.5.1 Chart determination . . . 62
4.5.2 Taylor Expansion . . . 66
4.5.3 Estimating the total error . . . 69
4.6 Conclusion and Open Questions . . . 71

Chapter 5: Curse of Dimensionality in Neural Network Optimization 73
5.1 Introduction . . . 73
5.2 Setup . . . 79
5.3 Main Theorems . . . 81
5.4 Key Lemmas . . . 83
5.4.1 Growth of second moments under the 2-Wasserstein gradient flow . . . 83
5.4.2 Barron norms and second moments . . . 84
5.4.3 Slow approximation property across infinitely many time scales . . . 87
5.4.4 Low complexity estimates . . . 93
5.4.5 Curse of dimensionality in numerical integration . . . 94
5.5 Proofs of the Main Theorems . . . 103
5.5.1 Proof of Theorem 5.3.1 . . . 103
5.5.2 Proof of Theorem 5.3.3 . . . 104
5.5.3 Proof of Theorem 5.3.4 . . . 105
5.6 Conclusion . . . 114

Chapter 6: Neural Collapse in the Orthoplex Regime 117
6.1 Introduction . . . 117
6.2 Softmax codes in the Orthoplex regime . . . 119
6.3 Generalized Neural Collapse . . . 122
6.3.1 HardMax problem and Neural Collapse . . . 123
6.3.2 Tammes problem, Softmax codes, and Rattlers . . . 124
6.3.3 Variability Collapse and Self-Duality . . . 128
6.4 Conclusion and Open Questions . . . 130

Bibliography 132

List of Tables

3.1 Frame Quantization for FNN with 3 layers. The test accuracy for the pretrained neural network is $97.72 \pm 0.21\%$. 38
3.2 Frame Quantization for Neural Network with 2 Residual Blocks. The test accuracy for the pretrained neural network is $97.72 \pm 0.20\%$. 40
3.3 1-Bit Frame Quantization for FNN with 3 layers 41
3.4 1-Bit Frame Quantization for Neural Network with 2 Residual Blocks 42
3.5 Test accuracy of b-bit quantized ResNet-18 on CIFAR-10, using Frame Quantization with redundancy ratio $r \in \{1, 1.1, 1.2, 1.3\}$ and GPFQ with calibration set of size $c \in \{128, 256\}$. 43
3.6 1-Bit Frame Quantization for ResNet-18 with CIFAR-10 dataset 43

List of Figures

3.1 Worst-case error $\|f(X) - f_Q(X)\|$ for FNN with 3 layers. 39
3.2 $\log \mathrm{E}_X[\|f(X) - f_Q(X)\| \times N/\delta]$ for FNN with 3 layers. 39

5.1 Geometric description of the proof of Theorem 5.3.3. The green curve with arrow illustrates the sublinear growth of the Barron norm, which follows from Lemma 5.4.1 and Lemma 5.4.3. The shallow neural network at the initialization is denoted as f_{π^t}, represented as a circle filled with dark blue. The shallow neural network training time t is denoted as f_{π^t}, represented as a circle filled with sky-blue. The black dotted line represents Theorem 5.3.1, the existence of $C^r(Q)$ function with slow approximation property. 105

List of Notations

$\mathbb{C}$	Complex numbers
$\mathbb{R}$	Real numbers
$\mathbb{N}$	Set of positive integers
$\mathbb{N}_{\curlyvee}$	Set of nonnegative integers
σ	Neural network activation function
U_Q	Uniform distribution on set Q
$C(X)$	Set of continuous functions on set X
$C^r(X)$	Set of functions that are r times continuously differentiable on X
$\|x\|_X$	Norm of $x \in X$, where X is a normed vector space
$\|x\|$	Euclidean norm of a vector $x \in \mathbb{R}^d$
$\|A\|$	Matrix 2-norm of a matrix A, which is the largest singular value of A
$\|x\|$	Absolute value of $x \in \mathbb{R}$
$\|x\|_1$	1-norm $\sum_{i=1}^d \|x_i\|$ for $x = (x_1, \cdots, x_d) \in \mathbb{R}^d$
$\|x\|_\infty$	The largest absolute value of entries in vector x
$\|A\|_\infty$	The largest absolute value of entries in a matrix A
$\|f\|_\infty$	Supremum norm of function f
B^X	Closed unit ball centered at 0 in X, $\{x \in X : \|x\|_X \leq 1\}$
$B_\epsilon(x)$	Open ball of radius ϵ centered at x, $\{z \in X : \|z - x\|_X < \epsilon\}$
$\boldsymbol{\delta}_x$	Dirac measure at x
$\mathbb{1}_A$	Indicator function of A
$*$	Convolution operator between two functions
$\circ$	Function composition
I_d	Identity matrix of size $d \times d$
$\|\cdot\|_{L^p}$	L^p norm
$\|\cdot\|_0$	Number of nonzero entries in a matrix or a vector
$\lceil\cdot\rceil$	Ceiling operator
F	Finite frame in $\mathbb{R}^d$
H_N^d	Harmonic frame with N elements in $\mathbb{R}^d$
$\mathcal{M}$	Compact Riemannian manifold
$L(X, Y)$	Set of bounded linear operators between two normed spaces X and Y
$\mathrm{Rad}(\mathcal{F}, S)$	Rademacher complexity of $\mathcal{F}$ with respect to set S
ω_d	Volume of a unit ball in $\mathbb{R}^d$
$[n]$	Index set $\{1, \cdots, n\}$ for $n \in \mathbb{N}$
S^{d-1}	Unit sphere in $\mathbb{R}^d$
$\mathrm{conv}\, X$	Convex hull of set $X \subset \mathbb{R}^d$
$\mathrm{dist}(x, X)$	Distance between $x \in \mathbb{R}^d$ and $X \subset \mathbb{R}^d$

Chapter 1: Introduction

1.1 Motivation and Scope

Deep learning has achieved remarkable success in solving complex, high-dimensional tasks that traditional methods struggled to address, including computer vision, natural language processing, protein folding prediction, and the resolution of high-dimensional partial differential equations in scientific computing. The driving force behind these advancements is the deep neural network—an architecture comprising multi-layered artificial neurons loosely inspired by the structure of the human brain.

Modern deep neural networks are heavily over-parameterized and rely on a vast number of computational units, which frequently yields state-of-the-art performance. However, the massive scale of these models introduces significant practical limitations: 1) Their enormous storage requirements prohibit deployment on edge or portable devices, and 2) their computational demands result in unsustainable energy consumption. Consequently, a substantial body of research has emerged aimed at compressing deep neural networks without compromising performance, with quantization, pruning, and low-rank approximation serving as the primary techniques.

Nevertheless, many existing compression techniques prioritize empirical success over rigorous mathematical foundations. For instance, unlike signal processing—where quantization is supported by robust mathematical frameworks—neural network quantization often lacks such theoretical guarantees. This gap in the literature motivated my research into neural network quantization with provable guarantees, which is detailed in Section 3.

A core theoretical inquiry in this space revolves around the approximation theory of quantized neural networks. While general neural network approximation theory has evolved significantly since

the 1980's, the specific approximation theory for quantized networks remains underexplored. My foundational interest in quantization theory naturally guided me toward this domain. Section 4 presents my findings regarding the universal approximation theory of quantized neural networks.

Despite deep neural networks serving as the state-of-the-art architecture across various disciplines, many aspects of their underlying mechanics remain a theoretical mystery. Within neural network optimization theory, considerable efforts have been made to explain their effectiveness. Yet, the majority of these analyses depend on unrealistic assumptions—such as extreme over-parameterization or infinite-width limits—that do not reflect practical implementation. Such analyses are inherently challenging due to the non-convex nature of the optimization landscape. Furthermore, existing quantitative convergence results predominantly focus on the linear convergence rate of empirical risk, leaving the convergence behavior of population risk unaddressed. Motivated by this theoretical void, Section 5 demonstrates the manifestation of the curse of dimensionality within neural network optimization, thereby establishing a critical connection between neural network approximation and optimization theories. It reveals an intriguing paradox: while deep neural networks demonstrate exceptional empirical performance in high-dimensional problems, even shallow networks can theoretically fall victim to the curse of dimensionality.

Neural collapse is an intriguing phenomenon observed during the terminal phase of training deep neural networks for classification. It describes the convergence of class-mean feature vectors and linear classifier weights toward a geometric structure known as a simplex equiangular tight frame, up to a scaling factor. While this empirical finding has been theoretically justified by treating features as free optimization variables—a framework supported by neural network approximation theory—most existing proofs require the feature space dimension exceeds the number of classes. In Section 6, we provide a proof for the emergence of neural collapse in the opposite case, specifically within the

orthoplex regime.

1.2 Organization of the Dissertation

Chapter 2 introduces the foundational theory essential for the developments presented in subsequent chapters. It begins with an exploration of finite frame theory and quantization in signal processing, especially focusing on the first-order sigma-delta ($\Sigma\Delta$) quantization method. Furthermore, it addresses key concepts in manifolds and examines neural network architectures. Barron spaces and mean-field neural networks are also introduced at the last.

Chapter 3, based on our published work [27][1], introduces a post-training quantization method with provable guarantees. Drawing upon finite frame theory and first-order $\Sigma\Delta$ quantization techniques from signal processing, we establish a robust theoretical framework. This is supported by empirical validation on toy problems, alongside practical results demonstrating the method's efficacy on a ResNet-18 model for the CIFAR-10 classification task.

Chapter 4 details our ongoing research with Professor Wojciech Czaja and Professor Weilin Li, into the curse of dimensionality within the universal approximation theory of one-bit quantized neural networks. By restricting the input data to a compact manifold, we demonstrate that the exponential complexity required to achieve a supremum error ϵ scales with d, the intrinsic dimension of the data manifold, rather than the dimension of the entire space.

Chapter 5 is based on our paper [81][2]. We prove that functions susceptible to the curse of

[1] Appeared in *Journal of Fourier Analysis and Applications* (2025). The author of this thesis conducted the theoretical developments, including all theorems and proofs, performed the numerical experiments, and managed the editorial process under the guidance of Professor Wojciech Czaja.

[2] Accepted for publication in *Information and Inference: A Journal of the IMA*. While the research problem was proposed by Professor Haizhao Yang, the author of this thesis was responsible for the entire execution of the study, including the derivation of all theorems and the preparation of the manuscript.

dimensionality when approximated via Barron spaces require exponential training time for the population risk to converge, and in some cases, may never achieve convergence. We analyze this phenomenon in the context of the target function's regularity and the selected class of activation functions. These findings effectively bridge the theoretical gap between neural network approximation and optimization.

Chapter 6 presents author's primary contributions to the recent joint preprint work [1][3]. This chapter investigates the geometry of n points in the unit sphere S^{d-1} within the 'orthoplex regime' $n \in [d+2, 2d]$. Building upon these geometric results, we analyze the neural collapse when the number of classes n falls within $[d+2, 2d]$, where d is the dimension of the feature space. The proofs and the manuscript of this chapter were independently developed and authored by the candidate.

[3]While the research problem was proposed by Professor Dustin Mixon, the author of this thesis was responsible for the theoretical developments in Sections 2 and 3 of this paper. Additionally, the author formulated the core proposal and established the proof of Lemma 14 in Section 4 of the paper.

Chapter 2: Preliminaries

This chapter establishes the fundamental concepts and theoretical framework essential for the subsequent discussions. Specifically, Sections 2.1 and 2.2 support the analysis in Chapter 3, while Section 2.3 provides the necessary background for Chapter 4. Section 2.4 covers the basic concepts of neural networks, which are essential for all subsequent chapters. Finally, the material in Sections 2.5 and 2.6 is presented in preparation for Chapter 5.

2.1 Finite Frames

In this section, we review key concepts from frame theory. Our focus is restricted to finite frames in $\mathbb{R}^d$. For a broader discussion of finite frames in general Hilbert spaces, we refer the reader to [16].

Definition 2.1.1. *A set $\{e_1, \cdots, e_N\}$ in $\mathbb{R}^d$ is a finite frame for $\mathbb{R}^d$ if there exists $0 < A \leq B < \infty$ such that*

$$A\|x\|^2 \leq \sum_{i=1}^{N} |\langle x, e_i \rangle|^2 \leq B\|x\|^2 \tag{2.1}$$

holds for all $x \in \mathbb{R}^d$. Here, $\|\cdot\|$ denotes the Euclidean norm.

The constants A and B are called *frame bounds,* and $\{\langle x, e_i \rangle\}_{i=1}^N$ are called the *frame coefficients* of x with respect to frame $\{e_1, \cdots, e_N\}$. A frame $\{e_1, \cdots, e_N\}$ in $\mathbb{R}^d$ is called *tight* if $A = B$. If a finite tight frame $F = \{e_1, \cdots, e_N\}$ satisfies $\|e_i\| = 1$ for all i, then we say F is a *finite unit-norm tight frame* (FUNTF).

Definition 2.1.2. *Let $F = \{e_1, \cdots, e_N\}$ be a finite frame for $\mathbb{R}^d$. The linear function $S : \mathbb{R}^d \to \mathbb{R}^d$*

defined by

$$Sx = \sum_{i=1}^{N} \langle x, e_i \rangle e_i, \tag{2.2}$$

is the frame operator of F.

Note that if $F = \{e_1, \cdots, e_N\}$ is a finite frame for $\mathbb{R}^d$ with frame bounds A and B, then it is easy to see that S is a $d \times d$ positive definite matrix satisfying $AI_d \leq S \leq BI_d$, where I_d is the identity matrix of $\mathbb{R}^d$. Therefore, its inverse S^{-1} exists and it satisfies $\frac{1}{B} I_d \leq S^{-1} \leq \frac{1}{A} I_d$. This inverse operator S^{-1} is called the *dual frame operator*. Multiplying (2.2) by S^{-1} gives us an atomic decomposition of an arbitrary $x \in \mathbb{R}^d$:

$$x = S^{-1}Sx = S^{-1}\Big\{ \sum_{i=1}^{N} \langle x, e_i \rangle e_i \Big\} = \sum_{i=1}^{N} \langle x, e_i \rangle S^{-1} e_i. \tag{2.3}$$

It is straightforward to check $\{S^{-1}e_1, \cdots, S^{-1}e_N\}$ is a frame for $\mathbb{R}^d$ with frame bounds $1/B$ and $1/A$. We say that $\{S^{-1}e_1, \cdots, S^{-1}e_N\}$ is the *canonical dual frame* of F. For detailed proofs of the aforementioned statements, see [16].

When $F = \{e_1, \cdots, e_N\}$ is a tight frame for $\mathbb{R}^d$ with frame bound A, then $S = AI_d$ and $S^{-1} = \frac{1}{A} I_d$. In this case, (2.3) gives us the following frame expansion.

Lemma 2.1.3. *Let* $F = \{e_1, \cdots, e_N\}$ *be a finite tight frame for* $\mathbb{R}^d$ *with frame bound* A. *Then for any* $x \in \mathbb{R}^d$, *we have the following frame expansion of* x*:*

$$x = \frac{1}{A} \sum_{i=1}^{N} \langle x, e_i \rangle e_i. \tag{2.4}$$

Moreover, if F is a FUNTF, we can express the frame bound A in terms of d and N.

Theorem 2.1.4. *Let $F = \{e_1, \cdots, e_N\}$ be a FUNTF for $\mathbb{R}^d$ with frame bound A. Then, $A = N/d$ holds.*

Proof. From the assumption, for any $x \in \mathbb{R}^d$, we have $A\|x\|^2 = \sum_{i=1}^N |\langle x, e_i\rangle|^2$. Choose an orthonormal basis $\{v_1, \cdots, v_d\}$ of $\mathbb{R}^d$. Taking $x = v_j$, we get

$$A = A\|v_j\|^2 = \sum_{i=1}^N |\langle v_j, e_i\rangle|^2$$

for $j = 1, \cdots, d$. The summation over j's gives us

$$Ad = \sum_{j=1}^d A\|v_j\|^2 = \sum_{j=1}^d \sum_{i=1}^N |\langle v_j, e_i\rangle|^2 = \sum_{i=1}^N \sum_{j=1}^d |\langle e_i, v_j\rangle|^2 = \sum_{i=1}^N \|e_i\|^2 = N.$$

Note that the last equality holds since $\|e_i\| = 1$ for all i's. Therefore, we conclude that $A = N/d$. □

Hence, when F is a FUNTF, we have the following frame expansion

$$x = \frac{d}{N} \sum_{i=1}^N \langle x, e_i\rangle e_i \tag{2.5}$$

for any $x \in \mathbb{R}^d$.

Now we review the concept of *frame variation*, which is important in quantization with finite frames.

Definition 2.1.5. *Given a finite frame $F = \{e_1, \cdots, e_N\}$ for $\mathbb{R}^d$, its frame variation with respect to a*

permutation p of $\{1, 2, \cdots, N\}$ is defined as

$$\sigma(F, p) := \sum_{i=1}^{N-1} \|e_{p(i)} - e_{p(i+1)}\|. \tag{2.6}$$

Frame variation is a measurement that captures the interdependency between the frame elements resulting from the choice of p. If $\sigma(F, p)$ is small, then we can say that the frame elements do not oscillate too much in that ordering. When $F = \{e_1, \cdots, e_N\}$ is a finite unit-norm frame, then an obvious upper bound for a frame variation would be

$$\sigma(F, p) = \sum_{i=1}^{N-1} \|e_{p(i)} - e_{p(i+1)}\| \leq \sum_{i=1}^{N-1} \{\|e_{p(i)}\| + \|e_{p(i+1)}\|\} = 2(N-1).$$

However, this upper bound is too large and does not show the effect of p on $\sigma(F, p)$. In [118], the author proves the following theorem, which leads to the existence of a permutation p of $\{1, 2, \cdots, N\}$ that makes the frame bound is $O(N^{1-\frac{1}{d}})$.

Theorem 2.1.6. *Let $\{e_1, \cdots, e_N\} \subset [-\frac{1}{2}, \frac{1}{2}]^d$ with $d \geq 3$. Then, there exists a permutation p of $\{1, 2, \cdots, N\}$ such that*

$$\sum_{i=1}^{N-1} \|e_{p(i)} - e_{p(i+1)}\| \leq 2\sqrt{d+3}N^{1-\frac{1}{d}} - 2\sqrt{d+3}. \tag{2.7}$$

Note that for a unit-norm frame $F = \{e_1, \cdots, e_N\}$ for $\mathbb{R}^d$, the set $\frac{1}{2}F = \{\frac{1}{2}e_1, \cdots, \frac{1}{2}e_N\}$ is a subset of $[-\frac{1}{2}, \frac{1}{2}]^d$. In [10], the authors combined these results to obtain the following result.

Theorem 2.1.7. *Let $F = \{e_1, \cdots, e_N\}$ be a unit-norm frame for $\mathbb{R}^d$, $d \geq 3$. Then, there exists a*

permutation p of $\{1, 2, \cdots, N\}$ such that

$$\sigma(F, p) \leq 4\sqrt{d+3}N^{1-\frac{1}{d}} - 4\sqrt{d+3}. \tag{2.8}$$

While Theorem 2.1.7 provides a general bound on frame variation for finite unit-norm frames, there are specific types of frames where the resulting frame variations can be uniformly bounded by a constant which is independent of N. Here, we introduce the harmonic frames, which is a well-known example of FUNTFs achieving uniformly bounded frame variation.

Definition 2.1.8. *For $N \geq d \geq 2$, the harmonic frame $H_N^d = \{e_j\}_{j=0}^{N-1}$ in $\mathbb{R}^d$ is defined as*

$$e_j = \sqrt{\frac{2}{d}}\Big[\cos\frac{2\pi j}{N}, \sin\frac{2\pi j}{N}, \cos\frac{2\pi 2j}{N}, \sin\frac{2\pi 2j}{N}, \cdots, \cos\frac{2\pi\frac{d}{2}j}{N}, \sin\frac{2\pi\frac{d}{2}j}{N}\Big]$$

when d is even, and

$$e_j = \sqrt{\frac{2}{d}}\Big[\frac{1}{\sqrt{2}}, \cos\frac{2\pi j}{N}, \sin\frac{2\pi j}{N}, \cos\frac{2\pi 2j}{N}, \sin\frac{2\pi 2j}{N}, \cdots, \cos\frac{2\pi\frac{d-1}{2}j}{N}, \sin\frac{2\pi\frac{d-1}{2}j}{N}\Big]$$

when d is odd.

It is easy to check that the harmonic frames are FUNTFs. Moreover, harmonic frames achieve uniformly bounded frame variation for identity permutation.

Lemma 2.1.9. [12] *Let $H_N^d = \{e_j\}_{j=0}^{N-1}$ be the harmonic frame in $\mathbb{R}^d$ with $N \geq d \geq 2$. Let p be the identity permutation. Then we have*

$$\sigma(H_N^d, p) \leq \frac{2\pi(d+1)}{\sqrt{3}}.$$

For more details on finite frames with uniformly bounded frame variation, we refer to [14].

2.2 Quantization for Finite Frames

Let $F = \{e_1, \cdots, e_N\}$ be a finite frame in $\mathbb{R}^d$. Let S be the frame operator of F. Then any $x \in \mathbb{R}^d$ can be represented as a linear representation using frame coefficients and the canonical dual frame of F:

$$x = \sum_{i=1}^{N} \langle x, e_i \rangle S^{-1} e_i.$$

One can say that this frame expansion discretely encodes x by the frame coefficients $\{\langle x, e_i \rangle\}_{i=1,\cdots,N}$. Quantization refers to replacing the frame coefficients by elements from a fixed finite set, which is often referred to as the *quantization alphabet*.

We begin with some common definitions of quantization alphabet in [10–12, 16, 134]. Given $K \in \mathbb{N}$ and $\delta > 0$, the *midrise quantization alphabet* A_K^δ is defined as

$$A_K^\delta = \Big\{(-K + \frac{1}{2})\delta, (-K + \frac{3}{2})\delta, \cdots, -\frac{1}{2}\delta, \frac{1}{2}\delta, \cdots, (K - \frac{1}{2})\delta\Big\}. \tag{2.9}$$

The $2K$*-level midrise uniform scalar quantizer* Q with step size δ is defined as

$$Q(u) = \arg\min_{q \in A_K^\delta} |u - q|.$$

Therefore, $Q(u)$ is the element in A_K^δ that is closest to u.

Memoryless scalar quantization (MSQ) is a classical method in quantization, which replaces the frame coefficients by $\{Q(\langle x, e_i \rangle)\}_{i=1,\cdots,N}$. This is simple and fast, but one cannot enjoy the benefit of the redundancy in frames in the reconstruction of signals.

On the other hand, *Sigma-Delta ($\Sigma\Delta$) quantization* methods [10–12, 28, 43, 58, 85, 111] are more robust algorithms that exploit the redundancy in frames. $\Sigma\Delta$ quantization methods are iterative algorithms that compensate the quantization errors from the previous steps. Here, we focus on the first-order $\Sigma\Delta$ quantization method. For general rth order $\Sigma\Delta$ quantization, we refer to Chapter 8 in [16].

Definition 2.2.1. *Given $K \in \mathbb{N}$ and $\delta > 0$, let the midrise quantization alphabet be A_K^δ, the $2K$-level midrise uniform scalar quantizer with stepsize δ be Q, and p be a permutation of $\{1, 2, \cdots, N\}$. For a given input sequence $\{x_1, x_2, \cdots, x_N\}$, the associated first-order* $\Sigma\Delta$ quantization *is defined by the iteration:*

$$u_n = u_{n-1} + x_{p(n)} - q_n,$$

$$q_n = Q(u_{n-1} + x_{p(n)}),$$

for $n = 1, 2, \cdots, N$ where $u_0 = 0$. This produces the quantized sequence $\{q_1, \cdots, q_N\}$ and an auxiliary sequence $\{u_0, \cdots, u_N\}$ of state variables.

It is possible to consider nonzero initial condition $\|u_0\| < \delta/2$, but for simplicity, we will only consider the case $u_0 = 0$ as in [12]. This first-order $\Sigma\Delta$ system is stable, in the sense that the state variables are uniformly bounded if the input sequence is bounded.

Lemma 2.2.2. *If the input sequence $\{x_1, \cdots, x_N\}$ in Definition 2.2.1 satisfies $|x_i| \leq (K - 1/2)\delta$ for $i = 1, \cdots, N$, then $|u_i| \leq \delta/2$ for $i = 0, 1, \cdots, N$.*

Proof. We proceed by induction. Since $u_0 = 0$, we have $|u_1| = |x_{p(1)} - Q(x_{p(1)})| \leq \delta/2$. The base

case holds. Now suppose $|u_i| \leq \delta/2$ for some $i \leq N-1$. Then,

$$|u_i + x_{p(i+1)}| \leq |u_i| + |x_{p(i+1)}| \leq \delta/2 + (K - 1/2)\delta = K\delta.$$

Therefore, by the definition of A_K^δ and the uniform scalar quantizer Q, we get

$$|u_{i+1}| = |u_i + x_{p(i+1)} - q_{i+1}| = |u_i + x_{p(i+1)} - Q(u_i + x_{p(i+1)})| \leq \delta/2.$$

□

Note that the first-order $\Sigma\Delta$ quantization method in Definition 2.2.1 does not involve any frame. The first-order $\Sigma\Delta$ quantization method for finite frames operates by taking input sequence as frame coefficients $\{\langle x, e_i\rangle\}_{i=1,\cdots,N}$.

Definition 2.2.3. *Let $F = \{e_1, \cdots, e_N\}$ be a finite frame for $\mathbb{R}^d$ where $d \geq 3$ and let p be a permutation of $\{1, 2, \cdots, N\}$. Choose $x \in \mathbb{R}^d$ with frame expansion $x = \sum_{i=1}^N x_i S^{-1} e_i$ where $x_i = \langle x, e_i\rangle$, and S be the frame operator. Then, the quantized expansion $\bar{x}$ of x is defined as:*

$$\bar{x} = \sum_{i=1}^{N} q_i S^{-1} e_{p(i)}, \tag{2.10}$$

where $\{q_1, \cdots, q_N\}$ are the quantized sequence from the first-order $\Sigma\Delta$ quantization in 2.2.1.

We provide now an error bound on the approximation error $\|x - \bar{x}\|$. Here, we only state the result for the case of a FUNTF. For general results, see [12] and [10].

Theorem 2.2.4. *Let $F = \{e_1, \cdots, e_N\}$ be a finite unit-norm tight frame for $\mathbb{R}^d$, and let p be a permutation of $\{1, 2, \cdots, N\}$. Let $x \in \mathbb{R}^d$ satisfy $\|x\| \leq (K-1/2)\delta$ and have the frame expansion $x =$*

$\sum_{i=1}^{N}\langle x, e_i\rangle S^{-1}e_i$, *where* S^{-1} *is the inverse frame operator for* F. *Then,* $\bar{x}$ *satisfies the approximation error*

$$\|x - \bar{x}\| \leq \frac{\delta d}{2N}(\sigma(F,p) + 1).$$

Proof. Since $\|x\| \leq (K - 1/2)\delta$, we have $|x_i| = |\langle x, e_i\rangle| \leq \|x\| \leq (K - 1/2)\delta$. Therefore by Lemma 2.2.2, we have $|u_i| \leq \delta/2$ for $i = 0, \cdots, N$. Since F is a FUNTF, $S = \frac{N}{d} I_d$ holds and

$$x = \frac{d}{N}\sum_{i=1}^{N} x_i e_i = \frac{d}{N}\sum_{i=1}^{N} x_{p(i)} e_{p(i)}, \quad \bar{x} = \sum_{i=1}^{N} q_i S^{-1} e_{p(i)} = \frac{d}{N}\sum_{i=1}^{N} q_i e_{p(i)}.$$

Therefore, the approximation error can be bounded as

$$\begin{aligned}
\|x - \bar{x}\| &= \frac{d}{N}\|\sum_{i=1}^{N}(x_{p(i)} - q_i)e_{p(i)}\| = \frac{d}{N}\|\sum_{i=1}^{N}(u_i - u_{i-1})e_{p(i)}\| \\
&= \frac{d}{N}\|\sum_{i=1}^{N-1} u_i(e_{p(i)} - e_{p(i+1)}) + u_N e_{p(N)}\| \\
&\leq \frac{d}{N}\Big(\sum_{i=1}^{N-1}|u_i|\|e_{p(i)} - e_{p(i+1)}\| + |u_N|\|e_{p(N)}\|\Big) \\
&\leq \frac{d}{N}\Big(\sum_{i=1}^{N-1}\frac{\delta}{2}\|e_{p(i)} - e_{p(i+1)}\| + \frac{\delta}{2}\|e_{p(N)}\|\Big) \\
&= \frac{\delta d}{2N}(\sigma(F,p) + 1).
\end{aligned}$$

□

From the approximation error bound, one can see that it is desirable to choose a permutation p that makes the frame variation $\sigma(F,p)$ small. Combining Theorem 2.1.7 and Theorem 2.2.4, we obtain the following corollary, where the approximation error bound only depends on δ, d, and N.

Corollary 2.2.5. *Let $F = \{e_1, \cdots, e_N\}$ be a finite unit-norm tight frame for $\mathbb{R}^d$, $d \geq 3$, and let $x \in \mathbb{R}^d$ satisfy $\|x\| \leq (K - 1/2)\delta$. Let p be a permutation of $\{1, 2, \cdots, N\}$ that satisfies (2.8). Let $\bar{x}$ be the quantized expansion. Then, the approximation error satisfies*

$$\|x - \bar{x}\| \leq \frac{\delta d}{2N}(4\sqrt{d+3}N^{1-\frac{1}{d}} - 4\sqrt{d+3} + 1).$$

Corollary 2.2.5 shows that we can bound the approximation error by only using the variables δ, d, and N. We use this result to obtain error bounds between a neural network and its quantized version in Section 3.3.

For some specific FUNTFs, such as harmonic frames, their frame variations can be uniformly bounded by, independent of N. Therefore, they result in a tighter approximation bound $C_d\delta/N$, where C_d is a constant that depends only on the dimension d. For example, when we use harmonic frames with identity permutation, then the approximation error under the first-order $\Sigma\Delta$ quantization method can be bounded as

$$\|x - \bar{x}\| \leq \frac{\delta d}{2N}\Big(\frac{2\pi(d+1)}{\sqrt{3}} + 1\Big).$$

2.3 Manifolds

In this section, we briefly review some basic facts in manifolds. For a more detailed background knowledge, we refer to [61]. Let $\mathcal{M}$ be a d-dimensional compact Riemannian manifold isometrically embedded in $\mathbb{R}^D$.

Definition 2.3.1. *We say that (U, ϕ) is a chart on $\mathcal{M}$ if $U \subset \mathcal{M}$ is open and $\phi : U \to \mathbb{R}^d$ is a homeomorphism.*

Definition 2.3.2. *We say that two charts (U, ϕ) and (V, ψ) on $\mathcal{M}$ are C^k compatible if and only if the two maps $\phi \circ \psi^{-1} : \psi(U \cap V) \to \phi(U \cap V)$ and $\psi \circ \phi^{-1} : \phi(U \cap V) \to \psi(U \cap V)$ are both C^k.*

Definition 2.3.3. *A C^k atlas for $\mathcal{M}$ is a collection of pairwise C^k compatible charts $\{(U_i, \phi_i)\}_{i \in \mathcal{I}}$ such that $\cup_{i \in \mathcal{I}} U_i = \mathcal{M}$. A manifold is smooth if it has a C^∞ atlas.*

Now we define differentiable functions on $\mathcal{M}$.

Definition 2.3.4. *A function $f : \mathcal{M} \to \mathbb{R}$ is C^s if for any chart (U, ϕ), the composition $f \circ \phi^{-1} : \phi(U) \to \mathbb{R}$ is s-times continuously differentiable.*

We remark that the definition of C^s functions does not depend on the choice of the chart (U, ϕ). Suppose (V, ψ) is another chart on $\mathcal{M}$ with $U \cap V \neq \emptyset$. Then we have $f \circ \psi^{-1} = (f \circ \phi^{-1}) \circ (\phi \circ \psi^{-1})$. Since M is smooth, (U, ϕ) and (V, ψ) are C^∞ compatible. Therefore, $f \circ \psi^{-1}$ is also C^s function.

Next, we introduce *reach* of a manifold, which is a geometric concept that extends the concept of curvature to manifolds.

Definition 2.3.5. *Let $S(\mathcal{M}) = \left\{x \in \mathbb{R}^D : \exists p \neq q \in \mathcal{M}, \|x - p\| = \|x - q\| = \inf_{y \in \mathcal{M}} \|y - x\|\right\}$, which is the set of points that have at least two nearest neighbors in $\mathcal{M}$. The reach $\tau > 0$ is defined as*

$$\tau = \inf_{x \in \mathcal{M}, y \in S(\mathcal{M})} \|x - y\|.$$

Finally, we introduce the partition of unity.

Definition 2.3.6. *A C^∞ partition of unity on $\mathcal{M}$ is a collection of nonnegative C^∞ functions $\rho_i : \mathcal{M} \to \mathbb{R}_{\geq 0}$ for all $i \in \mathcal{A}$, such that*

1. *supp(ρ_i) is locally finite,*

2. $\sum_{i\in\mathcal{A}} \rho_i = 1$.

For a smooth manifold, a C^∞ partition of unity always exists.

Proposition 2.3.7. *Let $\{U_i\}_{i\in\mathcal{I}}$ be an open cover of a compact smooth manifold $\mathcal{M}$. Then, there exists a C^∞ partition of unity $\{\rho_i\}_{i\in\mathcal{I}}$ where every ρ_i has a compact support which is a subset of U_i.*

2.4 Neural Networks

In this section, we introduce the mathematical formulation of neural networks.

First, a fully connected *feed-forward neural network* (FNN) with n layers is a function $f : \mathbb{R}^{m_0} \to \mathbb{R}^{m_n}$ which acts on data $x \in \mathbb{R}^{m_0}$ via

$$f(x) = h^{[n]} \circ \sigma \circ h^{[n-1]} \circ \cdots \circ \sigma \circ h^{[1]}(x), \tag{2.11}$$

where $h^{[l]}(x) = W_l x + b_l$ with *weight matrix* $W_l \in \mathbb{R}^{m_l \times m_{l-1}}$ and *bias* $b_l \in \mathbb{R}^{m_l}$, $l = 1, 2, \cdots, n$, and nonlinear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ which acts on each component of a vector. Here, we omit the activation for the last layer because the choice of the final activation can be different from σ. For example in classification problems, softmax activation function is used in the last step, while σ is often chosen as the *Rectified Linear Unit* (ReLU) function, defined as $x \mapsto \max\{0, x\}$ for $x \in \mathbb{R}$.

Next, we introduce the mathematical formulation for residual blocks and residual neural networks. We adopt a similar mathematical formulation from Chapter 3.8.2 in [40]. A *residual neural network* with n residual blocks is a function $g : \mathbb{R}^k \to \mathbb{R}^k$ defined as

$$g(x) = z^{[n]} \circ \sigma \circ z^{[n-1]} \circ \cdots \circ \sigma \circ z^{[1]}(x), \tag{2.12}$$

where each *residual block* $z^{[i]}$ has a structure $z^{[i]}(x) = W_{i,2}\sigma(W_{i,1}x + b_i) + x$, with weight matrices $W_{i,1} \in \mathbb{R}^{r_i \times k}, W_{i,2} \in \mathbb{R}^{k \times r_i}$ and bias $b_i \in \mathbb{R}^{r_i}$. For residual networks, we only consider ReLU as the activation function.

While various neural network architectures exist, such as transformers and graph neural networks, we restrict our focus to feedforward neural networks and residual neural networks, as they provide a representative framework for the study of deep learning theory. We further remark that the definitions introduced below regarding quantized neural networks can be naturally generalized to other types of neural network architectures.

For a neural network $F(x)$, the *quantized neural network* $F_Q(x)$ is defined as a network formed by replacing weight matrices and biases of $F(x)$ with quantized weight matrices and quantized biases from any quantization algorithm. For example, for a fully connected FNN $f(x)$ in (2.11), assume that Q_i's and c_i's are quantized weight matrices and quantized biases of W_i's and b_i's, respectively. Then,

$$f_Q(x) = h_Q^{[n]} \circ \sigma \circ h_Q^{[n-1]} \circ \sigma \cdots \circ \sigma \circ h_Q^{[1]}(x), \tag{2.13}$$

where $h_Q^{[i]}(x) = Q_i x + c_i$ for $i = 1, \cdots, n$. Similarly for a residual network $g(x)$ in (2.12),

$$g_Q(x) = z_Q^{[n]} \circ \sigma \circ z_Q^{[n-1]} \circ \cdots \circ \sigma \circ z_Q^{[1]}(x), \tag{2.14}$$

where $z_Q^{[i]}(x) = Q_{i,2}\sigma(Q_{i,1}x + c_i) + x$ for $i = 1, \cdots, n$. Here, $Q_{i,2}$'s, $Q_{i,1}$'s and c_i's are quantized weight matrices and quantized biases of $W_{i,2}$'s, $W_{i,1}$'s and b_i's respectively. In practice, bias vectors are not often quantized. In this case, we simply keep the bias vectors.

Note that if we write $\widetilde{W_l} = (W_l, b_l)$ and $\tilde{x} = (x^\top, 1)^\top$, then $h^{[l]}(x) = W_l x + b_l = \widetilde{W_l}\tilde{x}$ holds.

Therefore, without loss of generality, we will omit the biases in our analysis. This means we assume $b_i = 0$ for all i's, so $h^{[i]}(x)$'s in (2.11) can be regarded as $h^{[i]}(x) = W_i x$ and $z^{[i]}(x)$'s in (2.12) can be regarded as $z^{[i]}(x) = W_{i,2}\sigma(W_{i,1}x) + x$.

We introduce some definitions that is frequently used in Chapter 4.

Definition 2.4.1. *A FNN with n layers in the form of (2.11) is called a strict neural network. If the same activation is applied at the last layer, say*

$$f(x) = \sigma \circ h^{[n]} \circ \sigma \circ h^{[n-1]} \circ \cdots \circ \sigma \circ h^{[1]}(x),$$

then we say f is an activated neural network.

Note that from the definition, the only difference between a strict neural network and an activated neural network is whether the activation function σ is applied at the final layer.

We say that a strict neural network (2.11) has L layers, $N = \sum_{i=1}^{L} m_i$ nodes, and $P = \sum_{i=1}^{L} \|W_l\|_0 + \|b_i\|_0$ parameters. If L, N, P are all less or equal to some quantity T, then we say that such a neural network has size at most T. We say two FNNs f_1 and f_2 have the same structure if (1) they have the same number of layers, (2) they use the same activation σ, and (3) the affine maps in the ith layer have the same size of weight matrix and bias vector, for all $i = 1, \cdots, L$ where L is the number of layers. The same definition is made for activated neural networks because the only difference is the activation applied at the final layer.

When a strict (activated) neural network has all nonzero entries in its weight matrices and its bias vectors belong to $\{\pm 1\}$, we say that the network is a strict (activated) $\{\pm 1\}$ *quantized neural network* or a strict (activated) *one-bit neural network*.

2.5 Mean-field Parametrization of Neural Networks and Wasserstein Gradient Flows

A mean-field parametrization of two-layer (shallow) neural network is

$$f(x;\theta) = \frac{1}{m}\sum_{i=1}^{m} a_j \sigma(w_i^\top x + b_i)$$

where σ is activation, and $\theta = \{(a_i, w_i, b_i) : i = 1, \cdots, m\}$ represents the parameters of the network.

Note that such f in the mean-field parametrization admits an integral representation:

$$f(x) = \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} a\sigma(w^\top x + b)\pi(da \otimes dw \otimes db)$$

for a probability measure $\pi = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\delta}_{(a_i,w_i,b_i)}$.

The study of Wasserstein gradient flows in neural network training is motivated by the observation that, under mean-field parametrization, the evolution of neural network parameters under time-accelerated gradient flow can be equivalently described by the evolution of their distribution via the 2-Wasserstein gradient flow. This framework not only characterizes the training dynamics of two-layer neural networks with a finite number of neurons due to the aforementioned equivalence, but also provides the advantage of describing the training process for infinite-width shallow neural networks. The application of Wasserstein gradient flows in the mean-field regime has been successful in analyzing the convergence of neural network training [22, 72, 76, 84, 94]. For a more detailed discussion on Wasserstein gradient flows in neural network training, see [22, 23, 120]. For a broader perspective on Wasserstein gradient flows and optimal transport, refer to [115].

2.6 Barron Spaces

Barron spaces, introduced in [34], generalize Barron's seminal work [7] in neural network approximation theory. They present a set of functions comprising shallow neural networks with mean-field parametrization. Consequently, analyzing the Wasserstein gradient flow is a suitable approach for studying the optimization dynamics of functions within Baron spaces.

A function $f : X \to \mathbb{R}$ is said to be a Barron function if it admits an integral representation of the form

$$f(x) = \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} a\sigma(w^\top x + b)\pi(da \otimes dw \otimes db), \quad x \in X, \tag{2.15}$$

for some Borel probability measure π on $\mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$, and if its Barron norm, as defined in [34], is finite. Here, $\sigma : \mathbb{R} \to \mathbb{R}$ denotes an activation function. We introduce the Barron norms for ReLU activation and for other Lipschitz continuous function in Section 5.2.

Barron functions can be approximated by finite-width two-layer neural networks with a dimension-independent approximation rate in terms of the number of parameters, given various choices of activation functions. For example, [34] demonstrated that for the ReLU activation, these functions exhibit both dimension-independent approximation rates and low statistical complexity. These favorable properties extend to Barron functions with certain other activation functions [65].

Both finite-width and infinite-width two-layer networks in the mean-field scaling can be expressed in the integral form given in (2.15). This integral representation facilitates the study of parameter evolution under gradient flow by leveraging the 2-Wasserstein gradient flow in the parameter distribution, as discussed in [120, 121]. For a more comprehensive discussion on Barron spaces, see [34, 36, 37, 65].

In this work, we focus on Barron functions defined by (2.15) equipped with the finite Barron norms adopted from [34, 36, 37, 65]. Crucially, these norms are well-defined for only Lipschitz continuous activation functions, and therefore our primary analysis of Barron spaces is restricted to this class. For non-Lipschitz continuous activations, where these norms may be incompatible with our tools using 2-Wasserstein gradient flows, we focus instead on finite-width networks, which remains a realistic and important setting. We defer the detailed discussion to Section 5.5 and Section 5.5.3. For further literature on Barron spaces for general activation functions, including approximation rates and embedding properties, see [50, 105–107].

Chapter 3: Frame Quantization of Neural Networks

3.1 Introduction

Quantization is the process of compressing input from a continuous or large set of values into a small-sized discrete set. It gained popularity in *signal processing*, where one of its primary goals is obtaining a condensed representation of the analogue signal suitable for digital storage and recovery. Examples of quantization algorithms include truncated binary expansion, memoryless scalar quantization, and $\Sigma\Delta$ quantization. Among them, $\Sigma\Delta$ quantization methods stand out due to their theoretically guaranteed robustness. Mathematical foundations were developed in several seminal works [10–12, 28, 43], and have been carefully studied since, e.g., [44, 45, 59, 96].

In recent years, the concept of quantization captured the attention of the machine learning community. The quantization of *deep neural networks* (DNNs) is considered one of the most effective network compression techniques [38]. Computers express parameters of a neural network as 32-bit or 64-bit floating point numbers. In *neural network quantization*, one tries to replace these parameters using compact formats such as 8 bits (or lower), while preserving the architecture and performance of the network. An effective neural network quantization method enables users to run DNNs on portable devices, such as smartphones, without relying on external servers. This benefits storage requirements and helps avoid privacy-related issues. Due to these reasons, there have been numerous efforts to develop neural network quantization schemes that preserve the model accuracy.

Neural network quantization can be categorized into 2 classes. The first class is *Quantization-Aware-Training* (QAT). QAT methods [25, 54, 60, 69, 93, 130] retrain the given neural network, by restricting the domain of parameters to a finite set of alphabets. The second class is *Post-Training Quantization* (PTQ). PTQ methods [6, 47, 74, 75, 82, 83, 119, 125, 133, 134] take a pre-trained neural

network and convert it directly into a fixed-point neural network. They demand less computation because they do not require end-to-end training. In addition, unlike QAT methods, PTQ methods do not require the training dataset - they usually require only a small calibration set. These reasons make PTQ methods attractive. For a detailed explanation of QAT and PTQ methods, see [38] and references therein.

Despite the popularity of PTQ methods, unfortunately, most of them lack theoretical error analysis. While some algorithms [74,75,133,134] are equipped with error estimates, they are typically probabilistic estimates with additional requirements on input distributions. Moreover, their error estimates are proved only for feed-forward networks. However, there are popular neural networks with different architectures, such as ResNet [49], which is constructed by stacking together a series of residual blocks. Therefore, the design of a PTQ method with error estimates for different types of neural networks is still an open question.

To address this problem, we return to the methods of quantization from signal processing and investigate how to utilize them in the neural network setting while retaining their theoretical guarantees. We focus on $\Sigma\Delta$ quantization algorithms. While [28, 43] focus on $\Sigma\Delta$ quantization for bandlimited functions, [10–12] provide mathematical analysis of $\Sigma\Delta$ quantization for finite frames in $\mathbb{R}^d$ and $\mathbb{C}^d$, which is suitable for quantization of weight matrices in neural networks. Leveraging the tools from [10–12], we are able to provide a new PTQ algorithm with error estimates. Our contributions are threefold:

1. In Section 3.2 we propose an algorithm for quantizing layers of a pre-trained neural network using the first-order $\Sigma\Delta$ quantization algorithm for finite unit-norm tight frames in $\mathbb{R}^d$. Our algorithm does not require any training data, hence a data-free quantization. To the best of our

knowledge, our work is the first to utilize the tools from frame theory in this context (recent work [132] studies quantizing random Fourier features (RFFs) using $\Sigma\Delta$ algorithm, which is different from our study of DNNs).

2. We provide error estimates for both n-layer feed-forward neural networks and neural networks formed by a series of residual blocks in Section 3.3. These results demonstrate how to control the error using the step size and the number of frame elements.

3. Finally, in Section 3.4, we present numerical results for quantizing neural networks by applying our proposed algorithm. We apply the algorithm to several neural network architectures on the MNIST and CIFAR-10 classification tasks, empirically demonstrating its effectiveness. In addition, we numerically validate the theoretical error bounds, established in Section 3.3, with toy examples. We also provide numerical experiments and detailed explanations for the most extreme case, 1-bit quantization.

We note that this chapter is based on the published paper [27]. The author of this thesis conducted the theoretical developments, including all theorems and proofs, performed the numerical experiments, and managed the editorial process under the guidance of Professor Wojciech Czaja.

3.2 Frame Quantization for Neural Networks

In this section, we explain our method to quantize a weight matrix $W_i \in \mathbb{R}^{m_i \times m_{i-1}}$. Let $F_i = \{e_1^i, \cdots, e_{N_i}^i\}$ be a FUNTF for $\mathbb{R}^{m_i}$ that we are using for quantization. Write $W_i = [w_1^i, w_2^i, \cdots, w_{m_{i-1}}^i]$ where w_j^i is a $m_i \times 1$ column vector for $j = 1, 2, \cdots, m_{i-1}$. First, choose appropriate positive integer

K_i and step size $\delta_i > 0$ that satisfy

$$\max_{j=1,\cdots,m_{i-1}} \|w_j^i\| \leq (K_i - \frac{1}{2})\delta_i. \tag{3.1}$$

Next, we find a permutation p_i of $\{1, \cdots, N_i\}$ that satisfies (2.8). The algorithm to find such permutations is described in [10, 118], so we omit the details. Then, we compute the quantized expansions of w_j^i's, using frame F_i, constant K_i, and step size δ_i for $i = 1, \cdots, n$. Let q_j^i be the quantized expansion of w_j^i. Since we are using finite unit-norm tight frame, the dual frame operator S^{-1} is $\frac{m_i}{N_i} I_{m_i}$, where I_{m_i} is the identity matrix of $\mathbb{R}^{m_i}$. Therefore q_j^i's can be written as

$$q_j^i = \sum_{k=1}^{N_i} q_{j,k}^i S^{-1} e_{p_i(k)}^i = \frac{m_i}{N_i} \sum_{k=1}^{N_i} q_{j,k}^i e_{p_i(k)}^i. \tag{3.2}$$

Note that the first-order $\Sigma\Delta$ quantization forces $q_{j,k}^i \in A_{K_i}^{\delta_i}$ for all $(j,k) \in \{1, \cdots, m_{i-1}\} \times \{1, \cdots, N_i\}$. Therefore, at most $2K_i$ values are candidates for $q_{j,k}^i$. In our scenario, we store $C_i = [q_{j,k}^i]_{1 \leq j \leq m_{i-1}, 1 \leq k \leq N_i}$, which is an element in $(A_{K_i}^{\delta_i})^{m_{i-1} \times N_i}$. When we need a quantized neural network, then we use the finite unit-norm tight frames F_i's and matrices C_i's to reconstruct a network using (3.2).

Current neural network quantization methods convert a weight matrix $W = [w_{ij}]_{1 \leq i \leq m, 1 \leq j \leq n} \in \mathbb{R}^{m \times n}$ into a quantized matrix $Q = [q_{ij}]_{1 \leq i \leq m, 1 \leq j \leq n}$, where each q_{ij} uses fewer bits compared to the corresponding w_{ij}. Note that W and Q have the same dimensions. Our method differs from typical approaches because we are storing matrices C_i's that have different dimensions from the weight matrices W_i's. Due to their different dimensions, we will not say C_i's are quantized weights. Instead, we will say that the matrix $Q_i = [q_1^i, \cdots, q_{m_{i-1}}^i] \in \mathbb{R}^{m_i \times m_{i-1}}$, where the columns are the quantized expressions q_j^i's, is the *quantized weight matrix* of $W_i \in \mathbb{R}^{m_i \times m_{i-1}}$ for $i = 1, \cdots, n$.

Algorithm 1 Frame Quantization

Require: Weight matrix $W_i \in \mathbb{R}^{m_i \times m_{i-1}}$ and FUNTF $F_i = \{e_1^i, \cdots, e_{N_i}^i\}$ for $\mathbb{R}^{m_i}$.

1. Set level $K_i \in \mathbb{N}$ and stepsize δ_i that satisfy $\max_{j=1,\cdots,m_{i-1}} \|w_j^i\| \leq (K_i - \frac{1}{2})\delta_i$.
2. Find a permutation p_i of $\{1, \cdots, N_i\}$ that satisfies (2.8).

for $j = 1, \cdots, m_{i-1}$ **do**

1. Compute frame expansion $w_j^i = \frac{m_i}{N_i} \sum_{k=1}^{N_i} w_{j,k}^i e_k^i$.
2. Use first-order $\Sigma\Delta$ quantization algorithm 2.2.1 with K_i and δ_i to compute

$$q_j^i = \frac{m_i}{N_i} \sum_{k=1}^{N_i} q_{j,k}^i e_{p_i(k)}^i.$$

end for

3. Set $C_i = [q_{j,k}^i]_{1 \leq j \leq m_{i-1}, 1 \leq k \leq N_i} \in (A_{K_i}^{\delta_i})^{m_{i-1} \times N_i}$.

Ensure: Quantized matrix $Q_i = [q_1^i, \cdots, q_{m_{i-1}}^i] \in \mathbb{R}^{m_i \times m_{i-1}}$. Store C_i.

Note that the above algorithm uses first-order $\Sigma\Delta$ quantization for the column vectors of the weight matrix W_i. On the other hand, we may consider applying first-order $\Sigma\Delta$ quantization algorithm for the row vectors of the weight matrix W_i. In this case, we apply Algorithm 1 to $W_i^\top$. This framework is applied for neural networks with residual blocks in Section 3.3.2.

Although biases are not typically quantized in practice, if quantization of biases is needed for some purpose, we may also quantize biases by adding them to the final columns of weight matrices. This means that we can also quantize b_i's while we quantize the weight matrices W_i's by applying Algorithm 1 to matrices $\widetilde{W_i} = (W_i, b_i)$'s.

3.3 Error Estimates

Throughout the remainder of this chapter, we assume that the activation function σ is a Lipschitz continuous function with Lipschitz constant L, and that $\sigma(0) = 0$. This is a natural assumption, as many activation functions used in practice, such as ReLU, satisfy this condition. In addition, we assume that $m_i \geq 3$ for all i's. This is also a reasonable assumption, since typical neural network

architectures have more than two neurons in each layer.

We begin with some auxiliary lemmas, which play an important role in proving error estimates between a neural network and its approximation by a quantized network.

Lemma 3.3.1. *Let Q_i be the quantized matrix for weight matrix $W_i \in \mathbb{R}^{m_i \times m_{i-1}}$ using Algorithm 1. Then,*

$$\|W_i - Q_i\| \leq 2\sqrt{2}\delta_i m_i \sqrt{m_{i-1} m_i} N_i^{-\frac{1}{m_i}},$$

where δ_i is the step size and N_i is the number of elements in the frame $F_i = \{e_1^i, \cdots, e_{N_i}^i\}$ used in Algorithm 1.

Proof. Write $W_i = [w_1^i \cdots w_{m_{i-1}}^i], Q_i = [q_1^i, \cdots, q_{m_{i-1}}^i] \in \mathbb{R}^{m_i \times m_{i-1}}$ where w_j^i's and q_j^i's are $m_i \times 1$ column vectors. Since Algorithm 1 uses p_i that satisfies (2.8), Corollary 2.2.5 gives us the error estimate

$$\|w_j^i - q_j^i\| \leq \frac{\delta_i m_i}{2N_i}(4\sqrt{m_i + 3} N_i^{1-\frac{1}{m_i}} - 4\sqrt{m_i + 3} + 1) \tag{3.3}$$

for all $j = 1, \cdots, m_{i-1}$. Now, using (3.3) and the assumption $m_i \geq 3$, we have

$$\begin{aligned}
\|W_i - Q_i\| = \max_{\|x\|=1} \|(W_i - Q_i)x\| &\leq \max_{\|x\|=1} \{\sum_{j=1}^{m_{i-1}} |x_j| \times \|w_j^i - q_j^i\|\} \\
&\leq \frac{\delta_i m_i}{2N_i}(4\sqrt{m_i + 3} N_i^{1-\frac{1}{m_i}} - 4\sqrt{m_i + 3} + 1) \times \max_{\|x\|=1} \{\sum_{j=1}^{m_{i-1}} |x_j|\} \\
&\leq \frac{\delta_i m_i \sqrt{m_{i-1}}}{2N_i}(4\sqrt{m_i + 3} N_i^{1-\frac{1}{m_i}} - 4\sqrt{m_i + 3} + 1) \\
&\leq \frac{\delta_i m_i \sqrt{m_{i-1}}}{2N_i} \times 4\sqrt{m_i + 3} N_i^{1-\frac{1}{m_i}} \leq 2\sqrt{2}\delta_i m_i \sqrt{m_{i-1} m_i} N_i^{-\frac{1}{m_i}}.
\end{aligned}$$

□

Next, we provide an upper bound for the norm of the quantized matrices Q_i's. This is used in deriving the error estimate between an n-layer FNN and its corresponding quantized neural network.

Lemma 3.3.2. *Let Q_i be the quantized matrix of W_i. Then we have*

$$\|Q_i\| \leq 2\sqrt{2}\delta_j m_j \sqrt{m_j m_{j-1}} N_j^{-\frac{1}{m_j}} + \|W_i\|.$$

Proof. Using Lemma 3.3.1 and the triangle inequality of matrix norms, we have

$$\|Q_i\| \leq \|Q_i - W_i\| + \|W_i\| \leq 2\sqrt{2}\delta_i m_i \sqrt{m_i m_{i-1}} N_i^{-\frac{1}{m_i}} + \|W_i\|.$$

□

3.3.1 Feedforward Networks

In this section, we derive an upper estimate for $\|f(x) - f_Q(x)\|$, where $f(x)$ is a FNN with n layers and $f_Q(x)$ its quantized neural network. For convenience, we adopt the assumptions from Section 2.4 and omit the biases in our analysis. Thus, we write $f(x) = W_n(\sigma(\cdots\sigma(W_1x)\cdots))$ and $f_Q(x) = Q_n(\sigma(\cdots\sigma(Q_1x)\cdots))$.

Theorem 3.3.3. *Let $f(x) = W_n(\sigma(\cdots\sigma(W_1x)\cdots))$ be a FNN with n layers. Denote the quantized neural network obtained by Algorithm 1 as $f_Q(x) = Q_n(\sigma(\cdots\sigma(Q_1x)\cdots))$. Then, for any input*

$X \in \mathbb{R}^{m_0}$, *we have*

$$\|f(X) - f_Q(X)\| \leq L^{n-1}\|X\| \times \sum_{j=1}^{n} \Big\{ 2\sqrt{2}\delta_j m_j \sqrt{m_j m_{j-1}} N_j^{-\frac{1}{m_j}} \times \prod_{i=j+1}^{n} \|W_i\| \times \prod_{l=1}^{j-1} (2\sqrt{2}\delta_l m_l \sqrt{m_l m_{l-1}} N_l^{-\frac{1}{m_l}} + \|W_l\|) \Big\}. \tag{3.4}$$

Proof. Note that since σ is an $L-$Lipschitz continuous function with $\sigma(0) = 0$, we have

$$\|\sigma(v)\| = \|\sigma(v) - \sigma(0)\| \leq L\|v - 0\| = L\|v\|$$

for any vector v. Using this, for $j = 1, \cdots, n$, we have

$$\begin{aligned}
&\|W_n\sigma \cdots W_j\sigma Q_{j-1}\sigma Q_{j-2} \cdots Q_1 X - W_n\sigma \cdots W_{j+1}\sigma Q_j\sigma Q_{j-1}\sigma Q_{j-2} \cdots Q_1 X\| \\
&\leq L^{n-j}\|W_n\| \cdots \|W_{j+1}\| \times \|W_j - Q_j\| \|\sigma Q_{j-1}\sigma Q_{j-2} \cdots Q_1 X\| \\
&\leq L^{n-1}\|W_n\| \cdots \|W_{j+1}\| \times \|W_j - Q_j\| \times \|Q_{j-1}\| \cdots \|Q_1\| \|X\|.
\end{aligned}$$

Next, using Lemma 3.3.1, Lemma 3.3.2, and the triangle inequality, we get

$$\begin{aligned}
&\|f(X) - f_Q(X)\| = \|W_n(\sigma(\cdots\sigma(W_1 x)\cdots)) - Q_n(\sigma(\cdots\sigma(Q_1 x)\cdots))\| \\
&\leq \sum_{j=1}^{n} \|W_n\sigma \cdots W_j\sigma Q_{j-1}\sigma Q_{j-2} \cdots Q_1 X - W_n\sigma \cdots W_{j+1}\sigma Q_j\sigma Q_{j-1}\sigma Q_{j-2} \cdots Q_1 X\| \\
&\leq \sum_{j=1}^{n} L^{n-1}\|W_n\| \cdots \|W_{j+1}\| \times \|W_j - Q_j\| \times \|Q_{j-1}\| \cdots \|Q_1\| \|X\| \\
&\leq L^{n-1}\|X\| \times \sum_{j=1}^{n} \Big\{ 2\sqrt{2}\delta_j m_j \sqrt{m_j m_{j-1}} N_j^{-\frac{1}{m_j}} \times \prod_{i=j+1}^{n} \|W_i\|
\end{aligned}$$

$$\times \prod_{l=1}^{j-1} (2\sqrt{2}\delta_l m_l \sqrt{m_l m_{l-1}} N_l^{-\frac{1}{m_l}} + \|W_l\|) \Big\}.$$

□

Note that $\{W_i\}_{i=1}^n$, $\{m_i\}_{i=0}^n$, and L are constants determined by the given FNN $f(x)$. Therefore, the upper bound in (3.4) can be only controlled by the step sizes δ_i and the numbers of frame elements N_i. Once we fix δ_i's, then the upper bound in (3.4) can be written as $\sum_{j=1}^n O(N_j^{-1/m_j})\|X\|$. This shows we can leverage the redundancy of frames for the accuracy of quantized neural networks. On the other hand, if we fix frames $F_1, \cdots, F_n$, then the upper bound in (3.4) depends only on δ_i's and can be written as $\sum_{i=1}^n O(\delta_i)\|X\|$. Therefore, smaller δ_i's and larger N_i's would generate a quantized neural network with higher accuracy.

When the number of neurons in each hidden layer is the same, that is, when $m_1 = \cdots = m_{n-1}$, then we have a simplified version of Theorem 3.3.3.

Corollary 3.3.4. *Let $f(x) = W_n(\sigma(\cdots\sigma(W_1x)\cdots))$ be a FNN with n layers with $m_1 = m_2 = \cdots = m_{n-1} = m$. Fix a FUNTF $F \subset \mathbb{R}^m$ with N elements and a stepsize δ. Choose a permutation p of $\{1, 2, \cdots, N\}$ that satisfies (2.8). Now, use Algorithm 1 to quantize $W_1, \cdots, W_{n-1}$ and $W_n^\top$ with the same F, δ, and p. Let $M = \max\{m_0, m, m_n\}$. Then, the quantized neural network $f_Q(x) = Q_n(\sigma(\cdots\sigma(Q_1x)\cdots))$ satisfies*

$$\|f(X) - f_Q(X)\| \le 2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} L^{n-1}\|X\| \sum_{j=1}^n \Big\{ \prod_{i=j+1}^n \|W_i\| \prod_{l=1}^{j-1} (2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} + \|W_l\|) \Big\} \quad (3.5)$$

for any data $X \in \mathbb{R}^{m_0}$.

Proof. By Lemma 3.3.1, we have $\|W_i - Q_i\| \le 2\sqrt{2}\delta_i m_i \sqrt{m_{i-1}m_i} N_i^{-\frac{1}{m_i}} \le 2\sqrt{2}\delta M^2 N^{-\frac{1}{m}}$ for

$i = 1, \cdots, n-1$. For $i = n$, we have $\|W_n - Q_n\| = \|W_n^\top - Q_n^\top\| \leq 2\sqrt{2}\delta_n m_{n-1}\sqrt{m_{n-1}m_n} N_n^{-\frac{1}{m_{n-1}}} \leq 2\sqrt{2}\delta M^2 N^{-\frac{1}{m}}$. These estimates yield $\|Q_i\| \leq \|W_i - Q_i\| + \|W_i\| \leq 2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} + \|W_i\|$. Therefore, from the proof of Theorem 3.3.3, we get

$$\begin{aligned}
\|f(X) - f_Q(X)\| &\leq L^{n-1}\|X\| \times \sum_{j=1}^{n} \Big\{ \prod_{i=j+1}^{n} \|W_i\| \times \|W_j - Q_j\| \times \|Q_{j-1}\| \cdots \|Q_1\| \Big\} \\
&\leq L^{n-1}\|X\| \sum_{j=1}^{n} \Big\{ 2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} \prod_{i=j+1}^{n} \|W_i\| \prod_{l=1}^{j-1} (2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} + \|W_l\|) \Big\} \\
&= 2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} L^{n-1}\|X\| \sum_{j=1}^{n} \Big\{ \prod_{i=j+1}^{n} \|W_i\| \prod_{l=1}^{j-1} (2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} + \|W_l\|) \Big\}.
\end{aligned}$$

□

If we use a FUNTF for $\mathbb{R}^m$ with N elements where $N \geq \left(\frac{2\sqrt{2}\delta M^2}{\min\{\|W_1\|, \cdots, \|W_n\|\}} \right)^m$, then we have the following simplification.

Corollary 3.3.5. *If we have* $N \geq \left(\frac{2\sqrt{2}\delta M^2}{\min\{\|W_1\|, \cdots, \|W_n\|\}} \right)^m$ *in Corollary 3.3.4, then*

$$\|f(X) - f_Q(X)\| \leq \sqrt{2}\delta M^2 N^{-\frac{1}{m}} L^{n-1}\|X\| \prod_{i=1}^{n} \|W_i\| \sum_{j=1}^{n} \frac{2^j}{\|W_j\|},$$

for any input data $X \in \mathbb{R}^{m_0}$.

Proof. $N \geq \left(\frac{2\sqrt{2}\delta M^2}{\min\{\|W_1\|, \cdots, \|W_n\|\}} \right)^m$ implies $2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} \leq \min\{\|W_1\|, \cdots, \|W_n\|\}$. Hence we have

$$\prod_{i=j+1}^{n} \|W_i\| \prod_{l=1}^{j-1} (2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} + \|W_l\|) \leq \prod_{i=j+1}^{n} \|W_i\| \prod_{l=1}^{j-1} (\|W_l\| + \|W_l\|) = \frac{2^{j-1}}{\|W_j\|} \prod_{i=1}^{n} \|W_i\|.$$

Applying this to (3.5) yields the result. □

Note that the above results apply to general FUNTFs. However, certain families of FUNTFs

which have uniformly bounded frame variations, provide better error estimates. For instance, using harmonic frames $\{H_{N_i}^{m_i}\}_{i=1}^n$ and the identity permutation to quantize weight matrices W_i's using Algorithm 1, then one can easily prove a variant of Lemma 3.3.1:

$$\|W_i - Q_i\| \leq \frac{\delta_i m_i \sqrt{m_{i-1}}}{2N_i}\Big(\frac{2\pi(m_i+1)}{\sqrt{3}} + 1\Big) \leq \frac{\delta_i m_i \sqrt{m_{i-1}}}{2N_i} \times \frac{8\pi + \sqrt{3}}{3\sqrt{3}} m_i.$$

The last inequality holds since we assume $m_i \geq 3$ for all i's. Using this, we can show (3.4) can be rewritten as an explicit upper bound for harmonic frames, which is a stronger upper bound in terms of N_i's:

$$\begin{aligned}\|f(X) - f_Q(X)\| \leq L^{n-1}\|X\| \times \sum_{j=1}^{n} \Big\{ \frac{(8\pi+\sqrt{3})}{6\sqrt{3}} \delta_j m_j \sqrt{m_j m_{j-1}} N_j^{-1} \times \prod_{i=j+1}^{n} \|W_i\| \\ \times \prod_{l=1}^{j-1} \big(\frac{(8\pi+\sqrt{3})}{6\sqrt{3}} \delta_l m_l \sqrt{m_l m_{l-1}} N_l^{-1} + \|W_l\| \big) \Big\}. \end{aligned} \tag{3.6}$$

That is, treating δ_i's as fixed constants, we obtain $\|f(X) - f_Q(X)\| \leq \sum_{i=1}^n O(N_i^{-1})\|X\|$. Similar results hold for FUNTFs where their frame variations are uniformly bounded. These frames may help us to construct a quantized network with higher accuracy.

3.3.2 Neural Networks with Residual Blocks

We now establish error estimates for residual networks. We follow the discussion in Section 2.4 and omit biases for simplicity. We quantize all the weight matrices using the same FUNTF F with N elements, the same permutation p of $\{1, 2, \cdots, N\}$ satisfying (2.8), the same constant K, and the same step size δ as in Algorithm 1. Since $W_{i,1}$ and $W_{i,2}$ have different structures, we apply Algorithm 1 to $W_{i,1}^\top$ and $W_{i,2}$ to compress the residual network. The quantized weight matrices $Q_{i,1}$'s are transposes

of those obtained by applying Algorithm 1 to $W_{i,1}$'s.

Theorem 3.3.6. *Let $g(x)$ be a residual neural network with n residual blocks (2.12). Let*

$$\lambda = \max_{i=1,\cdots,n} \Big\{ \max\{\|W_{i,2}\|, \|W_{i,1}\|\} \Big\}, \quad r = \max_{i=1,\cdots,n} r_i.$$

Let $g_Q(x)$ be the quantized residual neural network of $g(x)$ (2.14) from Algorithm 1. Then for any input $X \in \mathbb{R}^k$, we have

$$\begin{aligned} \|g(X) - g_Q(X)\| \leq\ & 4\delta k\sqrt{r(k+3)}(\delta k\sqrt{r(k+3)} + \lambda)N^{-\frac{1}{k}}\|X\| \\ & \times \sum_{j=0}^{n-1}(\lambda^2+1)^j\Big((2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}} + \lambda)^2 + 1\Big)^{n-1-j}. \end{aligned} \tag{3.7}$$

Proof. For simplicity, let $A = 4\delta k\sqrt{r(k+3)}(\delta k\sqrt{r(k+3)}+\lambda)N^{-\frac{1}{k}}$ and $B = (2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}}+\lambda)^2 + 1$. Let $y_0(x) = y_{0,Q}(x) = x$ and define $y_i(x) = z^{[i]} \circ \sigma \cdots \circ \sigma \circ z^{[1]}(x)$, and $y_{i,Q}(x) = z_Q^{[i]} \circ \sigma \cdots \circ \sigma \circ z_Q^{[1]}(x)$ for $i = 1, \cdots, n$. We shall show that

$$\|y_i(X) - y_{i,Q}(X)\| \leq A\|X\| \sum_{j=0}^{i-1}(\lambda^2+1)^j B^{i-1-j}, \tag{3.8}$$

for $i = 1, \cdots, n$. Note that we have recurrence relations:

$$y_i(X) = W_{i,2}(\sigma(W_{i,1}(\sigma(y_{i-1}(X))))) + \sigma(y_{i-1}(X)), \tag{3.9}$$

$$y_{i,Q}(X) = Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1,Q}(X))))) + \sigma(y_{i-1,Q}(X)). \tag{3.10}$$

Using the triangle inequality repeatedly, we obtain the following:

$$\begin{aligned}\|y_i(X) - y_{i,Q}(X)\| &\leq \|W_{i,2}(\sigma(W_{i,1}(\sigma(y_{i-1}(X))))) - Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1,Q}(X)))))\| \\ &\quad + \|\sigma(y_{i-1}(X)) - \sigma(y_{i-1,Q}(X))\| \\ &\leq \|W_{i,2}(\sigma(W_{i,1}(\sigma(y_{i-1}(X))))) - W_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X)))))\| \\ &\quad + \|W_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X))))) - Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X)))))\| \\ &\quad + \|Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X))))) - Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1,Q}(X)))))\| \\ &\quad + \|\sigma(y_{i-1}(X)) - \sigma(y_{i-1,Q}(X))\|. \end{aligned} \tag{3.11}$$

Since σ is ReLU, we have $\|\sigma(x)-\sigma(y)\| \leq \|x-y\|$ and $\|\sigma(x)\| \leq \|x\|$ for any vectors $x, y \in \mathbb{R}^k$. Using this fact, the first term in (3.11) is bounded by:

$$\begin{aligned}&\|W_{i,2}(\sigma(W_{i,1}(\sigma(y_{i-1}(X))))) - W_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X)))))\| \\ &\leq \|W_{i,2}\|\|W_{i,1} - Q_{i,1}\|\|y_{i-1}(X)\| \leq \lambda\|W_{i,1} - Q_{i,1}\|\|y_{i-1}(X)\|.\end{aligned}$$

The second term in (3.11) is bounded by:

$$\begin{aligned}&\|W_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X))))) - Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X)))))\| \\ &\leq \|W_{i,2} - Q_{i,2}\|\|\sigma(Q_{i,1}(\sigma(y_{i-1}(X))))\| \leq \|W_{i,2} - Q_{i,2}\|\|Q_{i,1}\|\|y_{i-1}(X)\|.\end{aligned}$$

The third term in (3.11) is bounded by:

$$\|Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X))))) - Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1,Q}(X)))))\|$$

$$\leq \|Q_{i,2}\|\|Q_{i,1}\|\|y_{i-1}(X) - y_{i-1,Q}(X)\|.$$

Finally, the fourth term in (3.11) is bounded by:

$$\|\sigma(y_{i-1}(X)) - \sigma(y_{i-1,Q}(X))\| \leq \|y_{i-1}(X) - y_{i-1,Q}(X)\|.$$

From the four inequalities above, we obtain:

$$\begin{aligned}\|y_i(X) - y_{i,Q}(X)\| \leq \{\lambda\|W_{i,1} - Q_{i,1}\| + \|W_{i,2} - Q_{i,2}\|\|Q_{i,1}\|\} \times \|y_{i-1}(X)\| \\ + \{\|Q_{i,2}\|\|Q_{i,1}\| + 1\} \times \|y_{i-1}(X) - y_{i-1,Q}(X)\|. \end{aligned} \tag{3.12}$$

From the proof of Lemma 3.3.1 and Lemma 3.3.2, using $\|A\| = \|A^\top\|$ we have $\|W_{i,1} - Q_{i,1}\|, \|W_{i,2} - Q_{i,2}\| \leq 2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}}$ and $\|Q_{i,1}\|, \|Q_{i,2}\| \leq 2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}} + \lambda$. Applying these to (3.12), we have:

$$\begin{aligned}\|y_i(X) - y_{i,Q}(X)\| &\leq 2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}}(\lambda + 2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}} + \lambda)\|y_{i-1}(X)\| \\ &\quad + \left((2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}} + \lambda)^2 + 1\right)\|y_{i-1}(X) - y_{i-1,Q}(X)\| \\ &= A\|y_{i-1}(X)\| + B\|y_{i-1}(X) - y_{i-1,Q}(X)\|. \end{aligned} \tag{3.13}$$

From the recurrence relations (3.9) and (3.10), one can easily check that

$$\begin{aligned}\|y_i(X)\| &= \|W_{i,2}(\sigma(W_{i,1}(\sigma(y_{i-1}(X))))) + \sigma(y_{i-1}(X))\| \\ &\leq \|W_{i,2}\|\|W_{i,1}\|\|\|y_{i-1}(X)\| + \|y_{i-1}(X)\| \leq (\lambda^2 + 1)\|y_{i-1}(X)\|. \end{aligned} \tag{3.14}$$

Using (3.14) repeatedly, we obtain $\|y_i(X)\| \leq (\lambda^2+1)^i\|y_0(X)\| = (\lambda^2+1)^i\|X\|$, which together with (3.13) implies:

$$\begin{aligned}\|y_i(X) - y_{i,Q}(X)\| &\leq A\|y_{i-1}(X)\| + B\|y_{i-1}(X) - y_{i-1,Q}(X)\| \\ &\leq A(\lambda^2+1)^{i-1}\|X\| + B\|y_{i-1}(X) - y_{i-1,Q}(X)\|. \end{aligned} \tag{3.15}$$

To prove (3.8), we proceed by induction. When $i = 1$, (3.15) yields:

$$\|y_1(X) - y_{1,Q}(X)\| \leq A(\lambda^2+1)^0\|X\| + B\|y_0(X) - y_{0,Q}(X)\| = A\|X\|.$$

Therefore, (3.8) holds when $i = 1$. Now, let us assume that (3.8) holds for $i = t$. When $i = t+1$, by (3.15) and the induction hypothesis, we have

$$\begin{aligned}&\|y_{t+1}(X) - y_{t+1,Q}(X)\| \leq A(\lambda^2+1)^t\|X\| + B\|y_t(X) - y_{t,Q}(X)\| \\ &\leq A\|X\|\{(\lambda^2+1)^t + B\sum_{j=0}^{t-1}(\lambda^2+1)^j B^{t-1-j}\} = A\|X\|\sum_{j=0}^{t}(\lambda^2+1)^j B^{t-j}.\end{aligned}$$

Hence, (3.8) holds. Note that $g(x) = y_n(x)$ and $g_Q(x) = y_{n,Q}(x)$. Taking $i = n$ in (3.8) completes the proof. □

While the general structure of residual blocks consists of rectangular matrices $W_{i,1}$'s and $W_{i,2}$'s, some literature [40, Chapter 3.8.2] defines this structure using $k \times k$ square weight matrices. In this special case, Algorithm 1 can be applied directly to $W_{i,1}$'s and $W_{i,2}$'s without requiring a transpose. The following Corollary addresses the scenario where $r_i = k$ for all $k = 1, \cdots, n$.

Corollary 3.3.7. *Let $g(x)$ be a residual neural network with n residual blocks (2.12) where $r_i = k$ for*

all $k = 1, \cdots, n$.. *Let*

$$\lambda = \max_{i=1,\cdots,n} \Big\{ \max\{\|W_{i,2}\|, \|W_{i,1}\|\} \Big\}.$$

Let $g_Q(x)$ *be the quantized residual neural network of* $g(x)$ *(2.14) from Algorithm 1. Then for any input* $X \in \mathbb{R}^k$, *we have*

$$\begin{aligned} \|g(X) - g_Q(X)\| \leq\ & 4\delta k\sqrt{k(k+3)}(\delta k\sqrt{k(k+3)} + \lambda)N^{-\frac{1}{k}}\|X\| \\ & \times \sum_{j=0}^{n-1}(\lambda^2+1)^j\Big((2\delta k\sqrt{k(k+3)}N^{-\frac{1}{k}} + \lambda)^2 + 1\Big)^{n-1-j}. \end{aligned} \tag{3.16}$$

Theorem 3.3.6 shows that a smaller step size δ and larger N improve the accuracy of a quantized neural network. Treating δ as a fixed constant, we can view the upper bound in (3.7) as $O(N^{-1/k})\|X\|$, similar to the interpretation in Section 3.3.1. This result applies to general FUNTFs. However, for FUNTFs with uniformly bounded variation, the terms $N^{-\frac{1}{k}}$ in (3.7) can be replaced by N^{-1}, further reducing the reconstruction error.

3.4 Numerical Results

In this section, we present numerical results that illustrate the effectiveness of our proposed algorithm. We begin by evaluating the accuracy of quantized networks obtained via Algorithm 1 on the MNIST classification task. Our experiments are conducted on two neural network architectures: a 3-layer FNN and a network with 2 residual blocks. Each network is trained independently 10 times and subsequently quantized using various values of N and δ, where N denotes the number of elements in the FUNTF employed. We report the average accuracy and standard deviation across different pairs (δ, N), and numerically confirm consistency with the theoretical results established in Section

3.3. Next, we show how Algorithm 1 can benefit under extreme quantization settings from 1-bit quantization on the same neural network architectures. Finally, we demonstrate the performance of our method on a more realistic setting using the ResNet-18 architecture for CIFAR-10 classification. We compare the test accuracies of quantized networks with other PTQ benchmark and highlight the benefits of incorporating frame redundancy in Algorithm 1.

3.4.1 Feedforward Network with 3 Layers

We trained 10 3-layer FNNs with architecture $f(x) = h^{[3]} \circ \sigma \circ h^{[2]} \circ \sigma \circ h^{[1]}(x)$ where σ is the ReLU activation, $h^{[1]} : \mathbb{R}^{784} \to \mathbb{R}^{256}$, $h^{[2]} : \mathbb{R}^{256} \to \mathbb{R}^{256}$, and $h^{[3]} : \mathbb{R}^{256} \to \mathbb{R}^{10}$ are affine maps as in (2.11) without bias terms. The MNIST images were normalized by dividing each pixel value by 255. Training was carried out for 10 epochs using the Adam optimizer with its default hyper-parameters, a mini-batch size of 64, and the categorical cross-entropy loss function. After training, each network was quantized with the harmonic frame H_N^{256}. We quantized each FNN using $N \in \{256, 320, 384, 448, 512\}$ and $\delta \in \{1/16, 1/8, 1/4, 1/2, 1\}$. The numerical results are summarized in Table 3.1. As expected, quantizing with smaller δ and larger N generally led to higher test accuracy.

N	$\delta = 1/16$	$\delta = 1/8$	$\delta = 1/4$	$\delta = 1/2$	$\delta = 1$
256	$97.53 \pm 0.24\%$	$97.03 \pm 0.35\%$	$93.39 \pm 1.92\%$	$63.76 \pm 4.54\%$	$22.85 \pm 3.67\%$
320	$97.62 \pm 0.19\%$	$97.35 \pm 0.29\%$	$95.47 \pm 1.34\%$	$84.93 \pm 4.66\%$	$50.30 \pm 8.24\%$
384	$97.65 \pm 0.25\%$	$97.46 \pm 0.28\%$	$95.99 \pm 1.59\%$	$90.68 \pm 3.80\%$	$62.48 \pm 6.54\%$
448	$97.68 \pm 0.22\%$	$97.55 \pm 0.24\%$	$96.92 \pm 0.59\%$	$93.75 \pm 2.03\%$	$76.66 \pm 5.25\%$
512	$97.68 \pm 0.25\%$	$97.57 \pm 0.25\%$	$97.20 \pm 0.30\%$	$95.71 \pm 1.36\%$	$86.97 \pm 1.92\%$

Table 3.1: Frame Quantization for FNN with 3 layers. The test accuracy for the pretrained neural network is $97.72 \pm 0.21\%$.

We now demonstrate the connection between theory and numerical results. Since we are using harmonic frames with the same N and δ for quantizing each layer, the error bound (3.6) can be

represented as $\|f(X) - f_Q(X)\| \leq C\delta N^{-1}\|X\|$, where C is a positive constant depending only on the FNN f. In Figure 3.1 we present the worst-case error. The worst-case error decreases as N increases or δ decreases, but it may be large if the worst-case scenario corresponds to large $\|X\|$. To remove the dependence on X and to focus on the dependence of N and δ, we compute the average and represent the error bound as $\mathrm{E}_X\|f(X) - f_Q(X)\| \leq C \times \mathrm{E}\|X\| \times \delta N^{-1}$. Note that $\mathrm{E}\|X\|$ is now a constant depending only on the MNIST dataset, so by taking logarithm on both sides, we can write $\log \mathrm{E}_X[\|f(X) - f_Q(X)\| \times N/\delta] \leq K$, for some constant $K > 0$. In Figure 3.2, we present the values of $\log \mathrm{E}_X[\|f(X) - f_Q(X)\| \times N/\delta]$. We can see that the values are bounded and behave approximately constant as N grows. These numerical results imply that our theoretical error bound is tight in terms of δ and N.

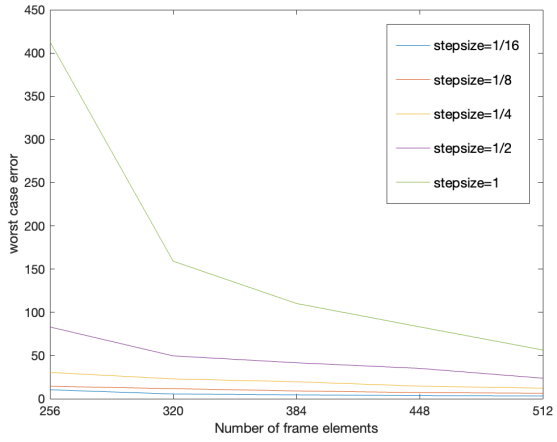

Figure 3.1: Worst-case error $\|f(X) - f_Q(X)\|$ for FNN with 3 layers.

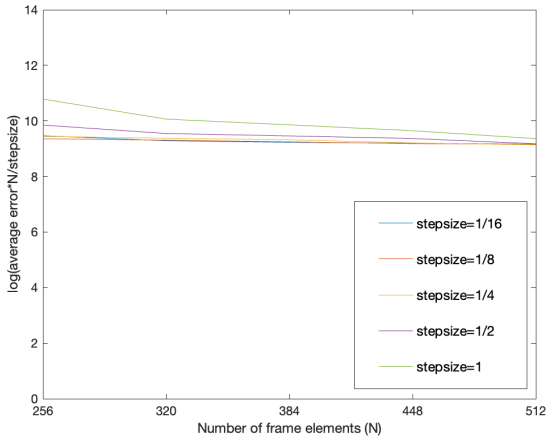

Figure 3.2: $\log \mathrm{E}_X[\|f(X) - f_Q(X)\| \times N/\delta]$ for FNN with 3 layers.

3.4.2 Network with 2 Residual Blocks

We trained 10 neural networks with architecture $g(x) = h^{[2]} \circ \sigma \circ z^{[2]} \circ \sigma \circ z^{[1]} \circ \sigma \circ h^{[1]}(x)$ where σ denotes the ReLU activation, $h^{[1]} : \mathbb{R}^{784} \to \mathbb{R}^{256}$ and $h^{[2]} : \mathbb{R}^{256} \to \mathbb{R}^{10}$ are affine maps in (2.11),

and $z^{[1]}, z^{[2]} : \mathbb{R}^{256} \to \mathbb{R}^{256}$ are residual blocks in (2.12) having the form of

$$z^{[1]}(x) = W_{1,2}\sigma(W_{1,1}x + b_1) + x, \quad z^{[2]}(x) = W_{2,2}\sigma(W_{2,1}x + b_2) + x,$$

with square weight matrices $W_{1,1}, W_{1,2}, W_{2,1}, W_{2,2} \in \mathbb{R}^{256\times256}$ and bias vectors $b_1, b_2 \in \mathbb{R}^{256}$. Again, the MNIST images were normalized by dividing each pixel value by 255. Training was carried out for 5 epochs using the Adam optimizer with its default hyper-parameters, a mini-batch size of 64, and the categorical cross-entropy loss function. After training, each network was quantized with the harmonic frame H_N^{256}. We quantized each neural network using $N \in \{256, 320, 384, 448, 512\}$ and $\delta \in \{1/16, 1/8, 1/4, 1/2, 1\}$. We report our results in Table 3.2. Similar to the results for quantizing FNNs, quantizing with smaller δ and larger N yielded test accuracy close to that of the original unquantized network. This demonstrates the impact of δ and N in Algorithm 1, which is consistent with the theoretical explanations given in Section 3.3.

N	$\delta = 1/16$	$\delta = 1/8$	$\delta = 1/4$	$\delta = 1/2$	$\delta = 1$
256	$97.44 \pm 0.28\%$	$96.17 \pm 0.67\%$	$77.20 \pm 6.23\%$	$15.61 \pm 2.89\%$	$9.68 \pm 2.55\%$
320	$97.52 \pm 0.32\%$	$97.09 \pm 0.35\%$	$92.54 \pm 1.19\%$	$41.03 \pm 6.94\%$	$11.67 \pm 2.58\%$
384	$97.60 \pm 0.19\%$	$97.23 \pm 0.27\%$	$95.12 \pm 0.62\%$	$66.27 \pm 8.36\%$	$17.84 \pm 2.83\%$
448	$97.64 \pm 0.22\%$	$97.37 \pm 0.28\%$	$96.35 \pm 0.42\%$	$85.28 \pm 3.53\%$	$28.96 \pm 6.04\%$
512	$97.66 \pm 0.20\%$	$97.54 \pm 0.25\%$	$97.00 \pm 0.29\%$	$92.30 \pm 3.11\%$	$45.80 \pm 10.09\%$

Table 3.2: Frame Quantization for Neural Network with 2 Residual Blocks. The test accuracy for the pretrained neural network is $97.72 \pm 0.20\%$.

3.4.3 1-Bit Quantization

1-bit quantization of neural network, in other words, *binarization of neural network*, refers to a method of compressing weights or even activations into 1-bit numbers. For example, [25] quantized the weights of the neural networks to $\{-1, 1\}$. Mathematical theories were also developed in 1-bit $\Sigma\Delta$

quantization for bandlimited signals [42] and approximation capabilities of 1-bit neural networks [46].

In our method, since we are storing q_{jk}^i's appearing in the quantized expansions, we will say 1-bit quantization is obtained when all q_{jk}^i's are in $\{-a, a\}$ for some positive number $a > 0$. Note that once we choose appropriate $K \in \mathbb{N}$ and step size $\delta > 0$, our method forces q_{jk}^i's as elements of the midrise quantization alphabet A_K^δ (2.9). If we recall the structure of $A_K^\delta = \{(-K + \frac{1}{2})\delta, (-K + \frac{3}{2})\delta, \cdots, -\frac{1}{2}\delta, \frac{1}{2}\delta, \cdots, (K - \frac{1}{2})\delta\}$, one can easily see that A_K^δ achieves a form of $\{-a, a\}$ if and only if $K = 1$.

Therefore, in our experiments for 1-bit quantization, we set $K = 1$ for all layers and found the uniform step size δ that satisfies (3.1) for $i = 1, \cdots, n$. For both neural networks in the previous experiments, setting $\delta = 8$ enables us to set $K = 1$. Our 1-bit quantization results can be found in Table 3.3 and Table 3.4.

Now we shall explain how our algorithm can benefit in storage while maintaining the accuracy. Consider 1-bit quantization for a neural network with 3 layers. We use the harmonic frame H_N^{256} to quantize the first and second weight matrices W_1 and W_2, the total bits that we need to store the quantized matrices Q_1 and Q_2 is $1 \times 784 \times N + 1 \times 256 \times N = 1040N$ bits. The total bits that we need to represent the weight matrices W_1 and W_2 are $32 \times 256 \times 784 + 32 \times 256 \times 256 = 8192 \times 1040$ bits. Suppose that we use $N = 7000$. Then the total bits that we can save is $8192 \times 1040 - 1040N = 1192 \times 1040$ bits, but we have an average accuracy of 97.29% for the quantized networks, where the average accuracy for the trained neural networks is 97.72%. So we can even benefit in storage for a 1-bit quantization case with a small sacrifice in terms of accuracy.

N	1000	2000	3000	4000	5000	6000	7000
Average	36.18%	74.09%	88.47%	94.62%	96.04%	96.83%	97.29%
Standard Deviation	7.65	4.71	1.82	0.44	0.82	0.36	0.25

Table 3.3: 1-Bit Frame Quantization for FNN with 3 layers

N	1000	2000	3000	4000	5000	6000	7000
Average	14.12%	31.52%	66.93%	86.47%	92.88%	95.28%	96.30%
Standard Deviation	2.52	6.56	5.69	4.04	1.44	0.64	0.50

Table 3.4: 1-Bit Frame Quantization for Neural Network with 2 Residual Blocks

3.4.4 ResNet-18 Quantization

For a more realistic application scenario, we evaluated the performance of our frame-based quantization algorithm on a ResNet-18 model, where its architecture is supported by Pytorch [88], trained for CIFAR-10 classification. The ResNet-18 architecture was trained using the categorical cross-entropy loss function on the CIFAR-10 training set with a mini-batch size of 128. We employed stochastic gradient descent with Nesterov momentum, setting the learning rate to 0.1, momentum to 0.9, and weight decay to 5×10^{-4}, over 200 training epochs. Input images were normalized to zero mean and unit variance using the standard per-channel mean and standard deviation values for CIFAR-10.

Following training, we applied Algorithm 1 to quantize the model under various settings. Note that when quantizing a weight matrix $W \in \mathbb{R}^{m\times l}$ using a FUNTF F with cardinality $|F| = N$, the quantized representation involves storing a matrix C which is a $l \times N$ matrix. Hence, assuming that each entry of W is originally stored in 32-bit precision and the level of quantization alphabet is set as $K = 2^{b-1}$, the storage required for C becomes a fraction $\frac{bN}{32m}$ of the original storage required for W.

In our experiments, we fixed the redundancy ratio $N/m = r$ uniformly across layers and used the harmonic frame H_N^m for quantizing a weight matrix $W \in \mathbb{R}^{m\times l}$ using Algorithm 1, and evaluated the classification test accuracy of the quantized models for various combinations of bit-widths $b \in \{3, 4\}$ and redundancy ratios $r \in \{1, 1.1, 1.2, 1.3\}$. Only the convolutional and fully connected linear weights

were quantized; batch normalization weights and biases were kept in full precision, which is consistent with standard PTQ methods. As a benchmark, we also conducted numerical experiments using Greedy Path Following Quantization (GPFQ) [74], and compared its test accuracy with that of our method. For GPFQ, we used a calibration dataset by choosing either 128 or 256 images from the CIFAR-10 training dataset, following the idea of using small calibration sets as described in [55]. The test accuracy results for both Frame Quantization and GPFQ are presented in Table 3.5. The pretrained ResNet-18 model achieved a baseline test accuracy of 90.46% prior to quantization.

	Pretrained	Frame Quantization				GPFQ	
		$r = 1$	$r = 1.1$	$r = 1.2$	$r = 1.3$	$c = 128$	$c = 256$
$b = 3$	90.46%	72.62%	76.63%	78.11%	87.56%	73.40%	67.70%
$b = 4$	90.46%	80.59%	88.6%	89.1%	89.78%	87.70%	88.30%

Table 3.5: Test accuracy of b-bit quantized ResNet-18 on CIFAR-10, using Frame Quantization with redundancy ratio $r \in \{1, 1.1, 1.2, 1.3\}$ and GPFQ with calibration set of size $c \in \{128, 256\}$.

We note that for both 3-bit and 4-bit quantization, while using no redundancy ($r = 1$) can achieve lower test accuracy than GPFQ, utilizing a small amount of redundancy of $r = 1.1$ led to achieving better test accuracy on the CIFAR-10 dataset. This highlights the effectiveness of incorporating redundancy of finite frames in Algorithm 1. To further demonstrate the strength of redundancy, we examined an extreme setting, 1-bit quantization. Even under this lowest-bit representation, the quantized network achieved test accuracy on CIFAR-10 that approached the original pretrained ResNet-18 accuracy of 90.46%, as the redundancy ratio r increased. The corresponding numerical results are reported in Table 3.6.

r	1	2	3	4	8	12	16
Test Accuracy	8.62%	27.21%	50.73%	80.3%	88.88%	89.48%	90.17%

Table 3.6: 1-Bit Frame Quantization for ResNet-18 with CIFAR-10 dataset

The numerical results in Table 3.5 and Table 3.6 highlight several advantages of our Frame Quantization algorithm. First, even without access to any training data for calibration, introducing a small amount of redundancy enables the quantized network to achieve higher test accuracy. Although this comes at the cost of slightly increased storage, it offers benefits in scenarios where data privacy is a concern. Second, the frame-based algorithm enables an accurate reconstruction of the original neural network from extremely low-bit representations while offering storage savings. For example, $r = 16$ in Table 3.6 corresponds to a 1-bit represented ResNet-18, which has test accuracy on CIFAR-10 close to that of the original pretrained network, but using the half amount of storage. While this may not represent the most storage-optimal form of 1-bit quantization, the frame-based representation is particularly advantageous in settings where models must be transmitted in highly compressed low-bit formats, an arrangement reminiscent of classical signal processing scenarios.

3.5 Conclusions and Future Work

We proposed a PTQ method with rigorous mathematical analysis of error estimates between a neural network and its quantized network. We use frame theory in neural network quantization for the first time. We also demonstrate that 1-bit quantization with our method has a benefit in storage while maintaining accuracy close to the original network. But still many open questions remain. Here, we list some of them.

Beyond FUNTFs : Algorithm 1 required a FUNTF for its execution. Designing a quantization method using a broader class of finite frames would be an appealing research direction in the future.

Higher-order $\Sigma\Delta$ method : While Algorithm 1 employs the first-order $\Sigma\Delta$ quantization method, a natural extension involves the utilization of higher-order $\Sigma\Delta$ quantization schemes. It remains an

open question how the error bounds scale with the number of frame elements under such higher-order methods; specifically, one would expect an accelerated error decay rate relative to the frame redundancy. Furthermore, identifying the optimal quantization rules that maintain stability in higher-order settings is a critical step for future theoretical development.

Inference Time : To the best of our knowledge, this work represents the first attempt to incorporate frame theory into the quantization of neural networks. Although algorithm 1 provides a rigorous justification for accuracy and storage efficiency, the reduction of inference time remains to be addressed. Developing a frame-theoretic quantization framework that facilitates hardware acceleration would be a fruitful direction for subsequent research. This may require developing some specific finite frames.

Quantization Alphabet : Algorithm 1 considers a uniformly spaced symmetric quantization alphabet. On the other hand, one may consider using a nonuniform asymmetric quantization alphabet. In this case, the design of quantization algorithm may depend on the distributions of both the dataset and the pretrained weight matrices.

Chapter 4: One-bit quantized neural networks on manifolds

4.1 Introduction

As noted in Section 3.1, practitioners have developed sophisticated quantization methods, focusing primarily on empirical performance rather than rigorous provable guarantees. Despite the practical success of quantized neural networks, a comprehensive understanding of their underlying mechanisms remains limited. A fundamental question concerns the approximation theory of quantized neural networks, which characterizes the required network complexity in terms of a desired approximation error ϵ. Developing this fundamental foundation is crucial for bridging the gap between theory and practice.

The universal approximation theory of one-bit neural networks was first established in [46] through the following theorem.

Theorem 4.1.1 (Theorem 4 in [46]). *Let $\sigma(x) = \frac{1}{2}\max\{x, 0\}$ be the scaled ReLU activation function. For any $s, d \in \mathbb{N}$, consider the unit ball $\mathcal{F}_{s,d}$ of $C^s([0,1]^d)$. That is,*

$$\mathcal{F}_{s,d} = \{f \in C^s([0,1]^d) : \sup_{|\mathbf{s}|_1 \le s} \|D^{\mathbf{s}} f\|_\infty \le 1\}.$$

For all $0 < \epsilon < 1/2$, there is a set $\mathcal{N}$ of strict one-bit networks with activation σ such that the following hold. They all have the same structure, at most $C_{d,s}\log_2(1/\epsilon)$ layers, at most $C_{d,s}\epsilon^{-d/s}(\log_2(1/\epsilon))^2$ many nodes and parameters, and

$$\sup_{f \in \mathcal{F}_{s,d}} \inf_{g \in \mathcal{N}} \|f - g\|_\infty \le \epsilon.$$

A natural question arises: can a universal approximation theorem of one-bit neural networks be established for functions defined on manifolds? Motivated by the manifold hypothesis, which posits that high-dimensional data often reside on or near a lower-dimensional manifold embedded within the ambient space, significant research efforts [8, 24, 26, 30, 95, 112, 114] have been dedicated to developing effective nonlinear dimensionality reduction methods. Consequently, in many practical applications of neural networks, the approximation error on these low-dimensional manifolds is of primary interest. Notably, several studies [19,20,98,101] have explored neural network approximation theory specifically on manifolds.

In this chapter, building upon Theorem 4.1.1 and the work in [19, 20], we develop the first universal approximation theory for one-bit neural networks approximating the Hölder function class on a d-dimensional compact Riemannian manifold $\mathcal{M}$ isometrically embedded in $\mathbb{R}^D$. Specifically, we demonstrate that the exponential dependence on the approximation error ϵ-in terms of the complexity of layers, nodes, and parameters-is governed solely by the intrinsic dimension d and the regularity of the Hölder function class. In addition, we characterize the weak quantitative dependence of the network size on the ambient dimension D. We state the theorem in Section 4.3.

4.2 Notation and Setting

Prior to stating the formal problem, we establish the notation and settings employed throughout this chapter. In this chapter, we consider FNNs equipped with the scaled activation $\sigma(x) = \frac{1}{2}\max\{x, 0\}$, following the convention in [46]. Note that the $\frac{1}{2}$ factor combined with $\{\pm 1\}$ quantization is mathematically equivalent to utilizing the standard ReLU activation in conjunction with a $\{\pm\frac{1}{2}\}$ alphabet. In practical applications, one-bit quantization is commonly implemented with scaling factors to mitigate approximation

errors by restoring magnitude information lost during quantization—a technique widely adopted in convolutional neural networks [93] and large language models [117]. These observation provide the theoretical rationale for adopting the $\frac{1}{2}$-scaled ReLU activation in this study.

We assume that

- $\mathcal{M}$ is a compact $d-$dimensional Riemannian manifold, isometrically embedded in $\mathbb{R}^D$
- $\mathcal{M}$ has reach $\tau > 0$.

The differentiability of functions on $\mathcal{M}$ does not depend on the choice of atlas. However, to restrict the target function space, we need to fix an atlas in order to quantify the magnitude of derivatives.

We start with choosing radius r as $r = 2^{-\beta}$ for $\beta = \lceil \log_2(4/\tau) \rceil$. This makes

$$r = 2^{-\beta} \leq 2^{-\log_2(4/\tau)} = \tau/4.$$

Since $\mathcal{M}$ is compact, there exists a finite set $\{c_1, \cdots, c_{C_\mathcal{M}}\} \subset \mathcal{M}$ such that $\mathcal{M} \subset \cup_{i=1}^{C_\mathcal{M}} B_r(c_i)$. Now define $U_i = \mathcal{M} \cap B_r(c_i)$ and P_i as the orthogonal projection onto the tangent space $T_{c_i}(\mathcal{M})$. The following lemma guarantees that $(U_i, P_i)_{i=1,\cdots,C_\mathcal{M}}$ is an atlas of $\mathcal{M}$.

Lemma 4.2.1 (Lemma 2 in [20]). *For $c \in \mathcal{M}$, if $r \leq \tau/4$, then $B_r(c) \cap \mathcal{M}$ is diffeomorphic to $\mathbb{R}^d$. In particular, the orthogonal projection P onto the tangent space $T_c(\mathcal{M})$ at c is a diffeomorphism.*

For simplicity, we denote $\mathcal{P} = \{P_1, \cdots, P_{C_\mathcal{M}}\}$. Now, we define our target function space $H^{s,\alpha}(\mathcal{M})$ using the atlas chosen above.

Definition 4.2.2. *For a positive integer s and $\alpha \in (0, 1]$, a function $f : \mathcal{M} \to \mathbb{R}$ is $(s + \alpha)-$Holder continuous if*

1. $f \circ P_i^{-1} \in C^s$ *with* $\left| D^{\mathbf{s}}(f \circ P_i^{-1})\big|_{P_i(x)} \right| \leq 1$ *for any* $|\mathbf{s}| \leq s, x \in U_i$,

2. *For any* $|\mathbf{s}|_1 = s$, *and* $x_1, x_2 \in U_i$,

$$\left| D^{\mathbf{s}}(f \circ P_i^{-1})\big|_{P_i(x_1)} - D^{\mathbf{s}}(f \circ P_i^{-1})\big|_{P_i(x_2)} \right| \leq \|P_i(x_1) - P_i(x_2)\|_2^{\alpha},$$

hold for all $i = 1, \cdots, C_{\mathcal{M}}$. *Moreover, we denote the collection of* $(s+\alpha)-$*Hölder functions on* $\mathcal{M}$ *as* $\mathcal{H}^{s,\alpha}(\mathcal{M})$.

Since $\mathcal{M}$ is compact, $\|x\|_\infty$ is bounded by some positive constant. Define $B \in \mathbb{N}$ as the smallest positive integer such that $\|x\|_\infty \leq B/2$ holds for any $x \in \mathcal{M}$. Note that B is determined only by the geometric structure of $\mathcal{M}$.

We write the tangent space at c_i as

$$T_{c_i}(\mathcal{M}) = \text{span}\{v_{i1}, \cdots, v_{id}\},$$

where $\{v_{i1}, \cdots, v_{id}\}$ form an orthonormal basis. Define $V_i = [v_{i1}, \cdots, v_{id}] \in \mathbb{R}^{D \times d}$ by concatenating v_{ij}'s as column vectors, and define

$$\phi_i(x) = b_i(V_i^T(x - c_i) + u_i),$$

for any $x \in U_i$. Here, scaling factor $b_i \in (0, 1]$ and translation vector u_i are chosen to guarantee $\phi_i(x) \in [\frac{1}{3}, \frac{2}{3}]^d$ for all $x \in U_i$. Note that all b_i, V_i, and u_i are determined by c_i, r, and $\mathcal{M}$. Note that r depends only on τ, hence ϕ_i depends only on c_i, τ, and $\mathcal{M}$. We denote $\Phi = \{\phi_i\}_{i=1,\cdots,C_{\mathcal{M}}}$.

Since $\mathcal{M}$ is a compact smooth manifold, there exists a C^∞ partition of unity $\mathcal{R} = \{\rho_i\}_{i=1,\cdots,C_{\mathcal{M}}}$

where the support of ρ_i is a subset of U_i for $i = 1, \cdots, C_{\mathcal{M}}$. If multiple such C^∞ partitions exist, we choose one and fix it. Note that throughout this chapter, $\mathcal{P}, \Phi$ and $\mathcal{R}$ remain fixed.

Throughout this chapter, C denotes a positive absolute constant whose value may change from line to line, depending on the context. $C_{t_1,\cdots,t_m,\mathcal{S}_1,\cdots,\mathcal{S}_n}$ denotes a positive constant that depends only on parameters $t_1, \cdots, t_m$ and set of functions $\{\mathcal{S}_1, \cdots, \mathcal{S}_n\}$.

We denote multi-indices and tuples with boldface letters. We write $\mathbf{0} = (0, \cdots, 0) \in \mathbb{R}^d$.Slightly abusing notation, $\mathbf{n}$ is special and we write $\mathbf{n} = (n, \ldots, n)$ for $n \in \mathbb{N} \cup \{0\}$. For $\mathbf{k}, \mathbf{p} \in \mathbb{N}^d$, we write $\mathbf{k} \leq \mathbf{p}$ as shorthand for $k_j \leq p_j$ for all $j \in \{1, \ldots, d\}$. For any $x = (x_1, \cdots, x_d) \in \mathbb{R}^d$, let $x^{\mathbf{k}} := x_1^{k_1} \cdots x_d^{k_d}$. The degree of $x^{\mathbf{k}}$ is defined to be $|\mathbf{k}|_1$. For any $t \in \mathbb{R}$, we let $t^{\mathbf{k}} = (t^{k_1}, \ldots, t^{k_d})$.

4.3 Main Theorem

The following theorem constitutes the central result of this chapter. We establish the first rigorous universal approximation theory for one-bit neural networks within the Hölder function class $\mathcal{H}^{s,\alpha}(\mathcal{M})$. Theorem 4.3.1 establishes the existence of a one-bit neural network architecture utilizing ReLU activations that achieves universal approximation for $\mathcal{H}^{s,\alpha}(\mathcal{M})$. In addition, it quantifies the requisite network complexity—including the number of layers, nodes, and parameters—as a function of the prescribed supremum norm error $\epsilon > 0$, and demonstrates the weak dependency of the network size on the ambient dimension D.

Theorem 4.3.1. *For any sufficiently small $\epsilon > 0$, there exists a set $\mathcal{N}$ of strict one-bit neural networks such that the following hold.*

1. *They all have the same structure,*

2. *Have at most $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}(\log_2 D + \log_2(1/\epsilon))$ layers,*

3. *Have at most* $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}\left(\epsilon^{-d/(s+\alpha)}(\log_2(1/\epsilon))^2 + D(\log_2(1/\epsilon))^2 + D(\log_2 D)^2\right)$ *nodes and parameters,*

4. $\sup_{f\in\mathcal{H}^{s,\alpha}} \inf_{g\in\mathcal{N}} \|f-g\|_\infty \leq \epsilon$.

4.4 Outline of the proof and supporting lemmas

We prove Theorem 4.3.1 by following the primary approach outlined in [19]. Using C^∞ partition of unity $\mathcal{R}$, we can decompose the target function f as $f = \sum_{i=1}^{C_\mathcal{M}} f\rho_i$. Our strategy is to find one-bit neural networks having the same structure that approximate $f\rho_i$. Since ρ_i has its support in U_i, by approximating an indicator function $\mathbb{1}_{U_i}$ with a one-bit neural network, we can efficiently assign inputs x to their respective neighborhoods U_i's. We also derive one-bit approximations for each $f\rho_i$. Using that multiplication operator can be approximated by one-bit neural networks (Lemma 4.4.8), we prove the existence of one-bit neural network that approximates the target function with the desired approximation error ϵ, and count the required number of layers, nodes, and parameters of the one-bit network.

The remainder of this section is devoted to establishing the technical machinery required for the proof of Theorem 4.3.1. We introduce several key lemmas that are frequently utilized throughout the proof. These results provide the fundamental building blocks for approximating several functions under the one-bit constraint. To maintain the focus of our exposition, proofs for the results previously established in [19, 20, 46] are omitted, and we direct the interested reader to those sources for detailed derivations.

4.4.1 One-bit neural network expansion

We begin by establishing that both the identity and absolute value functions can be implemented using one-bit neural networks. These lemmas serve as essential technical tools to align the number of layers of two distinct one-bit neural network architectures.

Lemma 4.4.1. *There exist strict one-bit neural networks with 2 layers and constant number of nodes and parameters, implementing $x \mapsto |x|$ and $x \mapsto x$.*

Proof. The following two equalities finish the proof.

$$|x| = \sigma(x) + \sigma(x) + \sigma(-x) + \sigma(-x) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \sigma(\begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}^T x).$$

$$x = \sigma(x) + \sigma(x) - \sigma(-x) - \sigma(-x) = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} \sigma(\begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}^T x).$$

□

Similarly, one can easily verify the following lemma.

Lemma 4.4.2. *There exist activated one-bit neural networks with 2 layers and 3 layers, both of them using constant number of nodes and parameters, implementing $x \to |x|$.*

Proof. Note that for 2-layer case,

$$|x| = \sigma(2|x|) = \sigma(\sigma(x) + \sigma(x) + \sigma(x) + \sigma(x) + \sigma(-x) + \sigma(-x) + \sigma(-x) + \sigma(-x)),$$

finishes the proof. For 3-layer case, use $|x| = \sigma(2|x|)$ again. □

Now we prove that for any $L \geq 2$, a constant function $x \mapsto n \in \mathbb{N}$ can be implemented by an activated one-bit neural network with L layers.

Lemma 4.4.3. *For any positive integer n and $L \geq 2$, there exists an activated one-bit neural network of L layers, implementing $x \to n$ with at most CLn nodes and parameters for some absolute constant $C > 0$.*

Proof. Let w be a $1 \times 4n$ matrix with its entries all equal to 1. Let A be a $2n \times d$ zero matrix. Then we have

$$n = \sigma(2n) = \sigma(4n\sigma(1)) = \sigma(w\sigma(Ax + w^T)).$$

Therefore, a constant function n can be implemented by an activated one-bit neural network of size $(2, 4n + 1, 8n)$.

Since $n = \sigma(2n)$ and constant function $2n$ can be implemented by an activated one-bit neural network with size $(w, 8n + 1, 16n)$, constant function n can be implemented by an activated one-bit neural network with size $(3, 8n + 2, 16n + 1)$.

Note that $n = \sigma(2n) = \sigma(\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \sigma(\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T n)$, so $n \to n$ can be implemented by an activated one-bit neural network with size $(2, 5, 8)$. Note that $L = 2 \times (L/2)$ when L is even and $L = 3 + 2 \times (L - 3)/2$ when L is odd for all $L \geq 2$. Then, we can implement a constant function n using a L-layer activated one-bit quantized neural network with nodes and parameters both CLn for some absolute constant C.

□

The following lemma demonstrates how the number of layers influences the expressive capacity of one-bit neural networks.

Lemma 4.4.4 (Lemma B.1 in [46]). *For any $m \in \mathbb{Z}$, there is an activated one-bit neural network that has size at most $C|m|$ and implements $x \mapsto 2^m \sigma(x)$.*

A two-layer structure is capable of representing multiplication by any positive integer, provided that the number of nodes is sufficiently large.

Lemma 4.4.5. *For any positive integer $K \geq 1$, there exists a strict one-bit neural network of 2 layers with $2K+1$ nodes and $4K$ parameters, implementing the map $x \mapsto Kx$ for $x \in \mathbb{R}$.*

Proof. Note that $Kx = 2K\sigma(x) - 2K\sigma(-x)$. Therefore, $x \mapsto Kx$ can be implemented by a strict one-bit neural network of 2 layers with $2K+1$ nodes and $4K$ parameters. □

The following lemma establishes the architectural complexity required for the approximation of affine transformations by one-bit neural networks. This result underpins the subsequent derivation of Theorem 4.3.1.

Lemma 4.4.6 (Lemma C.3 in [46]). *Let b, m, d, n be positive integers. For any $A \in \mathbb{R}^{n\times d}$ and $c \in \mathbb{R}^n$ such that $\|A\|_{\max}, \|c\|_\infty \leq 2^m$, there is a strict one-bit neural network that implements a function $\mathbb{R}^d \to \mathbb{R}^n$ such that $\|g(x) - (Ax+c)\|_\infty \leq 2^{-b+1}(d\|x\|_\infty + 1)$ holds for any $x \in \mathbb{R}^d$. This neural network has at most $C(m+b)$ layers, at most $Cd(m+b)^2 n$ nodes and parameters, and the structure only depends on b, m, d, n, and not on the entries of A and c.*

4.4.2 Approximating multiplication

Approximating a multiplication operator with a neural network is an important step in the proof of neural network approximation theory [19, 20, 46, 127]. To achieve this, approximating a square function via one-bit neural network is crucial, due to the identity $xy = \frac{1}{2}\{(x+y)^2 - x^2 - y^2\}$.

The following lemma states that a square function can be approximated by activated one-bit neural networks.

Lemma 4.4.7 (Lemma B.4 in [46]). *For any $\epsilon \in (0, \frac{1}{2})$, there exist two activated one-bit neural networks $S_\epsilon^+, S_\epsilon^-$ with activation σ, both with the same numbers of layers, and each of size $C \log_2(1/\epsilon)$, such that for all $x \in [0,1]$, it holds that*

$$S_\epsilon^-(x) \le x^2 \le S_\epsilon^+(x), \quad |S_\epsilon^+(x) - x^2| \le \epsilon, \quad |S_\epsilon^-(x) - x^2| \le \epsilon.$$

Now we obtain the following lemma, the existence of an one-bit neural network that approximates the multiplication operator $(x, y) \mapsto xy$.

Lemma 4.4.8. *Given $k \in \mathbb{N}$ and $\epsilon \in (0, 1/2)$, there exists a strict one-bit neural network $\hat{\times} : \mathbb{R}^2 \to \mathbb{R}$ with activation σ that has the depth and the size of $O(k + \log_2(1/\epsilon))$ such that for all $x, y \in [-2^k, 2^k]$, we have*

$$\left|\hat{\times}(x, y) - xy\right| \le \epsilon.$$

Proof. We set

$$\hat{\times}(x, y) = (2^{k+1})^2 \big(S_\delta^+(\frac{|x+y|}{2^{k+1}}) - S_\delta^+(\frac{|x-y|}{2^{k+1}})\big). \tag{4.1}$$

Since $|x|, |y| \leq 2^k$, we have $\frac{|x+y|}{2^{k+1}}, \frac{|x-y|}{2^{k+1}} \leq 1$. Therefore, by Lemma 4.4.7, we bound the error of $\hat{\times}$

$$\begin{aligned}
\left|\hat{\times}(x,y) - xy\right| &= (2^{k+1})^2 \left| S_\delta^+(\frac{|x+y|}{2^{k+1}}) - S_\delta^+(\frac{|x-y|}{2^{k+1}}) - ((\frac{|x+y|}{2^{k+1}})^2 - (\frac{|x-y|}{2^{k+1}})^2) \right| \\
&\leq (2^{k+1})^2 \left| S_\delta^+(\frac{|x+y|}{2^{k+1}}) - (\frac{|x+y|}{2^{k+1}})^2 \right| + (2^{k+1})^2 \left| S_\delta^+(\frac{|x-y|}{2^{k+1}}) - (\frac{|x-y|}{2^{k+1}})^2 \right| \\
&\leq 2^{2k+3}\delta.
\end{aligned}$$

We choose $\delta = \frac{\epsilon}{2^{2k+3}}$ to ensure $\left|\hat{\times}(x,y) - xy\right| \leq \epsilon$ for all $x, y \in [-2^k, 2^k]$. Note that from Lemma 4.4.2 and Lemma 4.4.4, there exist two activated one-bit neural networks that have the same number of layers and the same size of Ck that implement $(x,y) \mapsto \frac{|x+y|}{2^{k+1}}$ and $(x,y) \mapsto \frac{|x-y|}{2^{k+1}}$. Therefore, by Lemma 4.4.7, $S_\delta^+(\frac{|x+y|}{2^{k+1}})$ and $S_\delta^+(\frac{|x-y|}{2^{k+1}})$ can be both implemented by activated one-bit neural networks having the same number of layers and the same size of $Ck + C\log_2(1/\epsilon)$. Finally, by Lemma 4.4.4 again, $(2^{k+1})^2 S_\delta^+(\frac{|x+y|}{2^{k+1}})$ and $(2^{k+1})^2 S_\delta^+(\frac{|x-y|}{2^{k+1}})$ can be both implemented by activated one-bit neural networks that have the same number of layers and the same size of $O(k + \log_2(1/\epsilon))$. Hence, $\hat{\times} : \mathbb{R}^2 \to \mathbb{R}$ can be implemented by a strict one-bit neural network with the depth and the size of $O(k + \log_2(1/\epsilon))$. □

4.4.3 Composition of one-bit neural networks

When two functions $g_1 : \mathbb{R}^d \to \mathbb{R}^m$ and $g_2 : \mathbb{R}^m \to \mathbb{R}^n$ are implementable by one-bit neural networks, it is important to check whether their composition $g_2 \circ g_1 : \mathbb{R}^d \to \mathbb{R}^n$ is implementable by a one-bit neural network or not. When the inner function g_1 is implementable by an activated one-bit neural network, then it is straightforward to check that $g_2 \circ g_1 : \mathbb{R}^d \to \mathbb{R}^n$ is also implementable by a one-bit neural network.

Lemma 4.4.9. *If $g_1 : \mathbb{R}^d \to \mathbb{R}^m$ is implementable by an activated one-bit neural network of size*

(L_1, N_1, P_1) *and* $g_2 : \mathbb{R}^m \to \mathbb{R}^n$ *is implementable by a strict (resp., activated) one-bit neural network of size* (L_2, N_2, P_2)*, then* $g_2 \circ g_1 : \mathbb{R}^d \to \mathbb{R}^n$ *is implementable by a strict (resp., activated) one-bit neural network of size* $(L_1 + L_2, N_1 + N_2, P_1 + P_2)$.

One the other hand, if g_1 becomes a strict one-bit neural network in Lemma 4.4.9, then $g_2 \circ g_1$ is not necessarily implementable by a one-bit neural network of size $(L_1 + L_2, N_1 + N_2, P_1 + P_2)$, because the composition of the last layer of g_1 and the first layer of g_2 may not be a one-bit layer. In fact, one can construct a one-bit neural network, with size larger than $(L_1 + L_2, N_1 + N_2, P_1 + P_2)$, that implements the composition $g_2 \circ g_1$. To prove this, we begin with a lemma which is also straightforward to check by placing two networks in parallel.

Lemma 4.4.10. *If* $g_1 : \mathbb{R}^d \to \mathbb{R}^m$ *is implementable by a strict (resp., activated) one-bit neural network of size* (L, N_1, P_1) *and* $g_2 : \mathbb{R}^d \to \mathbb{R}^n$ *is implementable by a strict (resp., activated) one-bit neural network of size* (L, N_2, P_2)*, then* $g_1 \oplus g_2 : \mathbb{R}^d \to \mathbb{R}^{m+n}$ *is implementable by a strict (resp., activated) one-bit neural network of size* $(L, N_1 + N_2, P_1 + P_2)$.

Now we state the key lemma with its proof. This lemma states that any composition of one-bit neural networks can be also implemented by one-bit neural networks by using one more layer, constant factor more nodes and parameters.

Lemma 4.4.11. *If* $g_1 : \mathbb{R}^d \to \mathbb{R}^k$ *is implementable by a strict one-bit neural network of size* (L_1, N_1, P_1) *and* $g_2 : \mathbb{R}^k \to \mathbb{R}^n$ *is implementable by a strict (resp., activated) one-bit neural network of size* (L_2, N_2, P_2)*, then* $g_2 \circ g_1 : \mathbb{R}^d \to \mathbb{R}^n$ *is implementable by a strict (resp., activated) one-bit neural network of size* $(L_1 + L_2 + 1, N, P)$ *with* $N \leq CkN_1 + N_2$ *and* $P \leq CkN_1 + P_1 + P_2$ *for some absolute constant* $C > 0$.

Proof. We use the following observation. Let $Q_1 \in \mathbb{R}^{k\times l}$ and $Q_2 \in \mathbb{R}^{m\times k}$ be matrices with entries $\{-1, 0, 1\}$. Then, Q_2Q_1 is a $m \times l$ matrix where all entries are integers with magnitude at most k. Therefore, there exists a $m \times kl$ matrix with its entries $\{-1, 0, 1\}$ such that for any $x \in \mathbb{R}^l$, $Q_2Q_1x = Qy$ holds where $y \in \mathbb{R}^{kl}$ is a stack of k copies of x.

Now we return to the problem. Consider a strict one-bit neural network $g_1(x) = W_{L_1} \circ \sigma \circ \cdots \circ \sigma \circ W_1 \circ x$ where $W_i \circ z = A_i z + b_i$ is a one-bit affine function. For convenience, we shall write $A_{L_1} = A'$ and $b_{L_1} = d$. Then A' is a one-bit $k \times l$ matrix for some $l \geq 1$ and d is a one-bit $k \times 1$ vector. We denote $u(x) = \sigma \circ \cdots \circ \sigma \circ W_1 \circ x$. Note that u is an activated one-bit neural network having the same structure with g_1 until the $L_1 - 1$ layer. Since it is activated, u is always nonnegative.

Let W' be the one-bit affine function which represents the first layer of one-bit neural network implementation of g_2. Denote $W' \circ y = Ay + b$ with one-bit $m \times k$ matrix A and one-bit $m \times 1$ vector b.

From the above observation, one can find a one-bit matrix $E \in \mathbb{R}^{m\times kl}$ such that $AA'u = Eu^k$, where $u^k \in \mathbb{R}^{kl}$ is a stack of k copies of u. Similarly, Ad is a vector $m \times 1$ where all the entries are integers with a maximum magnitude k. Hence, there exists $m \times k$ one-bit matrix F such that $Fe = Ad$ for $k \times 1$ one-hot vector e.

Since u is nonnegative, by Lemma 4.4.2 and Lemma 4.4.10, $u \to u^k$ can be implemented by a 2-layered activated one-bit neural network with at most Ckl nodes and parameters. By Lemma 4.4.3, $u \to 1$ can be implemented by a 2-layered activated one-bit neural network with constant nodes and parameters. Therefore, by Lemma 4.4.10, $u \to (u^k, e)$ can be implemented by a 2-layered activated one-bit neural network with at most $C(kl + k)$ nodes and parameters. Denote this activated neural network as $\sigma \circ R \circ \sigma \circ S$ for one-bit affine functions R and S.

Now, define $Q = \left[\begin{array}{c|c} E & F \end{array}\right]$ to be a $m \times (kl+k)$ one-bit matrix, stacking E and F horizontally. Then we have

$$Q\big(\sigma \circ R \circ \sigma \circ S \circ u\big) = Eu^k + Fe = AA'u + Ad = A(A'u + d).$$

Therefore we have

$$W' \circ g_1(x) = A(g_1(x)) + b = A(A'u(x) + d) + b = Q(\sigma \circ R \circ \sigma \circ S \circ \sigma \circ W_{L_1-1} \cdots \circ \sigma \circ W_1 \circ x) + b.$$

Now we take the exact same nodes and parameters from the second layers of g_2. Then we obtain a one-bit neural network that implements $g_2 \circ g_1$. This proves the existence.

It is straight forward to check the number of layers is $L_1 + L_2 + 1$. The numbers of additional nodes and parameters that we used are at most Ckl. Since $l \leq N_1$ holds, the total numbers of nodes and parameters that we used are bounded by $CkN_1 + N_2$ and $CkN_1 + P_1 + P_2$ respectively.

□

4.4.4 Regularity considerations

Recall that our strategy involves decomposing the target function as$f = \sum_{i=1}^{C_{\mathcal{M}}} f\rho_i$, and then finding a one-bit approximation for each localized component. We denote $f_i = f\rho_i$ henceforth. Since ρ_i's are C^∞ functions and multiplication with smooth functions preserves the regularity, it follows that each f_i inherits the same regularity as the original target function f.

Lemma 4.4.12 (Lemma 2 in [19]). *$f_i = f\rho_i$ is s times continuously differentiable, and there exists a*

Holder coefficient L_i that depends only on $d, s, \alpha, r, P_i, \rho_i$, and ϕ_i, such that for any $|\mathbf{s}| = s$, we have

$$\left|D^{\mathbf{s}}(f_i \circ \phi_i^{-1})\big|_{\phi_i(x_1)} - D^{\mathbf{s}}(f_i \circ \phi_i^{-1})\big|_{\phi_i(x_2)}\right| \leq L_i \|\phi_i(x_1) - \phi_i(x_2)\|_2^{\alpha}, \quad \forall x_1, x_2 \in U_i.$$

Next, we introduce a slight variant of Theorem 4.1.1. We consider a restricted target function space that is Hölder continuous.

Lemma 4.4.13. *For any $s, d \in \mathbb{N}$ and $\alpha \in (0, 1]$, consider the set $C^{s,\alpha}([0,1]^d)$ which is the collection of f such that $\|f\|_{C^s} \leq 1$ and $\left|D^{\mathbf{s}} f(x) - D^{\mathbf{s}} f(y)\right| \leq \|x - y\|^{\alpha}$ for $|\mathbf{s}|_1 = s$. For all $0 < \epsilon < 1/2$, there is a set $\mathcal{N}$ of strict one-bit networks such that the following hold. They all have the same structure, at most $C_{d,s,\alpha} \log_2(1/\epsilon)$ layers, at most $C_{d,s,\alpha} \epsilon^{-d/(s+\alpha)} (\log_2(1/\epsilon))^2$ many nodes and parameters, and*

$$\sup_{f \in C^{s,\alpha}([0,1]^d)} \inf_{g \in \mathcal{N}} \|f - g\|_{\infty} \leq \epsilon.$$

Proof. For $x = (x_1, \cdots, x_d) \in \mathbb{R}^d$, define $\varphi(x) = \prod_{i=1}^d \max\{1 - |x_i|, 0\}$. For a positive integer $n \geq 1$ and $\mathbf{0} \leq \mathbf{k} \leq \mathbf{n}$, define the function $\varphi_{n,\mathbf{k},\mathbf{r}} : \mathbb{R}^d \to \mathbb{R}$ by $\varphi_{n,\mathbf{k},\mathbf{r}}(x) = (x - \frac{\mathbf{k}}{n})^{\mathbf{r}} \varphi(nx - \mathbf{k})$. It is straightforward to check each $\varphi_{n,\mathbf{k},\mathbf{r}}$ is compactly supported in a cube of side length $2/n$ with center $\mathbf{k}/n$ and $\{\varphi(n \cdot - \mathbf{k})\}_{\mathbf{0} \leq \mathbf{k} \leq \mathbf{n}}$ forms a nonnegative partition of unity for the unit cube $[0, 1]^d$.

Now for $f \in C^{s,\alpha}([0,1]^d)$, let

$$f_{n,r} = \sum_{\mathbf{0} \leq \mathbf{k} \leq \mathbf{n}} \sum_{0 \leq |\mathbf{r}|_1 \leq s} \frac{\partial^{\mathbf{r}} f(\frac{\mathbf{k}}{n})}{\mathbf{r}!} \varphi_{n,\mathbf{k},\mathbf{r}}.$$

Then we have

$$\begin{aligned}
\max_{x\in[0,1]^d}\Big|f(x)-f_{n,s}(x)\Big| &= \max_{\mathbf{x}\in[0,1]^d}\Big|\sum_{\mathbf{0}\le\mathbf{k}\le\mathbf{n}}\Big(f(x)-\sum_{0\le|\mathbf{r}|_1\le s}\frac{\partial^{\mathbf{r}}f(\frac{\mathbf{k}}{n})}{\mathbf{r}!}(x-\frac{\mathbf{k}}{n})^{\mathbf{r}}\Big)\varphi(nx-\mathbf{k})\Big| \\
&\le \max_{x\in[0,1]^d}2^d\sup_{\mathbf{k}:\|x-\frac{\mathbf{k}}{n}\|_\infty<\frac{1}{n}}\Big|f(x)-\sum_{0\le|\mathbf{r}|_1\le s}\frac{\partial^{\mathbf{r}}f(\frac{\mathbf{k}}{n})}{\mathbf{r}!}(x-\frac{\mathbf{k}}{n})^{\mathbf{r}}\Big| \\
&\le 2^d\max_{x\in[0,1]^d}\sup_{\mathbf{k}:\|x-\frac{\mathbf{k}}{n}\|_\infty<\frac{1}{n}}\sum_{\mathbf{r}:|\mathbf{r}|_1=s}\frac{1}{\mathbf{r}!}\Big|(x-\frac{\mathbf{k}}{n})^{\mathbf{r}}\Big|\times\max_{|\mathbf{r}|_1=s}\Big|D^{\mathbf{r}}f\big|_{\mathbf{y}}-D^{\mathbf{r}}f\big|_{\frac{\mathbf{k}}{n}}\Big| \\
&\le 2^d\max_{x\in[0,1]^d}\sup_{\mathbf{k}:\|x-\frac{\mathbf{k}}{n}\|_\infty<\frac{1}{n}}\sum_{\mathbf{r}:|\mathbf{r}|_1=s}\frac{1}{\mathbf{r}!}\Big|(x-\frac{\mathbf{k}}{n})^{\mathbf{r}}\Big|\times(\sqrt{d}\frac{1}{n})^{\alpha} \\
&\le 2^d d^{\frac{\alpha}{2}}(\frac{1}{n})^{s+\alpha}\sum_{\mathbf{r}:|\mathbf{r}|=s}\frac{1}{\mathbf{r}!}=\frac{2^{d+s}d^{\frac{\alpha}{2}}}{s!}(\frac{1}{n})^{s+\alpha},
\end{aligned}$$

where $\mathbf{y}$ is the linear interpolation of x and $\frac{\mathbf{k}}{n}$, determined by the Taylor remainder. Now choose the smallest n that is a power of 2 such that $\frac{2^{d+s}d^{\frac{\alpha}{2}}}{s!}(\frac{1}{n})^{s+\alpha}\le\frac{\epsilon}{3}$. The remaining proof is exactly the same as Theorem 4.1.1. □

4.4.5 Estimate under geometric constraints

The proof of Theorem 4.3.1 necessitates a uniform bound on the magnitude of the localized functions $f_i = f\rho_i$ over specific sub-regions of U_i. The following lemma provides the requisite quantitative estimates for these components.

Lemma 4.4.14 (Lemma 8 in [20]). *Define $\mathcal{K}_{i,\Delta} = \{x \in \mathcal{M} : r^2 - \Delta \le \|x - c_i\|^2 \le r^2\}$ for $i = 1, \cdots, C_{\mathcal{M}}$. Then, there exists a constant $c > 0$ that depends only on the upper bounds of the first derivatives of ρ_i's and ϕ_i's, such that*

$$\max_{x\in K_{i,\Delta}}|f_i(x)| \le \frac{c(\pi+1)}{r(1-r/\tau)}\Delta.$$

4.5 Proof of the Main Theorem

OWe prove Theorem 4.3.1 by utilizing the framework of [19] and the technical lemmas established above. We decompose the target function $f \in \mathcal{H}^{s,\alpha}(\mathcal{M})$ as

$$f = \sum_{i=1}^{C_{\mathcal{M}}} f_i, \quad f_i = f\rho_i.$$

Note that $f_i = f_i \mathbb{1}_{U_i}$ holds due to the compact support of ρ_i in U_i. The remainder of the proof is structured as follows: in Section 4.5.1, we construct one-bit neural network approximations for the indicator functions. Section 4.5.2 then focuses on the one-bit neural network approximation of each localized component f_i. Finally in Section 4.5.3, Lemma 4.4.8 is applied to synthesize these components into a unified one-bit neural network that approximates the target function f, while quantifying the resulting one-bit neural network complexity as a function of the desired accuracy ϵ in the supremum norm.

4.5.1 Chart determination

In this section, we aim to establish one-bit neural network approximations for the indicator functions $\mathbb{1}_{U_i}$'s. We begin by constructing one-bit neural network approximations of the squared Euclidean distance from c_i's. Let c_{ij} be the jth entry of $c_i \in \mathbb{R}^D$. By Lemma 4.4.6, there exist strict one-bit neural networks $g_{ij} : \mathbb{R} \to \mathbb{R}$ such that

1. They have the same structure that depends only on b,

2. Each g_{ij} has at most Cb layers,

3. Each g_{ij} has at most Cb^2 nodes and parameters,

4. $\left|g_{ij}(z) - \frac{(z-c_{ij})}{2B}\right| \leq 2^{-b+1}(|z|+1)$ for $z \in \mathbb{R}$.

Note that this implies

1. $\left|g_{ij}(z) - \frac{(z-c_{ij})}{2B}\right| \leq 2^{-b+1}(|z|+1) \leq 2^{-b}(B+2) \leq \frac{\nu}{2}$,

2. $\left|g_{jj}(z) + \frac{(z-c_{ij})}{2B}\right| \leq \left|g_{ij}(z) - \frac{(z-c_{ij})}{2B}\right| + \left|\frac{z-c_{ij}}{B}\right| \leq 2^{-b}(B+2) + 1 \leq 2$,

3. $\left|g_{ij}(z)\right| \leq \left|g_{ij}(z) - \frac{(z-c_{ij})}{2B}\right| + \left|\frac{z-c_{ij}}{2B}\right| \leq \frac{\nu}{2} + \frac{1}{2} \leq 1$,

for $z \in \left[-\frac{B}{2}, \frac{B}{2}\right]$ and $b = \lceil \log_2(\frac{2B+4}{\nu}) \rceil$, where $\nu < 1$ is chosen later.

By Lemma 4.4.2 and Lemma 4.4.11, $z \mapsto |g_{ij}(z)|$ can be implemented by an activated one-bit neural network with at most Cb layers and at most Cb^2 nodes and parameters, and the structure of this network depends only on b.

Choose S_ν^+ in Lemma 4.4.7. Then, by Lemma 4.4.9, $z \mapsto S_\nu^+(|g_j(z)|)$ can be implemented by an activated one-bit neural network with at most $Cb + C\log_2(1/\nu)$ layers and at most $Cb^2 + C\log_2(1/\nu)$ nodes and parameters, and its architecture depends only on b and ν.

Now, define $h_{i,b,\nu} : \mathbb{R}^D \to \mathbb{R}$ by $h_{i,b,\nu}(x) = 4B^2 \sum_{j=1}^D S_\nu^+(|g_{ij}(x_j)|)$, where x_j is jth entry of $x \in \mathbb{R}^D$. This map can be implemented by an activated one-bit neural network with at most $Cb + C\log_2(1/\nu)$ layers and at most $CB^2D(b^2 + \log_2(1/\nu))$ nodes and parameters, with its architecture depending only on b, B, D, and ν.

Then for any $x \in \mathcal{M}$, we have

$$|h_{i,b,\nu}(x) - \|x - c_i\|^2| = |h_{i,b,\nu}(x) - \sum_{j=1}^D (x - c_{ij})^2| = |h_{i,b,\nu}(x) - 4B^2 \sum_{j=1}^D (\frac{x_j - c_{ij}}{2B})^2|$$

$$
\begin{aligned}
&\leq 4B^2 \sum_{j=1}^{D} |S_\nu^+(|g_j(x_j)|) - (\frac{x_j - c_{ij}}{2B})^2| \\
&\leq 4B^2 \sum_{j=1}^{D} \big(|S_\nu^+(|g_j(x_j)|) - g_j(x_j)^2| + |g(x_j)^2 - (\frac{x_j - c_{ij}}{2B})^2|\big) \\
&\leq 4B^2 \sum_{j=1}^{D} \big(\nu + 2 \times \frac{\nu}{2}\big) \\
&\leq 8B^2 D\nu.
\end{aligned}
$$

Now, we construct a strict one-bit neural network approximation of the indicator function $\mathbb{1}_{U_i}$. For $\epsilon > 0$, let

$$
\theta = \lceil \log_2 \big(\frac{3r(1-\frac{r}{\tau})C_{\mathcal{M}}}{2c(\pi+1)} \times \frac{1}{\epsilon}\big) \rceil, \quad \gamma = 2^{-\theta}. \tag{4.2}
$$

where $c = c_{\mathcal{P},\Phi,\mathcal{R}} > 0$ is constant in Lemma 4.4.14.

Define

$$
T_\gamma(x) = \frac{2}{\gamma - 16B^2 D\nu}\sigma(-x + r^2 - 8B^2 D\nu) - \frac{2}{\gamma - 16B^2 D\nu}\sigma(-x + r^2 - \gamma + 8B^2 D\nu). \tag{4.3}
$$

It is straightforward to check that $0 \leq T_r(x) \leq 1$ for $x \in \mathbb{R}$. We choose $\nu = \frac{\gamma}{32B^2 D}$. Then, (4.3) can be rewritten as

$$
T_\gamma(x) = 2^{\theta+2}\sigma(-x + 2^{-2\beta} - 2^{-(\theta+2)}) - 2^{\theta+2}\sigma(-x + 2^{-2\beta} - 3 \times 2^{-(\theta+2)}). \tag{4.4}
$$

Our choice of $\nu = \frac{\gamma}{32B^2D} = \frac{1}{32B^2D}2^{-\theta}$ gives us

$$b = \lceil \log_2(\frac{2B+4}{\nu}) \rceil = \theta + \lceil \log_2(64B^2D(B+2)) \rceil. \tag{4.5}$$

Therefore, $h_{i,b,\nu}$ can be implemented by an activated one-bit neural network with at most $C(\theta + \log_2(B^3D))$ layers and at most $CB^2D(\theta^2 + (\log_2(B^3D))^2)$ nodes and parameters, and its architecture depends only on B, D, and θ. Note that $x \mapsto 2^m$ can be implemented by an activated one-bit neural network with size at most Cm. Then, $x \mapsto -h_{i,b,\nu}(x) + 2^{-2\beta} - 2^{-(\theta+2)}$ can be implemented by a strict one-bit neural network with at most $C(\theta + \log_2(B^3D))$ layers and at most $CB^2D(\theta^2 + (\log_2(B^3D))^2)$ nodes and parameters, and its architecture depends only on B, D, and θ. Now by Lemma 4.4.4 and Lemma 4.4.9, $x \mapsto 2^{\theta+2}\sigma(-h_{i,b,\nu}(x) + 2^{-2\beta} - 2^{-(\theta+2)})$ can be implemented by an activated one-bit neural network with at most $C(\theta + \log_2(B^3D))$ layers and at most $CB^2D(\theta^2 + (\log_2(B^3D))^2)$ nodes and parameters, and its architecture depends only on B, D, and θ. Similarly, $x \mapsto 2^{\theta+2}\sigma(-h_{i,b,\nu}(x) + 2^{-2\beta} - 3 \times 2^{-(\theta+2)})$ can be implemented by an activated neural network with the same layers of $x \mapsto 2^{\theta+2}\sigma(-h_{i,b,\nu}(x) + 2^{-2\beta} - 2^{-(\theta+2)})$, and at most $CB^2D(\theta^2 + (\log_2(B^3D))^2)$ nodes and parameters, and its architecture depends only on B, D, and θ.

Therefore, using (4.2), we can conclude that $T_\gamma(h_{i,b,\nu}(\cdot))$'s can be implemented by strict one-bit neural networks such that

1. They have the same structure for $i = 1, \cdots, C_{\mathcal{M}}$,
2. The architecture depends only on $B, D, \tau, \mathcal{P}, \Phi, \mathcal{R}$ and $C_{\mathcal{M}}$,
3. They have at most $C_{B,\tau,\mathcal{P},\Phi,\mathcal{R},C_{\mathcal{M}}}(\log_2(1/\epsilon) + \log_2 D)$ layers,
4. They have at most $C_{B,\tau,\mathcal{P},\Phi,\mathcal{R},C_{\mathcal{M}}}\big(D(\log_2(1/\epsilon))^2 + D(\log_2 D)^2\big)$ nodes and parameters.

From the construction, if $x \notin U_i$, which is equivalent to $\|x - c_i\|^2 \geq r^2$, then $h_{i,b,\nu}(x) \geq r^2 - 8B^2D\nu$. Therefore, $T_\gamma(h_{i,b,\nu}(x)) = 0$. If $x \in U_i$ and $\|x-c_i\|^2 \leq r^2-\gamma$, then $h_{i,b,\nu}(x) \leq r^2-\gamma+8B^2D\nu$ holds. Thus, it is straightforward to check $T_\gamma(h_{i,b,\nu}(x)) = 1$. We use this strict one-bit quantized neural network as the indicator function on U_i.

4.5.2 Taylor Expansion

In this section, we construct one-bit neural network approximations of f_i's. As in [19, Appendix B.2], we extend $f_i \circ \phi_i^{-1}$ to the whole cube $[0,1]^d$ by assigning $f_i \circ \phi_i^{-1}(x) = 0$ for $x \in [0,1]^d \setminus \phi_i(U_i)$. Since f_i vanishes on the complement of the support of ρ_i, it is easy to check that this extension preserves the regularity. We denote f_i^ϕ as this extension. Then by the Lemma 4.4.12, we still have the same estimate

$$\left|D^{\mathbf{s}} f_i^\phi\big|_{x_1} - D^{\mathbf{s}} f_i^\phi\big|_{x_2}\right| \leq L_i\|x_1 - x_2\|_2^\alpha, \quad \forall x_1, x_2 \in [0,1]^d.$$

Given $f \in \mathcal{H}^{s,\alpha}$, there exists a normalizing constant L that depends only on $d, s, \alpha, r, \mathcal{P}, \Phi$ and $\mathcal{R}$, such that $\frac{f_i^\phi}{L} \in C^{s,\alpha}([0,1]^d)$ for all $i = 1, \cdots, C_{\mathcal{M}}$, where $C^{s,\alpha}([0,1]^d)$ is the set defined in Lemma 4.4.13. Consequently, the application of Lemma 4.4.13 for these scaled functions $\frac{f_i^\phi}{L} \in C^{s,\alpha}([0,1]^d)$, followed by the use of Lemma 4.4.5 and Lemma 4.4.11, yields strict one-bit neural networks $g_1^\phi, \cdots, g_{C_{\mathcal{M}}}^\phi : [0,1]^d \to \mathbb{R}$ that satisfy the following conditions:

1. $\|g_i^\phi - f_i^\phi\|_\infty \leq \delta$ in $[0,1]^d$ for all $i = 1, \cdots, C_{\mathcal{M}}$,
2. $g_1^\phi, \cdots, g_{C_{\mathcal{M}}}^\phi$ have the same structure that depends only on δ, s, α, and L,
3. The structure has at most $C_{d,s,\alpha,L} \log_2(1/\delta)$ layers and at most $C_{d,s,\alpha,L}\delta^{-d/(s+\alpha)}(\log_2(1/\delta))^2$ many nodes and parameters.

Note that $f_i^\phi = f_i \circ \phi_i^{-1} = (f \circ \phi_i^{-1}) \times (\rho_i \circ \phi_i^{-1})$. Since $\{P_i\}_{i=1,\cdots,C_\mathcal{M}}, \{\rho_i\}_{i=1,\cdots,C_\mathcal{M}}$, and $\{\phi_i\}_{i=1,\cdots,C_\mathcal{M}}$ are fixed from the beginning, using $f \in H^{s,\alpha}$, we can conclude that there exists a positive constant G such that $\|\nabla f_i^\phi\| \leq G$ for all $i = 1, \cdots, C_\mathcal{M}$, such that G depends only on $\{P_i\}_{i=1,\cdots,C_\mathcal{M}}, \{\rho_i\}_{i=1,\cdots,C_\mathcal{M}}$ and $\{\phi_i\}_{i=1,\cdots,C_\mathcal{M}}$.

Recall $\phi_i(x) = b_i(V_i^T(x - c_i) + u_i)$ for any $x \in U_i$. Define $\kappa \in \mathbb{N}$ as the smallest positive integer such that

$$\max_{i=1,\cdots,C_\mathcal{M}} \left\{\|b_i V_i^T\|_\infty, \|b_i(-V_i^T c_i + u_i)\|_\infty\right\} \leq 2^\kappa.$$

Note that κ depends only on ϕ_i's. Since ϕ_i's are affine maps, by Lemma 4.4.6, there exist strict one-bit neural networks $\psi_i : \mathbb{R}^D \to \mathbb{R}^d$ such that

1. They have the same structure that depends only on κ, d, B, D, G and δ,
2. Each ψ_i has at most $C(\kappa + \lceil \log_2(\frac{(BD+2)G}{\delta}) \rceil)$ layers,
3. Each ψ_i has at most $C(\kappa + \lceil \log_2(\frac{(BD+2)G}{\delta}) \rceil)^2 d$ nodes and parameters,
4. $\|\psi_i(x) - \phi_i(x)\|_\infty \leq \frac{\delta}{G} \leq \frac{1}{3}$ for $x \in U_i$.

Therefore, by the triangle inequality, $\psi_i(U_i) \subset [0,1]^d$ holds for $i = 1. \cdots, C_\mathcal{M}$.

the

Now, define

$$\hat{f}_i = g_i^\phi \circ \psi_i. \tag{4.6}$$

Then we have

$$|f_i(x) - \hat{f}_i(x)| = |(f_i^\phi \circ \phi_i)(x) - (g_i^\phi \circ \psi_i)(x)|$$
$$\leq |(f_i^\phi \circ \phi_i)(x) - (f_i^\phi \circ \psi_i)(x)| + |(f_i^\phi \circ \psi_i)(x) - (g_i^\phi \circ \psi_i)(x)|. \tag{4.7}$$

Therefore by the mean value inequality, we have

$$|(f_i^\phi \circ \phi_i)(x) - (f_i^\phi \circ \psi_i)(x)| \leq G\|\phi_i(x) - \psi_i(x)\| \leq \delta. \tag{4.8}$$

Since $\psi_i(x) \in [0,1]^d$, the second term in (4.7) satisfies

$$|(f_i^\phi \circ \psi_i)(x) - (g_i^\phi \circ \psi_i)(x)| \leq \delta. \tag{4.9}$$

Hence, from (4.7),(4.8),and (4.9), $\hat{f}_i$ is an approximation of f_i with $\|f_i - \hat{f}_i\|_\infty \leq 2\delta$.

Recall that ψ_i's are strict one-bit neural networks with the same structure that depends only on $\kappa, d, B, D, G, \delta$, with at most $C(\kappa + \lceil \log_2(\frac{(BD+2)G}{\delta})\rceil)$ layers and at most $C(\kappa + \lceil \log_2(\frac{(BD+2)G}{\delta})\rceil)^2 d$ nodes and parameters. g_i^ϕ's are strict one-bit neural networks with the same structure that depends only on $\delta, s, \alpha, L,$, with at most $C_{d,s,\alpha,L} \log_2(1/\delta)$ layers and at most $C_{d,s,\alpha,L}\delta^{-d/(s+\alpha)}(\log_2(1/\delta))^2$ many nodes and parameters. The parameters κ, B, G, and L depend only on $\mathcal{M}, \Phi, \mathcal{P}$, and $\mathcal{R}$ that we fixed from our setting. Hence, by Lemma 4.4.11, $\hat{f}_i = g_i^\phi \circ \psi_i$'s can be implemented by strict one-bit neural networks such that

1. They have the same structure independent of f,

2. They have at most $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}(\log_2 D + \log_2(1/\delta))$ layers,

3. They have at most $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}\left(\delta^{-d/(s+\alpha)}(\log_2(1/\delta))^2+\log_2 D\log_2(1/\delta)+(\log_2 D)^2\right)$ nodes and parameters.

4.5.3 Estimating the total error

Define $\hat{f}=\sum_{i=1}^{C_{\mathcal{M}}}\hat{\times}(\hat{f}_i,T_\gamma(h_{i,b,\nu}))$, where $\hat{\times}$ is a strict one-bit neural network in Lemma 4.4.8 that satisfies

$$|\hat{\times}(x,y)-xy|\leq\eta,$$

for $x,y\in[-(1+2\delta),1+2\delta]$. For each i, we have

$$\|f_i-\hat{\times}(\hat{f}_i,T_\gamma(h_{i,b,\nu}))\|_\infty\leq\|f_i-\hat{f}_i\times T_\gamma(h_{i,b,\nu})\|_\infty+\|\hat{f}_i\times T_\gamma(h_{i,b,\nu})-\hat{\times}(\hat{f}_i,T_\gamma(h_{i,b,\nu}))\|_\infty.$$

Note that $f_i=f_i\mathbb{1}_{U_i}$. Therefore the first term can be decomposed as

$$\|f_i-\hat{f}_i\times T_\gamma(h_{i,b,\nu})\|_\infty\leq\|(f_i-\hat{f}_i)T_\gamma(h_{i,b,\nu})\|_\infty+\|f_i(\mathbb{1}_{U_i}-T_\gamma(h_{i,b,\nu}))\|_\infty.$$

Since $0\leq T_\gamma\leq 1$, we have $\|(f_i-\hat{f}_i)T_\gamma(h_{i,b,\nu})\|_\infty\leq\|f_i-\hat{f}_i\|_\infty\leq 2\delta$.

Recall that from the construction, we have (1) If $\|x-c_i\|^2>r^2$, then $T_\gamma(h_{i,b,\nu}(x))=0=\mathbb{1}_{U_i}(x)$ and (2) If $x\in U_i$ and $\|x-c_i\|^2<r^2-\gamma$, then $T_\gamma(h_{i,b,\nu}(x))=1=\mathbb{1}_{U_i}(x)$. Therefore by Lemma 4.4.14, we have

$$\|f_i(\mathbb{1}_{U_i}-T_\gamma(h_{i,b,\nu}))\|_\infty\leq 2\max_{x\in\mathcal{K}_{\rangle,\gamma}}|f_i(x)|\leq\frac{2c(\pi+1)}{r(1-r/\tau)}\gamma.$$

Adding all the inequalities, we have

$$\begin{aligned}\|f-\hat{f}\|_\infty &\le \sum_{i=1}^{C_{\mathcal{M}}} \|f_i - \hat{\times}(\hat{f}_i, T_\gamma(h_{i,b,\nu}))\|_\infty \\ &\le C_{\mathcal{M}}\Big(\eta + 2\delta + \frac{2c(\pi+1)}{r(1-r/\tau)}\gamma\Big) \\ &\le C_{\mathcal{M}}(\eta+2\delta) + \frac{\epsilon}{3}.\end{aligned}$$

We take $\eta = \frac{\epsilon}{3C_{\mathcal{M}}}$ and $\delta = \frac{\epsilon}{6C_{\mathcal{M}}}$. Then, $\hat{f}$ satisfies $\|f-\hat{f}\|_\infty \le \epsilon$. Now, it remains to verify that $\hat{f}$ can be implemented by one-bit neural network where the structure is independent of f, and count the number of layers, nodes, and parameters.

Lemma 4.4.1 states that $z \mapsto z \in \mathbb{R}$ can be implemented by a strict one-bit neural network with 2 layers and constant size of nodes and parameters. Also, Lemma 4.4.2 states that $z \mapsto |z| \in \mathbb{R}$ can be implemented by activated one-bit neural networks with 2 layers and 3 layers, both having constant size of nodes and parameters. Hence, by Lemma 4.4.11 and using the fact that $T_\gamma(\cdot) \in [0,1]$, $x \mapsto \hat{f}_i$ and $x \mapsto T_\gamma(h_{i,b,\nu})$ can be implemented by strict one-bit neural networks having the same number of layers, where

1. The number of layers is at most $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}(\log_2 D + \log_2(1/\epsilon))$,
2. Strict one-bit neural network that implements $\hat{f}_i$ has at most $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}\Big(\epsilon^{-d/(s+\alpha)}(\log_2(1/\epsilon))^2 + \log_2 D \log_2(1/\epsilon) + (\log_2 D)^2\Big)$ nodes and parameters,
3. Strict one-bit neural network that implements $T_\gamma(h_{i,b,\nu})$ has at most $C_{B,\tau,\mathcal{P},\Phi,\mathcal{R},C_{\mathcal{M}}}\big(D(\log_2(1/\epsilon))^2 + D(\log_2 D)^2\big)$ nodes and parameters.

This implies the map $x \mapsto (\hat{f}_i, T_\gamma(h_{i,b,\nu}))$ can be implemented by a strict one-bit neural network

with at most $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}(\log_2 D + \log_2(1/\epsilon))$ layers and $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}\Big(\epsilon^{-d/(s+\alpha)}(\log_2(1/\epsilon))^2 + D(\log_2(1/\epsilon))^2 + D(\log_2 D)^2\Big)$ nodes and parameters. Since B, τ and $C_{\mathcal{M}}$ depend on $\mathcal{M}$, we simplified the constant above.

From Lemma 4.4.8, $\hat{\times}(\cdot,\cdot) : [-1-2\delta, 1+2\delta] \to \mathbb{R}$ can be implemented by a strict one-bit neural network with size at most $C_{C_{\mathcal{M}}} \log_2(1/\epsilon)$. Therefore, by Lemma 4.4.11, each $\hat{\times}(\hat{f}_i, T_\gamma(h_{i,b,\nu}))$ can be implemented by strict neural networks of the same structure with at most $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}(\log_2 D + \log_2(1/\epsilon))$ layers and $C_{d,s,\alpha,\mathcal{M},\Phi,\mathcal{P},\mathcal{R}}\Big(\epsilon^{-d/(s+\alpha)}(\log_2(1/\epsilon))^2 + D(\log_2(1/\epsilon))^2 + D(\log_2 D)^2\Big)$ nodes and parameters.

Recall from (4.1) that in the last layer, there is no bias applied. Note that in Lemma 4.4.11, the last layer remains the same in the composition. Therefore, adding $C_{\mathcal{M}}$ terms of $\hat{\times}(\hat{f}_i, T_\gamma(h_{i,b,\nu}))$ can be implemented by a strict one-bit neural network. Note that the construction of one-bit neural networks is independent of the choice of $f \in \mathcal{H}^{s,\alpha}$. This completes the proof.

4.6 Conclusion and Open Questions

In this chapter, we established Theorem 4.3.1, which constitutes the first rigorous treatment of the universal approximation theory for one-bit neural networks approximating the Hölder function class on a d-dimensional compact Riemannian manifold $\mathcal{M}$ isometrically embedded in $\mathbb{R}^D$. The derivation of this result relies on a series of technical lemmas that are of independent interest. Specifically, Theorem 4.3.1 demonstrates that the exponential dependence on the approximation error ϵ, in terms of the complexity of layers, nodes, and parameters, is governed solely by the intrinsic dimension d and the regularity of the Hölder function class. Furthermore, we quantified the weak dependence of the network size on the ambient dimension D. The importance of Theorem 4.3.1 lies in its ability to

provide a theoretical foundation and mathematical explanation for the performance of quantized neural networks on high-dimensional data that possess an intrinsic low-dimensional geometry.

Based on our results, we propose several open problems.

Alternative Activation Functions : While the approximation capabilities of one-bit neural networks utilizing quadratic and ReLU activations were established in [46], the theoretical landscape for other activation functions remains largely unexplored. This line of inquiry is expected to identify the specific properties of activations that are most conducive to, or even optimized for, quantized environments, potentially leading to the design of specialized activations for one-bit architectures.

Architecture Complexity and Approximation Rates : Moving beyond existence results, the precise approximation rates of one-bit neural networks relative to their depth and width need rigorous investigation. This endeavor necessitates identifying the minimal width and parameter complexity required to approximate a given function class under prescribed depth constraints. Such investigations are crucial for elucidating the scaling laws of quantized architectures and providing a robust theoretical road map for the design of resource-efficient one-bit models.

Extension to Transformer-based Models : Unlike traditional FNN architectures, the transformer utilizes self-attention to process global information, posing new challenges for approximation theory. Extending our framework for one-bit neural networks to these attention-based models is a promising avenue for future work. Such an extension would provide critical insights into the expressive capacity of quantized transformers when approximating functions on complex manifolds.

Chapter 5: Curse of Dimensionality in Neural Network Optimization

5.1 Introduction

The curse of dimensionality refers to the exponential growth of computational complexity or data requirements with respect to the dimension of the computation or input space. This phenomenon arises in various fields, including Nearest Neighbor algorithms [99, Chapter 19], numerical methods for solving partial differential equations [3], and kernel-based methods [116]. It is also observed in the theory of artificial neural networks, particularly in approximation theory [29, 39, 77, 101, 104, 127, 128] and generalization theory [62, 67, 131].

The significance of the curse of dimensionality extends beyond infeasible computational complexity and limited resources; it also restricts a model's ability to learn and generalize, particularly in high-dimensional spaces. Therefore, understanding this phenomenon and developing strategies to overcome it remains a crucial research topic. In neural network approximation and generalization theory, the theoretical analysis and design of neural network architectures or algorithms to mitigate the curse of dimensionality—whether in terms of the required number of network parameters or the necessary amount of data—are active areas of research [4, 5, 13, 15, 17, 18, 40, 67, 80, 89, 110].

The curse of dimensionality has rarely been explored in the context of neural network optimization theory, particularly concerning the computational expense of gradient descent-based training. This is largely due to the inherently challenging nature of the non-convex optimization problem. Extensive research has been dedicated to analyzing convergence properties under an over-parameterized (i.e., sufficiently wide) regime [2, 21, 32, 33, 64, 66, 86, 109, 135, 137, 138]. Most of these studies aim to establish positive results, demonstrating linear convergence to global or local minima of the empirical risk function with high probability, provided that certain assumptions on network width and training

data hold. Although a negative result was shown in [100] that exponential convergence time can occur, this result is derived from a one-dimensional linear neural network—a highly specific and atypical setting. Furthermore, in that case, the exponential dependence is only related to the depth of the neural network, instead of the dimension of the learning target.

While the curse of dimensionality in neural network optimization remains an open question, an interesting negative result is presented in [121] for shallow network training without imposing any assumptions on network width. Specifically, it is shown that there exists a Lipschitz continuous target function for which the population risk cannot decay faster than $t^{-\frac{4}{d-2}}$ under the gradient flow of either empirical risk or population risk in the mean-field regime.

Theorem 5.1.1. [121, Theorem 1] *Let the training samples be independent and identically distributed from the uniform distribution on* $[0,1]^d$. *Let* $\sigma : \mathbb{R} \to \mathbb{R}$ *be a Lipschitz continuous activation function. There exists a target function* ϕ *with its Lipschitz constant and* L^∞ *norm bounded by* 1 *such that, when a shallow neural network with activation function* σ *is trained in the mean-field regime by the gradient flow of either the population risk or the empirical risk to learn* ϕ, *then*

$$\limsup_{t\to\infty} \left[t^\gamma \|f_t - \phi\|^2_{L^2([0,1]^d)} \right] = \infty,$$

holds for all $\gamma > \frac{4}{d-2}$. *Here,* f_t *denotes the shallow neural network at training time* t.

This result suggests that, in general, when learning Lipschitz continuous functions using a shallow neural network, gradient flow training requires at least $\Omega((\frac{1}{\epsilon})^{\frac{d-2}{4}})$ units of time to achieve a population risk smaller than $\epsilon > 0$. The space of Lipschitz continuous functions is vast, making it unsurprising that a particularly challenging target function can be found within this space, leading to the curse of dimensionality in optimization.

A fundamental question, however, is whether this curse persists when considering a more restricted and structured function space. In this section, the focus is placed on smooth function spaces for two primary reasons: 1) Smooth functions frequently arise as solutions to partial differential equations (PDEs). It has been conjectured that deep learning-based PDE solvers may circumvent the curse of dimensionality associated with high-dimensional PDEs [41, 48, 91, 108]. 2) The smoothness of a learning target can introduce additional beneficial structures that could potentially mitigate learning difficulties. Therefore, it is crucial to determine whether smoothness is the key property required to overcome the curse of dimensionality. To address this question, the impact of target function smoothness on the curse of dimensionality in neural network optimization is investigated, a topic that has not been extensively explored in the literature. In particular, the results obtained align with findings in neural network approximation theory, where it is well established that, in general, a shallow neural network requires $O(\epsilon^{-\frac{d}{r}})$ neurons to approximate a function in C^r within a d-dimensional space [77, 127–129].

The answer to the fundamental question raised above can be formalized as follows.

Theorem 5.1.2. *Let the training samples be independent and identically distributed from the uniform distribution on $[0,1]^d$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a Lipschitz continuous activation function and r be a positive integer with $r < d/2$. There exists a target function $\phi \in C^r([0,1]^d)$ such that, when a shallow neural network with activation function σ is trained in the mean-field regime by the gradient flow of either the population risk or the empirical risk to learn ϕ, then*

$$\limsup_{t\to\infty} \left[t^\gamma \|f_t - \phi\|^2_{L^2([0,1]^d)} \right] = \infty,$$

holds for all $\gamma > \frac{4r}{d-2r}$. Here, f_t denotes the shallow neural network at training time t.

In the worst-case scenario, the L^2 population risk cannot decay at a rate faster than $t^{-\frac{4r}{d-2r}}$. This result suggests that, in general, when learning a r times continuously differentiable function using a shallow neural network, gradient flow training requires at least $\Omega((\frac{1}{\epsilon})^{\frac{d-2r}{4r}})$ units of time to achieve a population risk smaller than $\epsilon > 0$. For fixed ϵ and r, this quantity grows exponentially with the dimension d, illustrating the curse of dimensionality in neural network training. Note that there is no assumption on the number of training samples and the neural network width, which makes Theorem 5.1.2 holds uniformly. A more formal mathematical statement for Theorem 5.1.2 appears as Theorem 5.3.3.

Furthermore, a new problem concerning the impact of activation functions on the curse of dimensionality in neural network optimization is addressed. While most commonly used activation functions, such as the rectified linear unit (ReLU), Gaussian-error linear unit (GELU), Sigmoid, Tanh, Swish, and Sinusoid, are Lipschitz continuous, a growing body of research has focused on activation functions that do not possess this property. Examples include the quadratic activation function $\sigma(x) = x^2$, which has been utilized to analyze the optimization landscape and generalization ability of shallow neural networks [31, 97, 109], as well as the ReLU^k activation function (or Rectified Power Unit) $\sigma(x) = \max\{0, x\}^k$, which has been applied in neural network approximation theory [106, 107, 122–124] and in the study of partial differential equations [73]. Recently, an advanced activation function—comprising a combination of ReLU, the floor function $[x]$, the exponential function 2^x, and the Heaviside function $\mathbf{1}_{x\geq 0}$—was proposed in [102, 103] to enhance the approximation power of neural networks. As the study of novel activation functions continues to advance, it is natural to investigate how these functions influence the curse of dimensionality in neural network optimization. This issue has not been addressed in the literature, e.g., only Lipschitz continuous activation functions are considered in [121]. In this chapter, we settle this question for a broad family of locally Lipschitz

continuous activation functions that includes several favorable cases, such as the quadratic activation and the ReLU^k activation. The formal statement appears in Theorem 5.3.4.

The contributions of this chapter can be summarized as follows.

1. We establish in Theorem 5.3.1 that in general, $C^r([0,1]^d)$ functions with $r < d/2$ are poorly approximated by two-layer neural networks. Specifically, the optimal approximation rate in the $L^2([0,1]^d)$ topology using Barron functions with a Barron norm bounded by κ cannot exceed the rate $\kappa^{-\frac{2r}{d-2r}}$ for such functions. Note that our approximation result differs from the majority of neural network approximation theory literature, which typically expresses approximation rates in terms of the number of parameters in the network architecture. As a corollary, it is proven that $C^r([0,1]^d)$ is not contained in the Barron space when $r < d/2$. Although sufficient regularity, specifically $r > d/2+1$, is known to guarantee that a function belongs to the Barron space [7,34], no prior results have explored the relationship between function regularity and Barron spaces for lower regularity. Our findings demonstrate that regularity with $r < d/2$ is insufficient for a function to belong to the Barron space.

2. In Theorem 5.3.3, we prove that learning functions that suffer from poor approximation, as identified in Theorem 5.3.1, requires an exponential training time under gradient flow to achieve a desired accuracy, leading to the curse of dimensionality in optimization. Mathematically, under gradient flow training of two-layer neural networks in the mean field regime, the population risk cannot decay faster than $t^{-\frac{4r}{d-2r}}$, highlighting the curse of dimensionality in neural network optimization. In Theorem 5.3.3, we analyze the case of Lipschitz continuous activation functions and allow infinite-width shallow network training. If the activation function is continuously differentiable, then the gradient flow is guaranteed to exist. For piecewise differentiable activation

functions such as ReLU and others, the existence of gradient flows has been established in [22, 120]. Note that our focus is not on the existence of gradient flows.

3. Finally, in Theorem 5.3.4, we demonstrate that the curse of dimensionality in neural network optimization persists even when using locally Lipschitz continuous activation functions. Specifically, we show that in general, the population risk under gradient flow training of finite-width shallow neural network in the mean-field regime cannot decay faster than $t^{-\frac{(4+2\delta)r}{d-2r}}$ when learning a $C^r([0,1]^d)$ function using locally Lipschitz continuous activation function whose Lipschitz constant in $[-x, x]$ is bounded by $O(x^\delta)$ for any $x \in \mathbb{R}$. Notably, the activation functions $\sigma(x) = x^2$ and $\sigma(x) = \max\{0, x\}^k$ satisfy this condition.

To the best of our knowledge, this is the first mathematical work that demonstrates the influence of the target function's regularity on the curse of dimensionality in neural network training. In contrast to most works in neural network optimization, the theorems do not impose any conditions on the neural network width or the sample size. The only assumption that the data is drawn from the uniform distribution in $[0,1]^d$, a natural choice that lets us identify the population risk with one-half of the square of the L^2 norm.

The content of this chapter is based on [81][1]. While the research problem was proposed by Professor Haizhao Yang, the author of this thesis was responsible for the entire execution of the study, including the derivation of all theorems and the preparation of the manuscript.

[1] Accepted for publication in *Information and Inference: A Journal of the IMA*.

5.2 Setup

In this chapter, we additionally adopt the following notation. $Q = [0,1]^d$ denotes the unit cube in $\mathbb{R}^d$. $B'_\epsilon(x)$ denotes the projection of the ball $B_\epsilon(x)$ from $\mathbb{R}^d/\mathbb{Z}^d$ onto Q.

To illustrate this notation, consider the following examples: 1) When $d = 1$, the set $B'_{1/8}(1/16)$ is given by $B'_{1/8}(1/16) = [0, 3/16) \cup (15/16, 1]$. 2) When $d = 2$, the set $B'_{1/4}(\frac{1}{2}, 0)$ is given by $B'_{1/4}\left(\frac{1}{2}, 0\right) = \left\{(x,y) \in [0,1]^2 : \left(x - \frac{1}{2}\right)^2 + y^2 \leq \left(\frac{1}{4}\right)^2\right\} \cup \left\{(x,y) \in [0,1]^2 : \left(x - \frac{1}{2}\right)^2 + (y-1)^2 \leq \left(\frac{1}{4}\right)^2\right\}$. If a constant $\alpha \in \mathbb{R}^n$ is written as $\alpha = \alpha_{\beta_1, \cdots, \beta_m}$ for some parameters $\beta_1, \cdots, \beta_m$, this means α is a constant that depends only on $\beta_1, \cdots, \beta_m$.

Let $f : X \to \mathbb{R}$ be a function defined on a compact set $X \subset \mathbb{R}^d$. Define D_f as the collection of appropriate Borel probability measures π in the integral representation (2.15) to generate f. The Barron space consists of functions that admit the integral representation (2.15) with a finite Barron norm. To emphasize the dependence on the activation function σ, the Barron space is denoted as $B_\sigma(X)$. We only consider the Barron space where σ is a Lipschitz continuous activation function. If D_f is nonempty, the Barron norm $\|\cdot\|_{B_\sigma}(X)$ of a Barron function f is defined as follows: When σ is the ReLU activation function, the norm is given by

$$\|f\|_{B_\sigma(X)} \coloneqq \inf_{\pi \in D_f} \mathbb{E}_\pi[|a|(\|w\|_1 + |b|)] < \infty. \tag{5.1}$$

Otherwise, for other Lipschitz continuous activation functions, the norm is defined as

$$\|f\|_{B_\sigma(X)} \coloneqq \inf_{\pi \in D_f} \mathbb{E}_\pi[|a|(\|w\|_1 + |b| + 1)] < \infty. \tag{5.2}$$

For simplicity, since the primary focus is on the domain $X = Q$, the notation is further simplified by writing B_σ henceforth.

If a Borel probability measure π generates a function f through the integral representation (2.15), we denote it as f_π to emphasize its dependence of π. We assume that the data distribution is uniform on Q. Then given a target function f^*, the population risk $\mathcal{R}_p$ and the empirical risk $\mathcal{R}_n$ are

$$\mathcal{R}_p(\pi) := \frac{1}{2}\int_Q (f_\pi(x) - f^*(x))^2 dx, \quad \mathcal{R}_n(\pi) := \frac{1}{2n}\sum_{i=1}^{n}(f_\pi(x_i) - f^*(x_i))^2$$

where x_i's are independent and identically distributed training samples drawn from the uniform distribution on Q. Note that the population risk can be written as $\frac{1}{2}\|f_\pi - f^*\|^2_{L^2([0,1]^d)}$. For a Borel probability measure π, we denote its second moment as

$$N(\pi) := \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} a^2 + \|w\|^2 + b^2 \pi(da \otimes dw \otimes db).$$

We adopt the framework of [121] and investigate gradient flow training as evolution of the parameter distribution under the 2-Wasserstein gradient flow. In this framework, if training started with the initial parameter distribution π^0, the two-layer neural network at time t is described by f_{π^t}, where π^t is the parameter distribution at time t under the 2-Wasserstein gradient flow. In the finite-width training regime with m neurons, we denote the parameter distribution at time t as π_m^t to emphasize the number of neurons.

5.3 Main Theorems

Our first result establishes the existence of functions in $C^r(Q)$ that are poorly approximated by shallow neural networks. This result is formally stated in the following theorem.

Theorem 5.3.1. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a Lipschitz continuous activation function, and let r be a positive integer such that $r < d/2$. Then, there exists a function $\phi \in C^r(Q)$ satisfying*

$$\limsup_{\kappa \to \infty} \left[\kappa^\gamma \inf_{\|f\|_{B_\sigma} \leq \kappa} \|\phi - f\|_{L^p(Q)} \right] = \infty,$$

for any $\gamma > \frac{r/d}{1/2 - r/d} = \frac{2r}{d-2r}$ and any $p \in [2, \infty]$.

A direct consequence of Theorem 5.3.1 is the relationship between the smoothness of function spaces and Barron spaces, as stated in the following corollary. This follows from a simple observation: if $C^r(Q) \subset B_\sigma(Q)$, then $\limsup_{\kappa \to \infty} \left[\kappa^\gamma \inf_{\|f\|_{B_\sigma} \leq \kappa} \|\phi - f\|_{L^p(Q)} \right] = 0$ for any $\phi \in C^r(Q)$, which contradicts Theorem 5.3.1. Note that it has been established that when $r > d/2 + 1$, C^r functions belong to the Barron space with ReLU activation functions; see [7, Section 4, point 15] and [120, Corollary B.6].

Corollary 5.3.2. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a Lipschitz continuous activation function, and let r be a positive integer such that $r < d/2$. Then $C^r(Q) \not\subset B_\sigma(Q)$.*

For functions that suffer from poor approximation as stated in Theorem 5.3.1, the curse of dimensionality also manifests in the number of training steps required when they are learned by shallow neural networks. This result is formally stated in the following theorem. Note that within the framework of Section 5.2, the training dynamics of a shallow neural network can be equivalently

interpreted as the evolution of parameter measures π^t.

Theorem 5.3.3. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a Lipschitz continuous activation function and r be a positive integer with $r < d/2$. There exists a target function $\phi \in C^r(Q)$ with $\|\phi\|_{C^r} \leq 1$ such that, if the parameter measures π^t with $N(\pi^0) < \infty$ evolve by the 2-Wasserstein gradient flow of either the population or the empirical risk, then they satisfy*

$$\limsup_{t\to\infty} \left[t^\gamma \mathcal{R}_p(\pi^t)\right] = \infty,$$

for all $\gamma > \frac{4r}{d-2r}$. Here, the parameter measures π^t define integral representations 2.15 of shallow neural networks f_{π^t} with activation σ under the training to learn ϕ.

Theorem 5.3.3 holds uniformly in both the network width and the number of training data. This is due to two key reasons: 1) Theorem 5.3.1 is an approximation result related to the size of Barron norms, not to the number of parameters. 2) The growth of the second moment of the parameter measure is at most sublinear in time, which is stated in Lemma 5.4.1, and this bound does not involve the sample size.

For a certain class of locally Lipschitz activation functions, similar results hold when training finite-width shallow neural networks. This result is formally stated in the following theorem.

Theorem 5.3.4. *Let r be a positive integer with $r < d/2$, and let σ be a locally Lipschitz continuous activation function. Define L_x as the Lipschitz constant of σ on the closed interval $[-x, x]$. Assume that $L_x = O(x^\delta)$ for some $\delta \geq 0$. Then, for any positive integer m, there exists a function $\phi \in C^r(Q)$ such that if the parameter measures π_m^t, with m neurons, evolve by the 2-Wasserstein gradient flow of*

either the population or empirical risk, then they satisfy

$$\limsup_{t\to\infty} \left[t^{\gamma}\mathcal{R}_p(\pi_m^t)\right] = \infty,$$

for all $\gamma > \frac{(4+2\delta)r}{d-2r}$. *Here, the parameter measures* π_m^t *define integral representations 2.15 of shallow neural networks* $f_{\pi_m^t}$ *with activation* σ *and* m *neurons, under the training to learn* ϕ.

Theorem 5.3.4 states that once we fix positive integers r and m, then for any dimension $d > 2r$, there exists $\phi \in C^r(Q)$ such that $\Omega((\frac{1}{\epsilon})^{\frac{d-2r}{(4+2\delta)r}})$ of units of time may be insufficient to achieve the population risk less than $\epsilon > 0$ through gradient flow training via shallow neural network with m neurons. This demonstrates the curse of dimensionality in finite-width shallow neural network training. Although ϕ depends on the width m, Theorem 5.3.4 holds uniformly in the number of training data. This uniformity follows from Lemma 5.4.1, which is independent of the sample size.

It is noteworthy that when $\delta = 0$, the activation function σ is globally Lipschitz continuous. In this case, Theorem 5.3.4 corresponds to the finite-width shallow neural network training result established in Theorem 5.3.3.

5.4 Key Lemmas

5.4.1 Growth of second moments under the 2-Wasserstein gradient flow

An important lemma is introduced to demonstrate the sublinear growth of second moments under the 2-Wasserstein gradient flow. Consider the function $f_\pi(x) = \int_\Theta \phi(\theta, x)\pi(d\theta)$, expressed as an integral representation of parametrized functions $\{\phi(\theta, x)\}_{\theta\in\Theta}$. Let f^* be the target function to be

learned, and define the risk functional as

$$\mathcal{R}(\pi) = \frac{1}{2}\int_{\mathbb{R}^d} (f_\pi(x) - f^*(x))^2 \mathbb{P}(dx), \tag{5.3}$$

for some data distribution $\mathbb{P}$ on $\mathbb{R}^d$. For instance, when $\mathbb{P}$ is the uniform distribution on Q, the risk functional (5.3) corresponds to the population risk. Alternatively, if $\mathbb{P}$ is the empirical measure associated with a finite set of training samples $\{x_i\}_{i=1}^n \subset Q$, i.e., $\mathbb{P} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{\delta}_{x_i}$, then (5.3) represents the empirical risk. In either case, if the parameter distribution π^t evolves according to the 2-Wasserstein gradient flow of the risk functional $\mathcal{R}$, then the second moment $N(\pi^t)$ exhibits at most sublinear growth over time. This result is formally stated in the following lemma.

Lemma 5.4.1 ([120, Lemma 3.3]). *If π^t evolves according to the 2-Wasserstein gradient flow of $\mathcal{R}$ and satisfies $N(\pi^0) < \infty$, then*

$$N(\pi^t) \leq 2[N(\pi^0) + \mathcal{R}(\pi^0)t]. \tag{5.4}$$

5.4.2 Barron norms and second moments

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be an L-Lipschitz continuous activation function, and let $X \subset \mathbb{R}^d$ be a compact domain. Then, for any Barron function in $B_\sigma(X)$, bounds on its Barron norm can be established.

Lemma 5.4.2. *Any function $f \in B_\sigma(X)$ is Lipschitz continuous, with its Lipschitz constant bounded above by $L\|f\|_{B_\sigma(X)}$.*

Proof. Choose a Borel probability measure $\pi \in D_f$. Then for any $x, y \in X$, we have

$$\begin{aligned}
|f(x) - f(y)| &= \left| \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} a(\sigma(w^T x + b) - \sigma(w^T y + b))\pi(da \otimes dw \otimes db) \right| \\
&\leq \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a||\sigma(w^T x + b) - \sigma(w^T y + b)|\pi(da \otimes dw \otimes db) \\
&\leq \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a| \times L|(w^T x + b) - (w^T y + b)|\pi(da \otimes dw \otimes db) \\
&\leq L \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a| \times \|w\| \times \|x - y\|\pi(da \otimes dw \otimes db) \\
&\leq L\|x - y\| \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a| \times \|w\|_1\pi(da \otimes dw \otimes db) \\
&\leq L\|x - y\| \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a|(\|w\|_1 + |b|)\pi(da \otimes dw \otimes db) \\
&= L \times \mathbb{E}_\pi[|a|(\|w\|_1 + |b|)] \times \|x - y\|.
\end{aligned}$$

Now take the infimum over all the set D_f, and we can conclude that f is Lipschitz continuous on X with its Lipschitz constant bounded above by $L\|f\|_{B_\sigma(X)}$. □

This result yields a lower bound on $\|f\|_{B_\sigma(X)}$. The following lemma provides an upper bound using the second moments of D_f. This result plays a crucial role in the proofs of Theorems 5.3.3 and 5.3.4.

Lemma 5.4.3. *Let π be a Borel probability measure on $\mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$ such that $N(\pi) < \infty$. Then, the integral representation (2.15) defines a Barron function with its Barron norm bounded above by* $\|f\|_{B_\sigma(X)} \leq \left(\frac{\sqrt{d}}{2} + 1\right) N(\pi) + \frac{1}{2}$.

Proof. From Cauchy-Schwarz inequality, $\|w\|_1 \leq \sqrt{d}\|w\|_2$ holds. Since π is a probability measure,

$\mathbb{E}_\pi[1] = 1$ holds. Therefore we get

$$\begin{aligned}\mathbb{E}_\pi[|a|(\|w\|_1 + |b| + 1)] &\leq \sqrt{d} \times \mathbb{E}_\pi[|a|\|w\|_2] + \mathbb{E}_\pi[|a||b|] + \mathbb{E}_\pi[|a|] \\ &\leq \sqrt{d} \times \mathbb{E}_\pi[\frac{1}{2}a^2 + \frac{1}{2}\|w\|_2^2] + \mathbb{E}_\pi[\frac{1}{2}a^2 + \frac{1}{2}b^2] + \mathbb{E}_\pi[\frac{1}{2}a^2 + \frac{1}{2}] \\ &\leq (\frac{\sqrt{d}}{2} + 1) \times \mathbb{E}_\pi[a^2 + \|w\|_2^2 + b^2] + \frac{1}{2} = (\frac{\sqrt{d}}{2} + 1)N(\pi) + \frac{1}{2}.\end{aligned}$$

Write $K = \max_{x \in X} \|x\|$. Note that $|\sigma(w^T x + b)| \leq |\sigma(w^T x + b) - \sigma(0)| + |\sigma(0)| \leq L|w^T x + b| + |\sigma(0)|$ holds due to Lipchitz continuity of σ. Then we get

$$\begin{aligned}&\int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a\sigma(w^T x + b)|\pi(da \otimes dw \otimes db) \\ &\leq \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a|(L|w^T x + b| + |\sigma(0)|)\pi(da \otimes dw \otimes db) \\ &\leq \max\{L, |\sigma(0)|\} \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a|(|w^T x| + |b| + 1)\pi(da \otimes dw \otimes db) \\ &\leq \max\{L, |\sigma(0)|\} \times \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a|(\|w\|\|x\| + |b| + 1)\pi(da \otimes dw \otimes db) \\ &\leq \max\{L, |\sigma(0)|\} \times \max\{K, 1\} \times \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a|(\|w\| + |b| + 1)\pi(da \otimes dw \otimes db) \\ &\leq \max\{L, |\sigma(0)|\} \times \max\{K, 1\} \times \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} |a|(\|w\|_1 + |b| + 1)\pi(da \otimes dw \otimes db) \\ &= \max\{L, |\sigma(0)|\} \times \max\{K, 1\} \times \mathbb{E}_\pi[|a|(\|w\|_1 + |b| + 1)] \\ &\leq \max\{L, |\sigma(0)|\} \times \max\{K, 1\} \times \{(\frac{\sqrt{d}}{2} + 1)N(\pi) + \frac{1}{2}\} < \infty.\end{aligned}$$

Hence, integral representation $f(x) = \int_{\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} a\sigma(w^T x + b)\pi(da \otimes dw \otimes db)$ is well defined for all $x \in X$. Now the definition of Barron norm gives us

$$\|f\|_{B_\sigma(X)} \leq \mathbb{E}_\pi[|a|(\|w\|_1 + |b| + 1] \leq (\frac{\sqrt{d}}{2} + 1)N(\pi) + \frac{1}{2}.$$

□

5.4.3 Slow approximation property across infinitely many time scales

The following sequence plays a crucial role in constructing an element in a Banach space that exhibits poor approximation properties under appropriate time scales.

Definition 5.4.4. *A sequence $\{n_k\} \subset \mathbb{N}$ is said to be a super-exponentially increasing sequence if it is strictly increasing with $n_1 \geq 2$ and satisfies*

$$\sum_{l>k} \frac{1}{n_l} \leq \frac{2}{n_k^{k+1}} \tag{5.5}$$

for all $k \in \mathbb{N}$.

A super-exponentially increasing sequence $\{n_k\}$ satisfies the inequality

$$\sum_{i\geq 1} \frac{1}{n_i} = \frac{1}{n_1} + \sum_{l>1} \frac{1}{n_l} \leq \frac{1}{n_1} + \frac{2}{n_1^2} \leq \frac{1}{2} + \frac{2}{2^2} = 1.$$

This implies that $\lim_{k\to\infty} n_k = \infty$. An example of a super-exponentially increasing sequence is given by $n_k = 2^{k^k}$. To verify this, observe that

$$\sum_{l>k} \frac{1}{n_l} \leq \sum_{j\geq 1} 2^{-k^k(k+j)^j} = \sum_{j\geq 1} (\frac{1}{n_k})^{-(k+j)^j} \leq \sum_{j\geq 1} (\frac{1}{n_k^{k+1}})^j = \frac{1}{n_k^{k+1}} \times \frac{1}{1 - \frac{1}{n_k^{k+1}}} \leq \frac{2}{n_k^{k+1}}$$

Thus, the required condition holds for all $k \in \mathbb{N}$.

A technical lemma is introduced to demonstrate that if a sequence of linear operators exhibits different behavior in a Banach space Y and a sequence of subsets $\{X_k\}_{k\geq 1}$, then there exists an element

in the unit ball of Y that is poorly approximated by the elements of X_k under certain time scales.

Lemma 5.4.5. *Let Y, Z, W be normed linear spaces such that Y is a Banach space with a continuous embedding $Y \hookrightarrow Z$. Suppose there exist linear operators $A_n, A \in L(Z, W)$ satisfying*

$$\|A_n - A\|_{L(Y,W)} \geq c_Y n^{-\beta}, \quad \|A_n - A\|_{L(Z,W)} \leq C_Z,$$

for some $0 < \beta < \alpha$ and positive constants c_Y, C_Z that do not depend on n. Moreover, suppose there exist a super-exponentially increasing sequence $\{n_k\}$, a sequence $\{m_k\} \subset \mathbb{N}$, and a sequence of subsets $\{X_k\}_{k\geq 1} \subset Z$ such that $m_k = n_k^{[\sqrt{k}]}$ and

$$\sup_{x \in X_k} \|(A_{m_k} - A)(x)\|_W \leq \frac{c_Y m_k^{-\beta}}{8 n_k} = \frac{c_Y}{8} m_k^{-\beta - \frac{1}{[\sqrt{k}]}} \tag{5.6}$$

for all $k \geq 1$. Then, there exists an element $y \in B^Y$ such that for every $\gamma > \frac{\beta}{\alpha - \beta}$,

$$\limsup_{k\to\infty} \left[\left(\frac{m_k^{\alpha-\beta}}{n_k} \right)^{\gamma} \inf_{x \in X_k} \|x - y\|_Z \right] = \infty.$$

That is, under the time scales $t_k = \left(\frac{m_k^{\alpha-\beta}}{n_k}\right)^{\gamma}$, the element y is poorly approximated by the elements of X_k.

Proof. We construct such y in the following way.

Since $\|A_n - A\|_{L(Y,W)} = \sup_{\|x\|_Y = 1} \|(A_n - A)(x)\|_W \geq c_Y n^{-\beta}$, there exists a sequence $(y_n)_{n\geq 1}$ such that $\|y_n\|_Y = 1$ and $\|(A_n - A)(y_n)\|_W \geq \frac{c_Y}{2} n^{-\beta}$ for all $n \geq 1$. By the Hahn-Banach theorem, there exists a sequence $(w_n^*)_{n\geq 1} \subset W^*$ such that $\|w_n^*\|_{W^*} = 1$ and $w_n^* \circ (A_n - A)(y_n) = \|(A_n - A)(y_n)\|_W \geq \frac{c_Y}{2} n^{-\beta}$ for all $n \geq 1$.

Define a sequence $(\epsilon_k)_{k\geq 1}$ where $\epsilon_1 = 1$ and each $\epsilon_k \in \{-1, 1\}$ is chosen inductively such that

$$\epsilon_k \times w^*_{m_k} \circ (A_{m_k} - A)(\sum_{i=1}^{k-1} \frac{\epsilon_i}{n_i} y_{m_i}) \geq 0,$$

for all $k \geq 2$. One can easily check that $\big(\sum_{i=1}^{k} \frac{\epsilon_i}{n_i} y_{m_i}\big)_{k\geq 1}$ form a Cauchy sequence, and since Y is a Banach space, the infinite sum $y \coloneqq \sum_{i=1}^{\infty} \frac{\epsilon_i}{n_i} y_{m_i} \in Y$ is well defined.

To shorten notation, define $L_k = w^*_{m_k} \circ (A_{m_k} - A)$. Using $Y \hookrightarrow Z$, $\|A_n - A\|_{L(Z,W)} \leq C_Z$, and $\|w^*_n\|_{W^*} = 1$, one can easily verify that $L_k \in Y^*$ for all $k \geq 1$.

When $\epsilon_k = 1$, we have

$$\begin{aligned}
L_k y &= L_k(\sum_{i=1}^{k-1} \frac{\epsilon_i}{n_i} y_{m_i}) + L_k(\frac{\epsilon_k}{n_k} y_{m_k}) + L_k(\sum_{l>k} \frac{\epsilon_l}{n_l} y_{m_l}) \\
&\geq 0 + \frac{1}{n_k} L_k(y_{m_k}) - C^Y \sum_{l>k+1} \frac{1}{n_l} \\
&= \frac{1}{n_k} \times w^*_{m_k} \circ (A_{m_k} - A)(y_{m_k}) - C^Y \sum_{l>k} \frac{1}{n_l} \\
&= \frac{1}{n_k} \|(A_{m_k} - A)(y_{m_k})\|_W - C^Y \sum_{l>k1} \frac{1}{n_l} \\
&\geq \frac{1}{n_k} \big(\frac{c_Y}{2} m_k^{-\beta} - C^Y n_k \sum_{l>k} \frac{1}{n_l}\big).
\end{aligned}$$

Similarly, when $\epsilon_k = -1$, we have

$$\begin{aligned}
L_k y &= L_k(\sum_{i=1}^{k-1} \frac{\epsilon_i}{n_i} y_{m_i}) + L_k(\frac{\epsilon_k}{n_k} y_{m_k}) + L_k(\sum_{l>k} \frac{\epsilon_l}{n_l} y_{m_l}) \\
&\leq 0 - \frac{1}{n_k} L_k(y_{m_k}) + C^Y \sum_{l>k} \frac{1}{n_l} \\
&\leq -\frac{1}{n_k} \big(\frac{c_Y}{2} m_k^{-\beta} - C^Y n_k \sum_{l>k} \frac{1}{n_l}\big).
\end{aligned}$$

Therefore, we have $\frac{1}{n_k}\big(\frac{c_Y}{2}m_k^{-\beta} - C^Y n_k \sum_{l>k}\frac{1}{n_l}\big) \le |L_k y|$ for all $k \ge 1$.

Choose $x_k \in X_k$ as

$$\|y - x_k\|_Z \le \inf_{x\in X_k}\|y-x\|_Z + \frac{c_Y m_k^{-\beta}}{8C_Z n_k}.$$

Then,

$$\begin{aligned}
\frac{1}{n_k}\big(\frac{c_Y}{2}m_k^{-\beta} - C^Y n_k \sum_{l=k+1}^{\infty}\frac{1}{n_l}\big) \le |L_k(y)| &\le |L_k(y-x_k)| + |L_k(x_k)| \\
&\le C_Z\|y-x_k\|_Z + \|(A_{m_k} - A)(x_k)\|_W \\
&\le C_Z\big(\inf_{x\in X_k}\|y-x\|_Z + \frac{c_Y m_k^{-\beta}}{8C_Z n_k}\big) + \frac{c_Y m_k^{-\beta}}{8n_k} \\
&= C_Z \inf_{x\in X_k}\|y-x\|_Z + \frac{c_Y m_k^{-\beta}}{4n_k},
\end{aligned}$$

therefore we get

$$\frac{1}{C_Z n_k}\big(\frac{c_Y}{4}m_k^{-\beta} - C^Y n_k \sum_{l=k+1}^{\infty}\frac{1}{n_l}\big) \le \inf_{x\in X_k}\|y-x\|_Z.$$

Since $\{n_k\}_{k\ge1}$ is a super-exponentially increasing sequence, $m_k = n_k^{[\sqrt{k}]}$ implies $\lim_{k\to\infty}\frac{n_k^k}{m_k^\beta} = \infty$.

Hence, there exists $K_0 > 0$ such that for any $k \ge K_0$ we have

$$C^Y n_k \sum_{l=k+1}^{\infty}\frac{1}{n_l} \le C^Y n_k \frac{2}{n_k^{k+1}} = \frac{2C^Y}{n_k^k} \le \frac{c_Y}{8n_k^{\beta[\sqrt{k}]}} = \frac{c_Y}{8}m_k^{-\beta}.$$

Then for $k \ge K_0$,

$$\frac{c_Y}{8C_Z}n_k^{-1-\beta[\sqrt{k}]} = \frac{c_Y m_k^{-\beta}}{8C_Z n_k} \le \frac{1}{C_Z n_k}\big(\frac{c_Y}{4}m_k^{-\beta} - C^Y n_k \sum_{l=k+1}^{\infty}\frac{1}{n_l}\big) \le \inf_{x\in X_k}\|y-x\|_Z.$$

Now if we multiply $\big(\frac{m_k^{\alpha-\beta}}{n_k}\big)^\gamma$ on both sides, we get

$$\big(\frac{m_k^{\alpha-\beta}}{n_k}\big)^\gamma \times \frac{c_Y}{8C_Z} n_k^{-1-\beta[\sqrt{k}]} = \frac{c_Y}{8C_Z} n_k^{[\sqrt{k}]\times((\alpha-\beta)\gamma-\beta)-\gamma-1} \leq \big(\frac{m_k^{\alpha-\beta}}{n_k}\big)^\gamma \inf_{x\in X_{t_k}} \|y-x\|_Z.$$

Note that $\lim_{k\to\infty} n_k = \infty$ and $\lim_{k\to\infty} \big\{[\sqrt{k}]\times((\alpha-\beta)\gamma-\beta)-\gamma-1\big\} = \infty$ if $\gamma > \frac{\beta}{\alpha-\beta}$. Therefore for every $\gamma > \frac{\beta}{\alpha-\beta}$, we can conclude

$$\limsup_{k\to\infty} \big[\big(\frac{m_k^{\alpha-\beta}}{n_k}\big)^\gamma \inf_{x\in X_k} \|x-y\|_Z\big] \geq \frac{c_Y}{8C_Z} \limsup_{k\to\infty} n_k^{[\sqrt{k}]\times((\alpha-\beta)\gamma-\beta)-\gamma-1} = \infty.$$

□

Remark 5.4.1. *There are multiple choices for the sequence $\{m_k\}_{k\geq 1}$, and the choice $m_k = n_k^{[\sqrt{k}]}$ serves as a straightforward example that simplifies the proof. The construction of y is highly dependent on the sequences $\{n_k\}_{k\geq 1}$ and $\{m_k\}_{k\geq 1}$, implying that the existence of such an element y may not be unique. Furthermore, it is easily verified that $\lim_{k\to\infty} \left(\frac{m_k^{\alpha-\beta}}{n_k}\right)^\gamma = \infty$.*

Using Lemma 5.4.5, the following result can be established, addressing the case where a sequence of linear operators exhibits opposite behavior between two normed linear spaces. This result provides an improvement over [36, Lemma 2.3], which required $0 < \beta < \alpha/2$. The following lemma extends this condition to allow $0 < \beta < \alpha$.

Lemma 5.4.6. *Let X, Y, Z, W be normed linear spaces such that $X \subset Z$, and let Y be a Banach space with a continuous embedding $Y \hookrightarrow Z$. Suppose there exist linear operators $A_n, A \in L(Z, W)$ satisfying*

$$\|A_n - A\|_{L(X,W)} \leq C_X n^{-\alpha}, \quad \|A_n - A\|_{L(Y,W)} \geq c_Y n^{-\beta}, \quad \|A_n - A\|_{L(Z,W)} \leq C_Z,$$

for some $0 < \beta < \alpha$ and positive constants C_X, c_Y, C_Z that do not depend on n. Then, there exists an element $y \in B^Y$ such that for every $\gamma > \frac{\beta}{\alpha-\beta}$,

$$\limsup_{\kappa\to\infty}\left(\kappa^\gamma \inf_{\|x\|_X\le\kappa}\|x-y\|_Z\right)=\infty.$$

Proof. Choose any super-exponentially increasing sequence $\{n_k\}_{k\ge1}\subset\mathbb{N}$ and $m_k = n_k^{[\sqrt{k}]}$. Now let $X_k = t_k B^X$ for $k \ge 1$, where $t_k = \frac{c_Y m_k^{\alpha-\beta}}{8C_X n_k}$. Then, we have

$$\sup_{x\in X_k}\|(A_{m_k}-A)(x)\|_W = t_k\|A_{m_k}-A\|_{L(X,W)} \le t_k C_X m_k^{-\alpha} = \frac{c_Y m_k^{-\beta}}{8n_k}.$$

Now from Lemma 5.4.5, there exists $y \in B^Y$ such that

$$\limsup_{k\to\infty}\left[\left(\frac{m_k^{\alpha-\beta}}{n_k}\right)^\gamma \inf_{\|x\|_X\le t_k}\|x-y\|_Z\right]=\infty$$

holds for every $\gamma > \frac{\beta}{\alpha-\beta}$. Since $\frac{m_k^{\alpha-\beta}}{n_k} = \frac{8C_X}{c_Y}t_k$, this implies

$$\limsup_{k\to\infty}\left(t_k^\gamma \inf_{\|x\|_X\le t_k}\|x-y\|_Z\right)=\infty$$

for every $\gamma > \frac{\beta}{\alpha-\beta}$. Note that since $\alpha > \beta$ and $\lim_{k\to\infty} n_k = \infty$, we have

$$\lim_{k\to\infty} t_k = \lim_{k\to\infty}\frac{c_Y m_k^{\alpha-\beta}}{8C_X n_k} = \lim_{k\to\infty}\frac{c_Y}{8C_X}n_k^{(\alpha-\beta)[\sqrt{k}]-1}=\infty.$$

Therefore, we conclude

$$\limsup_{\kappa\to\infty}\left(\kappa^{\gamma}\inf_{\|x\|_X\leq\kappa}\|x-y\|_Z\right)\geq\limsup_{k\to\infty}\left(t_k^{\gamma}\inf_{\|x\|_X\leq t_k}\|x-y\|_Z\right)=\infty$$

for every $\gamma>\frac{\beta}{\alpha-\beta}$. □

5.4.4 Low complexity estimates

To apply Lemmas 5.4.5 and 5.4.6, it is necessary to construct appropriate linear operators $\{A_n\}_{n\geq1}$. Since the focus is on approximation rates in the L^2 topology, the space Z in both Lemma 5.4.5 and Lemma 5.4.6 must be chosen as $L^2(Q)$. In the space $L^2(Q)$, there are functions with undefined point evaluation at certain points. Therefore, following the approach in [36], we aim to define the linear operators $\{A_n\}_{n\geq1}$ and A as

$$A_n(f)=\frac{1}{n}\sum_{i=1}^{n}\fint_{B'_{\epsilon_n}(X_n^i)}f(x)dx,\quad A(f)=\int_Q f(x)dx, \tag{5.7}$$

where $\{X_n^i\}_{i=1}^n\subset Q$ represents an appropriate set of points and $\epsilon_n>0$ is a suitable radius. The integration over the projected ball $B'_{\epsilon_n}(X_n^i)$ is employed to mitigate boundary effects on the domain Q, as also noted in [36]. To establish the validity of this approach, we introduce the following lemma, which demonstrates the existence of suitable points $\{X_n^i\}_{i=1}^n\subset Q$.

Lemma 5.4.7. *Let σ be an L-Lipschitz continuous activation function. Then for any $n\in\mathbb{N}$ and any constant $0<\gamma_d\ll1$, which is independent of n, there exist n points $\{X_n^1,\cdots,X_n^n\}\subset Q$ satisfying*

$$\sup_{\phi\in B^X}\left\{\frac{1}{n}\sum_{i=1}^{n}\fint_{B'_{\epsilon_n}(X_n^i)}\phi\,dx-\int_Q\phi\,dx\right\}\leq6L\sqrt{\frac{2\log(2d)}{n}},$$

$$\sup_{\phi \in B^Z} \left\{ \frac{1}{n} \sum_{i=1}^{n} \fint_{B'_{\epsilon_n}(X_n^i)} \phi \, dx - \int_Q \phi \, dx \right\} \leq 3C,$$

for $X = B_\sigma$, $Z = L^2(Q)$, $\epsilon_n = \gamma_d n^{-1/d}$, *and* $C = C_{d,\gamma_d}$.

Proof. This result follows directly from [36, Lemma 3.3] and [36, Lemma A.10]. Any γ_d which satisfies

$$\gamma_d \times \fint_{B_1} |x| dx = \frac{cd}{d+1} \frac{1}{[(d+1)w_d]^{\frac{1}{d}}}$$

for some absolute constant $c \in (0, 1)$ is an appropriate choice, which is described in the proof of [36, Lemma 3.3]. Here, w_d denotes the volume of the unit ball in $\mathbb{R}^d$. □

5.4.5 Curse of dimensionality in numerical integration

The final step in applying Lemmas 5.4.5 and 5.4.6 is to determine an appropriate choice of γ_d such that the linear integral operators (5.7), constructed using the points from Lemma 5.4.7, exhibit different behavior in $C^r(Q)$. Intuitively, for properly scaled values of ϵ_n, the integral operators (5.7) should provide a good numerical approximation. Thus, it is natural to approach this problem using techniques from multivariate numerical integration, particularly in the context of the curse of dimensionality. For a detailed discussion on the curse of dimensionality in multivariate numerical integration, see [51, 52].

The following lemma demonstrates that the worst-case error in approximating integration using the operators in (5.7) suffers from the curse of dimensionality in $C^r(Q)$. In the proof, a function in $C^r(Q)$ is constructed to vanish in every $B'_{\epsilon_n}(X_n^i)$. This approach is inspired by [51,52], where a *fooling function* is obtained by applying a sequence of convolutions between a Lipschitz continuous function

and scaled indicator functions of a ball. After constructing this function, an appropriate choice of ϵ_n is determined to control the C^r norm and ensure proper integration over Q.

Lemma 5.4.8. *Let $C^r(Q)$ denote the space of all r-times continuously differentiable functions on Q, equipped with the norm*

$$\|f\|_{C^r} := \max_{|\beta|_1 \le r} \|D^\beta f\|_\infty,$$

where D^β represents the partial derivative of order $\beta \in \mathbb{N}_0^d$. Then, there exists a positive constant $\tau = \tau_{r,d}$ such that for any $\epsilon_n = \theta n^{-1/d}$ with $\theta = \theta_{d,r} \in (0, \tau]$ and any n points $\{x_1, \cdots, x_n\} \subset Q$, one can construct a function $\psi \in C^r(Q)$ satisfying

$$\|\psi\|_{C^r} \le 1, \quad \int_Q \psi(x)dx \ge K_{\theta,d,r} n^{-r/d}, \quad \psi|_{B'_{\epsilon_n}(x_i)} = 0,$$

for some positive constant $K_{\theta,d,r}$. Consequently, this implies that

$$\sup_{\|g\|_{C^r} \le 1} \left\{ \frac{1}{n} \sum_{i=1}^n \fint_{B'_{\epsilon_n}(x_i)} g\,dx - \int_Q g\,dx \right\} \ge K_{\theta,d,r} n^{-r/d}.$$

Before introducing the proof of Lemma 5.4.8, we begin with a lemma stating that a convolution of a Lipschitz function and an indicator function of a ball is a continuously differentiable function.

Lemma 5.4.9. *Let $f : \mathbb{R} \to \mathbb{R}$ be a K-Lipschitz continuous function. Let $\mathbb{1}_{B_r}$ be the indicator function of the ball B_r with center 0 and radius $r > 0$. Then, $f * \mathbb{1}_{B_r}$ is C^1 function with its derivatives bounded by $K \times |B_r|$, where $|B_r|$ is the volume of B_r.*

Proof. Fix an index $i \in \{1, \cdots, d\}$, and let e_i be the unit vector in $\mathbb{R}^d$ where all the entries are 0 except the ith coordinate, which is 1. By Rademacher's Theorem, f is differentiable almost everywhere.

Write $\partial_i f$ as the derivative of f with respect to x_i. From the definition of the Lipschitz constant and the derivative, it is obvious that $\|\partial_i f\|_\infty \leq K$. Therefore for any fixed $x \in \mathbb{R}^d$, we have

$$\lim_{h\to 0} \frac{f(x+he_i-y)-f(x-y)}{h} = \partial_i f(x-y),$$
$$\Big|\frac{f(x+he_i-y)-f(x-y)}{h} - \partial_i f(x-y)\Big| \leq 2K$$

almost everywhere in $y \in \mathbb{R}^d$. Hence, by Lebesgue's Dominated Convergence Theorem, for any $x \in \mathbb{R}^d$ we get

$$\begin{aligned}
&\lim_{h\to 0} \frac{f*\mathbb{1}_{B_r}(x+he_i)-f*\mathbb{1}_{B_r}(x)}{h} - \partial_i f * \mathbb{1}_{B_r}(x) \\
=&\lim_{h\to 0} \int_{B_r} \frac{f(x+he_i-y)-f(x-y)}{h} - \partial_i f(x-y)dy \\
=&\int_{B_r} \Big(\lim_{h\to 0} \frac{f(x+he_i-y)-f(x-y)}{h} - \partial_i f(x-y)\Big)dy = 0.
\end{aligned}$$

Hence, $f * \mathbb{1}_{B_r}$ is differentiable and its partial derivatives satisfy

$$\partial_i (f * \mathbb{1}_{B_r}) = \partial_i f * \mathbb{1}_{B_r}.$$

Since $\|\partial_i f\|_\infty \leq K$, it is easy to check $|\partial_i f * \mathbb{1}_{B_r}(x)| \leq K \times |B_r|$. Now, it remains to check that $\partial_i f * \mathbb{1}_{B_r}$ is continuous. From the definition of convolution,

$$\partial_i f * \mathbb{1}_{B_r}(x+z) - \partial_i f * \mathbb{1}_{B_r}(x) = \int_{\mathbb{R}^d} (\mathbb{1}_{B_r}(x+z-y) - \mathbb{1}_{B_r}(x-y))\partial_i f(y)dy.$$

Note that if $\|z\| \ll 1$, then

$$|\mathbb{1}_{B_r}(x+z-y) - \mathbb{1}_{B_r}(x-y)| \leq \mathbb{1}_{B_r(x+z)}(y) + \mathbb{1}_{B_r(x)}(y) \leq 2 \times \mathbb{1}_{B_{r+1}(x)}(y)$$

holds. Since $\|\partial_i f\|_\infty \leq K$, $|(\mathbb{1}_{B_r}(x+z-y) - \mathbb{1}_{B_r}(x-y))\partial_i f(y)|$ is bounded by $2K \times \mathbb{1}_{B_{r+1}(x)}(y)$, which is an integrable function in $\mathbb{R}^d$. Also note that

$$\lim_{\|z\|\to 0} \mathbb{1}_{B_r}(x+z-y) - \mathbb{1}_{B_r}(x-y) = 0.$$

Hence, by Lebesgue's Dominated Convergence Theorem,

$$\lim_{\|z\|\to 0} \partial_i f * \mathbb{1}_{B_r}(x+z) - \partial_i f * \mathbb{1}_{B_r}(x) = 0.$$

Therefore, the derivatives of $f * \mathbb{1}_{B_r}$ are continuous. □

Now we present the proof of Lemma 5.4.8.

Proof. Let $\{x_1, \cdots, x_n\}$ be the given n points in the unit cube Q. For each point x_i, there is a set Y_i of 3^d points in $\mathbb{R}^d$ such that for any $y \in Y_i, |(x_i)_j - y_j| \in \{0, 1\}$ for all $j \in \{1, \cdots, d\}$ where $(x_i)_j$ and y_j denote the j-th coordinate of x_i and y respectively. Define $S = Y_1 \cup \cdots \cup Y_n$ and write $S = \{s_1, \cdots, s_m\}$. It is clear that $m \leq 3^d n$. Define

$$P_{\rho'} = \bigcup_{i=1}^{m} B_{\rho'}(s_i),$$

for some $\rho' > 0$, to be specified later. For $A \subset \mathbb{R}^d$, we define the distance $d(x, A)$ between a point x

and a set A as

$$d(x, A) := \inf_{y \in A} \|x - y\|.$$

Let $h_{\rho'}(x) = \inf\left\{1, \frac{d(x,P_{\rho'})}{\rho'}\right\}$. From the construction, $\|h_{\rho'}\|_\infty = 1$. It is straightforward to check that $h_{\rho'}$ is Lipschitz continuous function with its Lipschitz constant less or equal to $1/\rho'$. Now, consider the normalized indicator function

$$g_\rho(x) = \frac{\mathbb{1}_{B_{\rho/r}}(x)}{|B_{\rho/r}|},$$

and define

$$f := h_{\rho'} * \underbrace{g_\rho * \cdots * g_\rho}_{r-\text{fold}} = h_{\rho'} *_r g_\rho.$$

where $|B_{\rho/r}|$ is the volume of ball $B_{\rho/r}$ with center origin and radius ρ/r for some $\rho > 0$, which is also specified later. Since $h_{\rho'}$ is Lipschitz continuous function, one can check that $f \in C^r$ using Lemma 5.4.9 and [51, Theorem 1-(5)]. Note that the support of the r-fold convolution of the function g_ρ is the $r-$fold Minkowski sum of the balls $B_{\rho/r}$, hence it is B_ρ. For simplicity, let us write g as the r-fold convolution of the function g_ρ.

Assume $\rho' > 2\rho$ for a moment and choose $x \in B_\rho(s_i)$. Recall $f(x) = \int_{\mathbb{R}^d} h_{\rho'}(x - t)g(t)dt$. If $\|t\| > \rho$, then since the support of g is B_ρ, we have $g(t) = 0$, hence $h_{\rho'}(x - t)g(t) = 0$. If $\|t\| \leq \rho$, we have

$$\|(x - t) - s_i\| \leq \|x - s_i\| + \|t\| \leq \rho + \rho < \rho'.$$

This implies $x - t \in B_{\rho'}(s_i)$, and therefore $x - t \in P_{\rho'}$, which makes $d(x - t, P_{\rho'}) = 0$. From the definition of $h_{\rho'}$, we get $h_{\rho'}(x - t) = 0$. Hence, we always obtain $h_{\rho'}(x - t)g(t) = 0$, which implies $f|_{B_\rho(s_i)} = 0$. This hold for any s_i, and therefore f vanishes in $\bigcup_{i=1}^m B_\rho(s_i)$. We will later choose ρ' and ρ based on this observation.

Now, observe the integration of f. It is obvious that $f \geq 0$. From the definition of $h_{\rho'}$, we have $h_{\rho'}(x) = 1$ if $d(x, P_{\rho'}) \geq \rho'$. Also note that $\|g_\rho\|_1 = \int_{\mathbb{R}^d} g_\rho(x)dx = 1$, and thus we get $\int_{\mathbb{R}^d} g(x)dx = 1$. Since the support of g is B_ρ, we have $\int_{B_\rho} g(x)dx = 1$. Note that for any $x \notin \bigcup_{i=1}^m B_{2\rho'+\rho}(s_i)$ and $z \in B_{\rho'}(s_i)$,

$$\begin{aligned}\|(x-t)-z\| &\geq \|x-z\| - \|t\| \geq \|x-s_i\| - \|z-s_i\| - \rho \\ &\geq (2\rho'+\rho) - \rho' - \rho = \rho'\end{aligned}$$

holds for any $t \in B_\rho$. Therefore, if $x \notin \bigcup_{i=1}^m B_{2\rho'+\rho}(s_i)$, then $d(x, P_{\rho'}) \geq \rho'$ and $h_{\rho'}(x-t) = 1$ for any $t \in B_\rho$. Hence, we obtain

$$f(x) = \int_{\mathbb{R}^d} h_{\rho'}(x-t)g(t)dt = \int_{B_\rho} h_{\rho'}(x-t)g(t)dt = \int_{B_\rho} 1 \times g(t)dt = 1$$

for $x \notin \bigcup_{i=1}^m B_{2\rho'+\rho}(s_i)$. Using this, the integration of f in Q can be bounded as

$$\begin{aligned}\int_Q f(x)dx &\geq 1 \times |Q \setminus \bigcup_{i=1}^m B_{2\rho'+\rho}(s_i)| \geq 1 - |\bigcup_{i=1}^m B_{2\rho'+\rho}(s_i)| \\ &\geq 1 - m \times |B_{2\rho'+\rho}| \\ &\geq 1 - n \times 3^d \times \omega_d(2\rho'+\rho)^d \end{aligned} \tag{5.8}$$

where ω_d is the volume of a unit ball in $\mathbb{R}^d$.

Next, we check the C^r norm of f. Let $e_j = (0, \cdots, 0, 1, 0, \cdots, 0)$ be the jth unit vector in $\mathbb{R}^d$, and denote the volume of a bounded Lebesgue measurable set $X \in \mathbb{R}^d$ as $\mathrm{vol}_d(X)$. Using the same argument in [51, Section 3] and [52, Section 2], for a Lipschitz continuous and bounded function $h(x)$

in $\mathbb{R}^d$, we get

$$\begin{aligned}
|D^{e_j}[h * g_\rho](x)| &= \frac{1}{\text{vol}_d(B_{\rho/r})} \left| \int_{B_{\rho/r} \cap e_j^\perp} [h(x + s + h_{\max}(s)e_j) - h(x + s - h_{\max(s)}e_j)] ds \right| \\
&\leq \frac{2\text{vol}_{d-1}(B_{\rho/r} \cap e_j^\perp)}{\text{vol}_d(B_{\rho/r})} \|h\|_\infty \leq \frac{2\text{vol}_{d-1}(B_{\rho/r})}{\text{vol}_d(B_{\rho/r})} \|h\|_\infty \\
&= \frac{2\omega_{d-1}(\rho/r)^{d-1}}{\omega_d(\rho/r)^d} \|h\|_\infty = \frac{2\omega_{d-1} r}{\omega_d \rho} \|h\|_\infty \\
&= \frac{k_d r}{\rho} \|h\|_\infty,
\end{aligned}$$

where

1. $e_j^\perp$ is the hyperplane orthogonal to e_j,
2. $h_{\max}(s) = \max\{h \geq 0 : s + he_j \in B_{\rho/r}\}$,
3. $k_d = \frac{2\omega_{d-1}}{\omega_d} > 0$ constant depending only on the dimension d.

This means that we can bound the sup-norm of a derivative as

$$\|D^{e_j}[h * g_\rho]\|_\infty \leq \frac{k_d r}{\rho} \|h\|_\infty. \tag{5.9}$$

Choose any $\beta \in \mathbb{N}_0^d$ with $|\beta|_1 = l \leq r$. Using (5.9) recursively, we get

$$\begin{aligned}
\|D^\beta f\|_\infty = \|D^\beta (h_{\rho'} * \underbrace{g_\rho * \cdots * g_\rho}_{r-\text{fold}})\|_\infty &\leq (\frac{k_d r}{\rho})^l \|h_{\rho'} * \underbrace{g_\rho * \cdots * g_\rho}_{(r-l)-\text{fold}}\|_\infty \\
&\leq (\frac{k_d r}{\rho})^l \|h_{\rho'}\|_\infty \times (\|g_\rho\|_1)^{r-l} = (\frac{k_d r}{\rho})^l
\end{aligned}$$

where the last inequality used Young's inequality $r - l$ times. Hence, we have

$$\max_{|\beta|_1=l} \|D^\beta f(x)\|_\infty \leq (\frac{k_d r}{\rho})^l.$$

If we choose $\rho \leq k_d r$, then we get a C^r norm bound

$$\|f\|_{C^r} = \max_{|\beta|_1 \leq r} \|D^\beta f(x)\|_\infty \leq (\frac{k_d r}{\rho})^r. \tag{5.10}$$

The final step is normalizing and choosing appropriate ϵ_n. Set $\rho' = 3\epsilon_n, \rho = \epsilon_n$, and

$$F(x) = \frac{f(x)}{(\frac{k_d r}{\epsilon_n})^r}.$$

From the analysis above, F is $C^r(\mathbb{R}^d)$ and $\|F\|_{C^r} \leq 1$ if we set $\epsilon_n \leq k_d r$. If we integrate F in the unit cube Q, we get

$$\begin{aligned}
\int_Q F(x)dx &= (\frac{\epsilon_n}{k_d r})^r \int_Q f(x)dx \\
&\geq (\frac{\epsilon_n}{k_d r})^r \left(1 - n \times 3^d \times \omega_d (2\rho' + \rho)^d\right) \\
&= (\frac{\epsilon_n}{k_d r})^r \left(1 - n \times 3^d \times \omega_d (7\epsilon_n)^d\right) \\
&= (\frac{\epsilon_n}{k_d r})^r \left(1 - n \times \omega_d (21\epsilon_n)^d\right)
\end{aligned}$$

where the inequality used is (5.8). Define

$$\tau := \min\left\{\frac{1}{21}\left(\frac{1}{2w_d}\right)^{1/d}, k_d\right\}. \tag{5.11}$$

Now, choose $\epsilon_n = \theta n^{-1/d}$, where $\theta \in (0, \tau]$ is a positive constant depending only on d. Then, it is easy to check

1. $\rho = \epsilon_n \leq \tau n^{-1/d} \leq \tau \leq k_d \leq k_d r$,

2. $1 - n \times \omega_d (21\epsilon_n)^d = 1 - \omega_d \times (21\tau)^d \geq 1/2$.

Then we obtain

$$\int_Q F(x)dx \geq \frac{1}{2} \times (\frac{\epsilon_n}{k_d r})^r = \frac{\theta^r}{2k_d^r r^r} n^{-r/d}, \quad \|F\|_{C^r} = (\frac{\epsilon_n}{k_d r})^r \|f\|_{C^r} \leq 1.$$

Let ψ be the restriction of F onto the unit cube Q. Denote $K_{\theta,d,r} = \frac{\theta^r}{2k_d^r r^r}$. Then $\|\psi\|_{C^r} \leq 1$ and $\int_Q \psi(x)dx = \int_Q F(x)dx \geq K_{\theta,d,r} n^{-r/d}$ are obvious. From the previous analysis, we showed that f vanishes in $\bigcup_{i=1}^m B_{\epsilon_n}(s_i)$. Therefore, F also vanishes in $\bigcup_{i=1}^m B_{\epsilon_n}(s_i)$. By recalling the definition of s_i's, it easy to check $F(x) = 0$ when $x \in \bigcup_{i=1}^n B'_{\epsilon_n}(x_i)$. Hence, the restriction ψ also satisfies $\psi|_{B'_{\epsilon_n}(x_i)} = 0$ and therefore $⨍_{B'_{\epsilon_n}(x_i)} \psi(x)dx = 0$ holds. Finally, we get

$$\begin{aligned}
&\sup_{\|g\|_{C^r} \leq 1} \{\frac{1}{n} \sum_{i=1}^n ⨍_{B'_{\epsilon_n}(x_i)} g dx - \int_Q g dx\} \\
&\geq \frac{1}{n} \sum_{i=1}^n ⨍_{B'_{\epsilon_n}(x_i)} (-\psi) dx - \int_Q (-\psi) dx = \int_Q \psi dx \\
&\geq K_{\theta,d,r} n^{-r/d}.
\end{aligned}$$

□

5.5 Proofs of the Main Theorems

In this section, we present the proofs of Theorems 5.3.1, 5.3.3, and 5.3.4. While the proofs of Theorems 5.3.1 and 5.3.3 rely on Lemma 5.4.6, the proof of Theorem 5.3.4 instead utilizes Lemma 5.4.5. This distinction arises for two reasons: 1) When σ is not a Lipschitz continuous function, the Contraction Lemma [99, Lemma 26.9] cannot be applied to bound the Rademacher complexity of the unit ball in B_σ. 2) If the activation is not Lipschitz continuous, bounding the Barron norm using the second moments (as in Lemma 5.4.3) may be infeasible. We provide a more detailed discussion in Section 5.5.3. However, if the analysis is restricted to finite-width training, which represents the realistic scenario, arguments similar to those presented in this section can be employed to establish Theorem 5.3.4.

5.5.1 Proof of Theorem 5.3.1

Proof. Define $X = B_\sigma(Q), Y = C^r(Q)$ equipped with the norm defined in Lemma 5.4.8, $Z = L^2(Q)$, and $W = \mathbb{R}$. Since Q is a compact set, It is clear that $X, Y \hookrightarrow Z$. Now we find linear operators A_n, A that satisfies conditions in Lemma 5.4.6. First, choose $\gamma = \gamma_{d,r}$ as $0 < \gamma \ll 1$ and $\gamma \leq \tau$, where τ is the constant in Lemma 5.4.8. Set $\epsilon_n = \gamma n^{-1/d}$. For any $n \in \mathbb{N}$, there exist n points $\{X_n^1, \cdots, X_n^n\} \subset Q$ that satisfies Lemma 5.4.7. Then with Lemma 5.4.8, we can conclude that these points satisfy inequalities:

$$\sup_{\phi \in B^X} \{\frac{1}{n}\sum_{i=1}^n \fint_{B'_{\epsilon_n}(X_n^i)} \phi dx - \int_Q \phi dx\} \leq 6L\sqrt{\frac{2\log(2d)}{n}},$$
$$\sup_{\phi \in B^Y} \{\frac{1}{n}\sum_{i=1}^n \fint_{B'_{\epsilon_n}(X_n^i)} \phi dx - \int_Q \phi dx\} \geq K_{\gamma,d,r} n^{-r/d},$$
$$\sup_{\phi \in B^Z} \{\frac{1}{n}\sum_{i=1}^n \fint_{B'_{\epsilon_n}(X_n^i)} \phi dx - \int_Q \phi dx\} \leq 3C_{d,\gamma}.$$

Define linear operators $A_n, A : Z \to \mathbb{R}$ as

$$A_n(\phi) = \frac{1}{n}\sum_{i=1}^{n} \fint_{B'_{\epsilon_n}(X_n^i)} \phi dx, \quad A(\phi) = \int_Q \phi dx \tag{5.12}$$

Then A_n, A satisfy the conditions in Lemma 5.4.6. Hence by Lemma 5.4.6, there exist a function $f \in C^r(Q)$ with $\|f\|_{C^r} \leq 1$ such that

$$\limsup_{\kappa\to\infty} \left(\kappa^\gamma \inf_{\|\phi\|_{B_\sigma}\leq\kappa} \|\phi - f\|_{L^2(Q)}\right) = \infty$$

holds for every $\gamma > \frac{r/d}{1/2-r/d} = \frac{2r}{d-2r}$. Note that since Q is compact with Lebesgue measure 1, $\|g\|_{L^2(Q)} \leq \|g\|_{L^p(Q)}$ holds for all continuous function g on Q and for all $p \in [2,\infty]$. Therefore, we can conclude

$$\limsup_{\kappa\to\infty} \left(\kappa^\gamma \inf_{\|\phi\|_{B_\sigma}\leq\kappa} \|\phi - f\|_{L^p(Q)}\right) = \infty$$

holds for every $\gamma > \frac{r/d}{1/2-r/d} = \frac{2r}{d-2r}$ and $p \in [2,\infty]$. □

5.5.2 Proof of Theorem 5.3.3

Proof. Choose any $\phi \in C^r(Q)$ that satisfies Theorem 5.3.1. Let π^0 be the Borel probability measure at the training initialization with $N(\pi^0) < \infty$, and let π^t be the evolution of π^0 by the Wasserstein gradient flow at time $t > 0$. By Lemma 5.4.1, $N(\pi^t) < \infty$. and therefore $f_{\pi^t} \in B_\sigma$ holds from Lemma 5.4.3. Moreover, Lemma 5.4.3 and Lemma 5.4.1 give us estimation of the Barron norm of f_{π^t} as

$$\|f_{\pi^t}\|_{B_\sigma} \leq (\frac{\sqrt{d}}{2}+1)N(\pi^t) + \frac{1}{2} \leq (\sqrt{d}+2)(N(\pi^0) + R(\pi^0)t) + \frac{1}{2}.$$

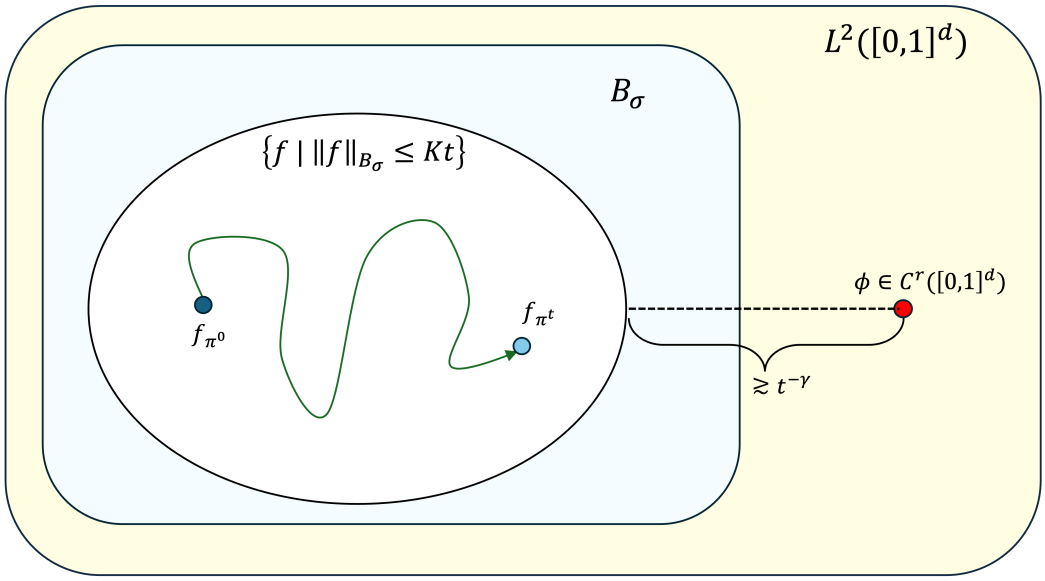

Figure 5.1: Geometric description of the proof of Theorem 5.3.3. The green curve with arrow illustrates the sublinear growth of the Barron norm, which follows from Lemma 5.4.1 and Lemma 5.4.3. The shallow neural network at the initialization is denoted as f_{π^t}, represented as a circle filled with dark blue. The shallow neural network training time t is denoted as f_{π^t}, represented as a circle filled with sky-blue. The black dotted line represents Theorem 5.3.1, the existence of $C^r(Q)$ function with slow approximation property.

Hence, there exists a positive constant $K = K_{\pi^0,\phi,d}$ such that $\|f_{\pi^t}\|_{B_\sigma} \leq Kt$ holds for $t \geq 1$. Note that the population risk at time t, $\mathcal{R}_p(\pi^t)$, is equal to $\frac{1}{2}\|\phi - f_{\pi^t}\|^2_{L^2(Q)}$. Now by Theorem 5.3.1,

$$\begin{aligned}\limsup_{t\to\infty}\left[t^\gamma \mathcal{R}_p(\pi^t)\right] &= \frac{1}{2}\limsup_{t\to\infty}\left(t^\gamma \|\phi - f_{\pi^t}\|^2_{L^2(Q)}\right) \geq \frac{1}{2}\limsup_{t\to\infty}\left(t^\gamma \inf_{\|f\|_{B_\sigma}\leq Kt}\|\phi - f\|^2_{L^2(Q)}\right) \\ &= \frac{1}{2}\left(\frac{1}{K}\right)^\gamma \times \limsup_{t\to\infty}\left(t^\gamma \inf_{\|f\|_{B_\sigma}\leq t}\|\phi - f\|^2_{L^2(Q)}\right) = \infty\end{aligned}$$

holds for any $\gamma > 2 \times \frac{r/d}{1/2 - r/d} = \frac{4r}{d-2r}$. □

5.5.3 Proof of Theorem 5.3.4

Unlike Lipschitz continuous activation functions, the Rademacher complexity of the unit ball in the Barron space cannot be controlled in the same manner as in [36, Lemma A.10], where the

Contraction Lemma [99, Lemma 26.9] plays a crucial role in the proof. Fortunately, for a locally Lipschitz continuous activation function σ whose Lipschitz constant in $[-x, x]$ is bounded by $O(x^\delta)$, a similar approach to the proof of Theorem 5.3.3 can be employed.

However, the analysis does not extend to infinite-width neural network training for two primary reasons. First, unlike Lemma 5.4.3, probability measures with finite second moments do not necessarily guarantee a well-defined integral representation. For instance, consider the case where $d = 1$, $\sigma(x) = \max\{0, x\}^2$, and the probability measure π is defined as $\pi(a = n, b = 0, w = n) = \frac{1}{Kn^4}$, for all $n \in \mathbb{N}$, where $K = \sum_{i=1}^{\infty} \frac{1}{i^4} < \infty$. It is straightforward to verify that $N(\pi) < \infty$, yet the integral representation (2.15) is undefined. While one might consider other notions of Barron spaces [50, 105–107], such settings make the control of Barron norms via $N(\pi^t)$-as in Lemma 5.4.3-intractable. Consequently, we cannot track the Barron norm of infinite-width neural networks under gradient flow, precluding the application of the same analysis.

Second, probability measures that define infinite-width neural networks do not impose uniform bounds on the parameters, making it difficult to control the Rademacher complexity. Nevertheless, when focusing on finite-width neural networks, the setting relevant to practical implementations, the curse of dimensionality can still be established. Since the set of shallow networks with m neurons does not constitute a normed vector space, the proof relies on Lemma 5.4.5 instead of Lemma 5.4.6.

In this section, we consider probability distributions of the form

$$\frac{1}{m} \sum_{i=1}^{m} \boldsymbol{\delta}_{(a_i, w_i, b_i)} \tag{5.13}$$

which correspond to a two-layer neural network $f(x) = \frac{1}{m} \sum_{i=1}^{m} a_i \sigma(w_i^T x + b_i)$ with m neurons in the mean-field setting. All distribution π_m discussed in this section adhere to the form given in (5.13).

Note that if π_m^t evolves by the 2-Wasserstein gradient flow of the risk function (5.3), then π_m^t remains in the form of (5.13) at all times. This follows from the the observations in [22, Proposition B.1] and [121, Lemma 3].

We begin with some definitions. Define two sets $F_{m,D}$ and S_D as

$$F_{m,D} := \{f_{\pi_m} : \pi_m = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\delta}_{(a_i,w_i,b_i)}, N(\pi_m) \leq D\},$$

$$S_D := \{a\sigma(w^T x + b) : a^2 + \|w\|_2^2 + b^2 \leq D\}.$$

From the definition, it is obvious that any element in $F_{m,D}$ can be expressed as a convex combination of single neurons of the form $a\sigma(w^T x+b)$ in S_{mD}. Hence, $F_{m,D}$ is a subset of the convex hull of S_{mD}, which implies for any finite set $S \in \mathbb{R}^d$,

$$\mathrm{Rad}(F_{m,D}, S) \leq \mathrm{Rad}(\mathrm{conv}(S_{mD}), S).$$

holds, where $\mathrm{Rad}(F, S)$ denotes Rademacher complexity of F with respect to S. For further details on Rademacher complexities, see [99, Chapter 26].

Let σ be a locally Lipschitz continuous function and denote $L_k = \sup_{x\neq y, x,y\in[-k,k]} \frac{|\sigma(x)-\sigma(y)|}{|x-y|}$ the Lipschitz constant of σ in the domain $[-k, k]$.

Lemma 5.5.1. *If* $\|w\|^2+b^2 \leq mD$, *then* $|\sigma(w^T x+b)-\sigma(w^T y+b)| \leq L_{\sqrt{mD(d+1)}}\|w^T(x-y)\|$ *holds for any* $x, y \in Q$.

Proof. For any $x \in Q$, $|w^T x+b| \leq \sqrt{(\|w\|_2^2+b^2)(d+1)} \leq \sqrt{mD(d+1)}$ holds. Then the inequality holds due to the definition of Lipschitz constant. □

Now we give an estimate on the empirical Rademacher complexity of S_{mD}.

Lemma 5.5.2. *Let $\{X_1, \cdots, X_n\} \subset Q$. Then we have*

$$\mathrm{Rad}(S_{mD}, \{X_1, \cdots, X_n\}) \leq \frac{L_{\sqrt{mD(d+1)}} \times mD\sqrt{d+1}}{2\sqrt{n}}. \tag{5.14}$$

Proof. Using Lemma 5.5.1 and the Contraction Lemma [99, Lemma 26.9], we get

$$\begin{aligned}
\mathrm{Rad}(S_{mD},\{X_1, \cdots, X_n\}) &= \mathbb{E}_\zeta\Big[\sup_{(a,b,c):a^2+\|w\|_2^2+b^2\leq mD} \frac{1}{n}\sum_{i=1}^n \zeta_i a\sigma(w^T X_i + b)\Big] \\
&\leq L_{\sqrt{mD(d+1)}} \times \mathbb{E}_\zeta\Big[\sup_{(a,b,c):a^2+\|w\|_2^2+b^2\leq mD} \frac{1}{n}\sum_{i=1}^n \zeta_i a(w^T X_i + b)\Big] \\
&= \frac{L_{\sqrt{mD(d+1)}}}{n} \times \mathbb{E}_\zeta\Big[\sup_{(a,b,c):a^2+\|w\|_2^2+b^2\leq mD} \sum_{i=1}^n \zeta_i \langle a(w^T, b)^T, (X_i^T, 1)^T\rangle\Big] \\
&= \frac{L_{\sqrt{mD(d+1)}}}{n} \times \mathbb{E}_\zeta\Big[\sup_{(a,b,c):a^2+\|w\|_2^2+b^2\leq mD} \langle a(w^T, b)^T, \sum_{i=1}^n \zeta_i (X_i^T, 1)^T\rangle\Big] \\
&\leq \frac{L_{\sqrt{mD(d+1)}}}{n} \times \mathbb{E}_\zeta\Big[\sup_{(a,b,c):a^2+\|w\|_2^2+b^2\leq mD} \|a(w^T, b)^T\|_2 \times \|\sum_{i=1}^n \zeta_i (X_i^T, 1)^T\|_2\Big] \\
&\leq \frac{L_{\sqrt{mD(d+1)}}}{n} \times \mathbb{E}_\zeta\Big[\frac{mD}{2} \times \|\sum_{i=1}^n \zeta_i (X_i^T, 1)^T\|_2\Big] \\
&\leq \frac{L_{\sqrt{mD(d+1)}} mD}{2n} \times \Big(\mathbb{E}_\zeta\big[|\sum_{i=1}^n \zeta_i (X_i^T, 1)^T\|_2]^2\big]\Big)^{1/2} \\
&\leq \frac{L_{\sqrt{mD(d+1)}} mD}{2n} \times \Big(n \times \max_i \|(X_i^T, 1)^T\|_2^2\Big)^{1/2} \\
&\leq \frac{L_{\sqrt{mD(d+1)}} mD}{2n} \times \sqrt{n \times (d+1)} \\
&= \frac{L_{\sqrt{mD(d+1)}} \times mD\sqrt{d+1}}{2\sqrt{n}}.
\end{aligned}$$

□

Corollary 5.5.3. *Let* $\{X_1, \cdots, X_n\} \subset Q$. *Then,*

$$\mathrm{Rad}(F_{m,D}, \{X_1, \cdots, X_n\}) \leq \frac{L_{\sqrt{mD(d+1)}} \times mD\sqrt{d+1}}{2\sqrt{n}}.$$

Proof. Note that

$$\mathrm{Rad}(F_{m,D}, \{X_1, \cdots, X_n\}) \leq \mathrm{Rad}(\mathrm{conv}(S_{mD}), \{X_1, \cdots, X_n\}) = \mathrm{Rad}(S_{mD}, \{X_1, \cdots, X_n\}).$$

Now use Lemma 5.5.2. □

Finally, we introduce two lemmas that assist our proof. Let U_Q be the uniform distribution on Q.

Lemma 5.5.4. *Let* $Z = L^2(Q)$. *Fix* $0 < \gamma \ll 1$ *and set* $\epsilon_n = \gamma n^{-1/d}$ *for* $n \in \mathbb{N}$. *Then,*

$$\mathbb{E}_{X_i \overset{\mathrm{iid}}{\sim} U_Q} \sup_{\phi \in B^Z} \Big[\frac{1}{n}\sum_{i=1}^{n} \fint_{B'_{\epsilon_n}(X_i)} \phi(x)dx - \int_Q \phi(x)dx\Big] \leq \sqrt{\frac{1 + a_d\gamma^d}{b_d\gamma^d}}$$

holds where a_d *and* b_d *are positive constant depending only on* d.

The proof of Lemma 5.5.4 can be found in the proof of [36, Lemma 3.3].

Lemma 5.5.5. *Let* F *be a a subset of* $C(Q)$. *Then for any* $0 < \epsilon < 1$, *we have*

$$\mathbb{E}_{X_i \overset{\mathrm{iid}}{\sim} U_Q} \sup_{f \in F} [\frac{1}{n}\sum_{i=1}^{n} \fint_{B'_\epsilon(X_i)} f(x)dx - \int_Q f(x)dx] \leq \mathbb{E}_{X_i \overset{\mathrm{iid}}{\sim} U_Q} \sup_{f \in F} [\frac{1}{n}\sum_{i=1}^{n} f(X_i) - \int_Q f(x)dx].$$

Proof. In this proof, we interpret $X_i + z$ as a shift on the flat torus. Note that for a fixed z, X_i and

$X_i + z$ have the same distribution. Therefore we have

$$
\begin{aligned}
&\mathbb{E}_{X_i \overset{\text{iid}}{\sim} U_Q} \sup_{f \in F} \Big[\frac{1}{n} \sum_{i=1}^{n} \fint_{B'_\epsilon(X_i)} f(x)dx - \int_Q f(x)dx \Big] \\
=&\mathbb{E}_{X_i \overset{\text{iid}}{\sim} U_Q} \sup_{f \in F} \Big[\frac{1}{n} \sum_{i=1}^{n} \fint_{B'_\epsilon(0)} f(X_i + z)dz - \int_Q f(x)dx \Big] \\
=&\mathbb{E}_{X_i \overset{\text{iid}}{\sim} U_Q} \sup_{f \in F} \Big[\fint_{B'_\epsilon(0)} \frac{1}{n} \sum_{i=1}^{n} \big(f(X_i + z) - \int_Q f(x)dx \big) dz \Big] \\
\leq&\mathbb{E}_{X_i \overset{\text{iid}}{\sim} U_Q} \fint_{B'_\epsilon(0)} \sup_{f \in F} \Big[\frac{1}{n} \sum_{i=1}^{n} \big(f(X_i + z) - \int_Q f(x)dx \big) \Big] dz \\
=&\fint_{B'_\epsilon(0)} \mathbb{E}_{X_i \overset{\text{iid}}{\sim} U_Q} \sup_{f \in F} \Big[\frac{1}{n} \sum_{i=1}^{n} \big(f(X_i + z) - \int_Q f(x)dx \big) \Big] dz \\
=&\mathbb{E}_{X_i \overset{\text{iid}}{\sim} U_Q} \sup_{f \in F} \Big[\frac{1}{n} \sum_{i=1}^{n} f(X_i) - \int_Q f(x)dx \Big].
\end{aligned}
$$

□

To prove Theorem 5.3.4, we begin with a lemma which is similar to Lemma 5.3.1 for locally Lipschitz activation functions that satisfy $L_t = O(t^\delta)$. We prove that for fixed $m \in \mathbb{N}$, there exists $\phi \in C^r(Q)$ which is poorly approximated by shallow neural networks with width m.

Lemma 5.5.6. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a locally Lipschitz function with $L_t = O(t^\delta)$ for some $\delta \geq 0$. Fix $m \in \mathbb{N}$. Assume $r < \frac{d}{2}$. Then there exists $\phi \in C^r(Q)$ such that*

$$
\limsup_{t \to \infty} \big(t^\gamma \inf_{f \in F_{m,t}} \|\phi - f\|_{L^2(Q)} \big) = \infty.
$$

holds for every $\gamma > \frac{(2+\delta)r}{d-2r}$.

Proof. Since the approximators have the same number of neurons, their union does not form a vector space. Therefore, instead of applying Lemma 5.4.6, we utilize Lemma 5.4.5.

Let $Y = C^r(Q)$ equipped with the norm defined in Lemma 5.4.8, $Z = L^2(Q)$, and $W = \mathbb{R}$. Given that $L_t = O(t^\delta)$, there exist $T = T_\sigma > 0$ and $C = C_\sigma > 0$ such that $L_t \leq Ct^\delta$ for $t \geq T$. Define $A \in L(Z, W)$ be defined as in (5.12) in the proof of Theorem 5.3.1. For any $n \in \mathbb{N}$, choose $\epsilon_n = \gamma n^{-1/d}$, as done in the proof of Theorem 5.3.1.

Note that for any super-exponentially increasing sequence $\{n_k\}_{k\geq 1}$ and $m_k = n_k^{[\sqrt{k}]}$, we have

$$\lim_{k\to\infty} \frac{m_k^{\frac{1}{2}-\frac{r}{d}}}{n_k} = \lim_{k\to\infty} n_k^{(\frac{1}{2}-\frac{r}{d})[\sqrt{k}]-1} = \infty. \tag{5.15}$$

Therefore, there exists $k_0 \in \mathbb{N}$ such that

$$T^2 \leq \Big(\frac{m_k^{\frac{1}{2}-\frac{r}{d}} K_{\gamma,d,r}}{24 n_k C m^{1+\frac{\delta}{2}} (d+1)^{\frac{1}{2}+\frac{\delta}{2}}}\Big)^{\frac{1}{1+\frac{\delta}{2}}} \tag{5.16}$$

for all $k \geq k_0$, where $K_{\gamma,d,r}$ is the constant in Lemma 5.4.8. Now, define $n'_k = n_{k+k_0}$. Then n'_k is also a super-exponentially increasing sequence because it is strictly increasing with $n'_1 = n_{k_0+1} > n_1 \geq 2$ and

$$\sum_{l>k} \frac{1}{n'_l} = \sum_{l>k+k_0} \frac{1}{n_l} \leq \frac{2}{n_{k+k_0}^{k+k_0+1}} \leq \frac{2}{n_{k+k_0}^{k+1}} = \frac{2}{(n'_k)^{k+1}}$$

holds for all $k \in \mathbb{N}$. Define $m'_k = (n'_k)^{[\sqrt{k}]}$. Then for any $k \geq 1$,

$$T^2 \leq \Big(\frac{(m'_k)^{\frac{1}{2}-\frac{r}{d}} K_{\gamma,d,r}}{24 n'_k C m^{1+\frac{\delta}{2}} (d+1)^{\frac{1}{2}+\frac{\delta}{2}}}\Big)^{\frac{1}{1+\frac{\delta}{2}}}$$

holds. From this observation, we can choose a super-exponentially increasing sequence $\{n_k\}_{k\geq 1}$ and $m_k = n_k^{[\sqrt{k}]}$ such that inequality (5.16) holds for all $k \geq 1$.

Choose a super-exponentially increasing sequence $\{n_k\}_{k\geq 1}$ and $m_k = n_k^{\lceil\sqrt{k}\rceil}$ that satisfy (5.16) for all $k \geq 1$. Define time scales $\{t_k\}_{k\geq 1}$ as $t_k = \left(\frac{m_k^{\frac{1}{2}-\frac{r}{d}} K_{\gamma,d,r}}{24 n_k C m^{1+\frac{\delta}{2}} (d+1)^{\frac{1}{2}+\frac{\delta}{2}}}\right)^{\frac{1}{1+\frac{\delta}{2}}}$. From (5.16), $t_k \geq T^2$ holds for all $k \geq 1$. Now, let us find appropriate $\{A_n\}_{n\geq 1} \in L(Z,W)$ for Lemma 5.4.5.

When $n \notin \{m_k\}_{k\geq 1}$, define $A_n \in L(Z,W)$ as in the proof of Theorem 5.3.1. Then we have

$$\|A_n - A\|_{L(Y,W)} \geq K_{\gamma,d,r} n^{-r/d}, \quad \|A_n - A\|_{L(Z,W)} \leq 3C_{d,\gamma}$$

for all $n \notin \{m_k\}_{k\geq 1}$. The constants $K_{\gamma,d,r}$ and $C_{d,\gamma}$ are identical to those used in the proof of Theorem 5.3.1.

When $n = m_k$, we construct the linear operators $\{A_{m_k}\}_{k\geq 1}$ as follows. Since $t_k \geq T^2$ for all $k \geq 1$, $\sqrt{mt_k(d+1)} \geq T$ hold, which implies $L_{\sqrt{mt_k(d+1)}} \leq C(\sqrt{mt_k(d+1)})^\delta$. From Lemma 5.5.5 and Corollary 5.5.3,

$$\begin{aligned}
&\mathbb{E}_{X_i \overset{\text{iid}}{\sim} U_Q} \sup_{f\in F_{m,t_k}} \Big[\frac{1}{m_k}\sum_{i=1}^{m_k} \fint_{B'_{\epsilon_{m_k}}(X_i)} f(x)dx - \int_Q f(x)dx\Big] \\
\leq &\mathbb{E}_{X_i \overset{\text{iid}}{\sim} U_Q} \sup_{f\in F_{m,t_k}} \Big[\frac{1}{m_k}\sum_{i=1}^{m_k} f(X_i) - \int_Q f(x)dx\Big] \\
\leq &2 \times \mathbb{E}_{X_i \overset{\text{iid}}{\sim} U_Q} \mathrm{Rad}(F_{m,t_k}, \{X_1, \cdots, X_{m_k}\}) \\
\leq &2 \times \frac{L_{\sqrt{mt_k(d+1)}} \times mt_k\sqrt{d+1}}{2\sqrt{m_k}} \\
\leq &\frac{C(\sqrt{mt_k(d+1)})^\delta mt_k\sqrt{d+1}}{\sqrt{m_k}} \\
= &Cm^{1+\frac{\delta}{2}}(d+1)^{\frac{1}{2}+\frac{\delta}{2}} \frac{t_k^{1+\frac{\delta}{2}}}{\sqrt{m_k}} = \frac{K_{\gamma,d,r} m_k^{-\frac{r}{d}}}{24 n_k}
\end{aligned} \tag{5.17}$$

holds. The second inequality holds because the expected value of the representativeness is bounded by twice the expected Rademacher complexity [99, Lemma 26.2]. Now with Lemma 5.5.4 and

(5.17), by using a simple probabilistic argument using the Markov inequality, there exist m_k points $\{X_{m_k}^1, \cdots, X_{m_k}^{m_k}\} \subset Q$ such that

$$\sup_{\phi \in B^Y} \Big[\frac{1}{m_k}\sum_{i=1}^{m_k} \fint_{B'_{\epsilon_{m_k}}(X_{m_k}^i)} \phi dx - \int_Q \phi dx\Big] \geq K_{\gamma,d,r} m_k^{-r/d},$$
$$\sup_{\phi \in B^Z} \Big[\frac{1}{m_k}\sum_{i=1}^{m_k} \fint_{B'_{\epsilon_{m_k}}(X_{m_k}^i)} \phi dx - \int_Q \phi dx\Big] \leq 3\sqrt{\frac{1+a_d\gamma^d}{b_d\gamma^d}},$$
$$\sup_{f \in F_{m,t_k}} \Big[\frac{1}{m_k}\sum_{i=1}^{m_k} \fint_{B'_{\epsilon_{m_k}}(X_{m_k}^i)} f(x) dx - \int_Q f(x) dx\Big] \leq \frac{K_{\gamma,d,r} m_k^{-\frac{r}{d}}}{8n_k}$$

hold, where the first inequality is due to Lemma 5.4.8. Now using these points $\{X_{m_k}^1, \cdots, X_{m_k}^{m_k}\} \subset Q$, we define $\{A_{m_k}\}_{k\geq 1} \subset L(Z,W)$ as

$$A_{m_k}(\phi) = \frac{1}{m_k}\sum_{i=1}^{m_k} \fint_{B'_{\epsilon_{m_k}}(X_{m_k}^i)} \phi dx.$$

With the constructed linear operators $\{A_n\}_{n\geq 1}$, by setting $X_k = F_{m,t_k}$ for $k \geq 1$, one can easily verify that the conditions in Lemma 5.4.5 are satisfied with $\alpha = \frac{1}{2}, \beta = \frac{r}{d}, c_Y = K_{\gamma,d,r}$ and $C_Z = \max\{3\sqrt{\frac{1+a_d\gamma^d}{b_d\gamma^d}}, 3C_{d,\gamma}\}$. Now from Lemma 5.4.5, there exists $\phi \in B^Y$ such that for every $\gamma > \frac{r/d}{1/2-r/d}$, we have

$$\limsup_{k\to\infty} \Big[\big(\frac{m_k^{1/2-r/d}}{n_k}\big)^\gamma \inf_{f\in F_{m,t_k}} \|f-\phi\|_{L^2(Q)}\Big] = \infty. \tag{5.18}$$

Note that $\frac{m_k^{\frac{1}{2}-\frac{r}{d}}}{n_k} = A t_k^{1+\frac{\delta}{2}}$ where $A = \frac{24Cm^{1+\frac{\delta}{2}}(d+1)^{\frac{1}{2}+\frac{\delta}{2}}}{K_{\gamma,d,r}}$ is a constant that does not depend on k. Hence, (5.18) can be written as

$$\limsup_{k\to\infty} \Big[t_k^{\gamma(1+\frac{\delta}{2})} \inf_{f\in F_{m,t_k}} \|f-\phi\|_{L^2(Q)}\Big] = \infty \tag{5.19}$$

for every $\gamma > \frac{r/d}{1/2-r/d}$. From (5.15), $\lim_{k\to\infty} t_k = \infty$ holds, and with (5.19), we conclude

$$\limsup_{t\to\infty} \left(t^\gamma \inf_{f\in F_{m,t}} \|\phi - f\|_{L^2(Q)}\right) \geq \limsup_{k\to\infty} \left(t_k^\gamma \inf_{f\in F_{m,t_k}} \|f - \phi\|_{L^2(Q)}\right) = \infty.$$

for every $\gamma > (1 + \frac{\delta}{2}) \times \frac{r/d}{1/2-r/d} = \frac{(2+\delta)r}{d-2r}$. This finishes the proof of Lemma 5.5.6. □

We now present the proof of Theorem 5.3.4.

Proof. Choose any $\phi \in C^r(Q)$ that satisfies Lemma 5.5.6. Let π_m^0 be the initial parameter distribution at time $t = 0$, and let π_m^t denote the evolution of π_m^0 under the Wasserstein gradient flow at time $t > 0$. By Lemma 5.4.1, there exists a positive constant $K = K_{\pi_m^0,\phi}$ such that $N(\pi_m^t) \leq Kt$ holds for all $t \geq 1$. Note that the population risk at time t, $R(\pi_m^t)$, is equal to $\frac{1}{2}\|\phi - f_{\pi_m^t}\|_{L^2(Q)}^2$. Therefore by Lemma 5.5.6,

$$\begin{aligned}
\limsup_{t\to\infty} \left[t^\gamma R(\pi_m^t)\right] &= \frac{1}{2} \limsup_{t\to\infty} \left(t^\gamma \|\phi - f_{\pi_m^t}\|_{L^2(Q)}^2\right) \\
&\geq \frac{1}{2} \limsup_{t\to\infty} \left(t^\gamma \inf_{N(\pi_m)\leq Kt} \|\phi - f_{\pi_m}\|_{L^2(Q)}^2\right) \\
&= \frac{1}{2} \limsup_{t\to\infty} \left(t^\gamma \inf_{f\in F_{m,Kt}} \|\phi - f\|_{L^2(Q)}^2\right) \\
&= \frac{1}{2}\left(\frac{1}{K}\right)^\gamma \times \limsup_{t\to\infty} \left(t^\gamma \inf_{f\in F_{m,t}} \|\phi - f\|_{L^2(Q)}^2\right) = \infty
\end{aligned}$$

holds for every $\gamma > 2 \times \frac{(2+\delta)r}{d-2r} = \frac{(4+2\delta)r}{d-2r}$. □

5.6 Conclusion

In this chapter, we investigate the curse of dimensionality in the optimization of shallow neural networks. Utilizing theories from Wasserstein gradient flows, Barron spaces, and multivariate numerical

integration, we demonstrate that the population risk can decrease at an extremely slow rate, potentially requiring exponential training time to achieve a small error, even for smooth functions. Furthermore, we establish that the curse of dimensionality persists when the activation function is locally Lipschitz continuous. As a supplementary result, we show that a function with smoothness $r < d/2$ cannot be guaranteed to belong to the Barron space.

While our result is the first to analyze the impact of target function's regularity on the curse of dimensionality in neural network training, there are several open questions remaining. Here, we list some of them.

Explicit construction : Theorem 5.3.3 and Theorem 5.3.4 present the existence of $C^r(Q)$ functions suffering from the curse of dimensionality in training, but the proofs rely on probabilistic argument. Therefore, it would be interesting to exhibit an explicit examples of such functions and to characterize them structurally.

Loss function : Our analysis considers training with the L^2 loss function to learn target functions with specific regularity. On the other hand, classification tasks typically employ the cross-entropy function for neural network training to learn target functions that differ in nature from those in our analysis. It is therefore worth investigating whether the curse of dimensionality in neural network optimization-the requirement for exponential training time under gradient flow-occurs in this setting. The next question is designing a loss function that encodes a priori information about the target function-for example, physical constraints coming from a PDE. Such an information-rich loss function could potentially help avoid the curse of dimensionality in training.

Accelerated gradient descent : In our work, we focus on the Wasserstein gradient flow, which captures gradient descent training and stochastic gradient descent training with small step sizes. Exploring whether the curse persists—or can be mitigated—when accelerated methods (e.g., Nesterov or heavy-

ball dynamics) are employed is an appealing direction. Developing new acceleration methods to circumvent the curse of dimensionality is also a valuable research direction.

Finally, we pose functional-analytic questions related to our results.

Extension of Lemma 5.4.6 : There are two interesting questions related to Lemma 5.4.6: 1) What happens if $\alpha = \beta$? Can we obtain a similar statement? 2) Is it possible to replace $\limsup$ with $\lim$ in Lemma 5.4.6? If so, this may lead to a stronger negative result in neural network optimization.

Barron space and regularity of functions : The most important Barron space $\mathcal{B}$ corresponds to the ReLU activation function. While it is known that $C^r(Q) \subset \mathcal{B}$ for $r > d/2 + 1$, Corollary 5.3.2 states that $C^r(Q) \not\subset \mathcal{B}$ when $r < d/2$. An interesting research direction is to investigate the case $r \in [d/2, d/2 + 1]$.

Chapter 6: Neural Collapse in the Orthoplex Regime

6.1 Introduction

In this chapter, we consider a FNN (2.4) with L layers, where the final layer $h^{[L]} : \mathbb{R}^d \to \mathbb{R}^n$ is a linear layer serving as a linear classifier for the classification task. Specifically, we define $h^{[L]}(x) = W^\top x$ for a weight matrix $W \in \mathbb{R}^{d\times n}$. We refer to the activated output after $L-1$ layers as the feature map $h_\theta(\cdot) : \mathbb{R}^{m_0} \to \mathbb{R}^d$, where θ represents all the parameters $\{(W_l, b_l)\}_{l=1}^{L-1}$. Suppose the training dataset consists of n classes with m samples per class, denoted by $\{x_{k,i}\}_{k=1,\cdots,n,i=1,\cdots,m}$.

Neural collapse (NC), discovered by Vardan Papyan, X.Y. Han, and David Donoho in [87], is an intriguing phenomenon observed in the terminal phase of training deep neural networks with linear classifiers for classification tasks. The terminal phase of training refers to the stage where the trained classifiers interpolates the training set. Neural collapse describes how the class-mean feature vectors and the linear classifier weights converge to a symmetric geometric structure known as a simplex equiangular tight frame (ETF).

- **NC**$_1$ (Variability collapse) : The features from the same class converge to their mean feature vector. That is, for $1 \le i \le m$ and $1 \le k \le n$,

$$h_{k,i} = h_\theta(x_{k,i}) \longrightarrow \bar{h_k} = \frac{1}{m}\sum_{i=1}^{m} h_\theta(x_k, i).$$

- **NC**$_2$ (Convergence to the simplex ETF) : The mean feature vectors (after centering by their global mean) converges to a simplex ETF, a matrix $E \in \mathbb{R}^{d\times n}$ with $n \le d+1$ such that

$$E^T E = \alpha(I_n - \frac{1}{n}J_n)$$

for some $\alpha > 0$, and J_n be a $n \times n$ matrix with its entries all equal to 1.

- **NC**$_3$ (Self-duality) The last-layer linear classifier W also converges to a simplex ETF.
- **NC**$_4$ (Nearest decision rule) The network classifier converges to choosing the class that has the nearest class mean.

This empirical finding has been theoretically substantiated within the unconstrained feature model (UFM) [79], where the features $h_{k,i}$'s are treated as free optimization variables. This abstraction is grounded in the universal approximation theory of neural networks, which posits that sufficiently deep and wide neural networks possess ample expressive power. Such a framework is particularly relevant to modern neural networks in the overparametrized regime.

However, most existing studies are limited to the case $d + 1 \geq n$. In practice, the opposite case occurs in tasks such as person-identification. Since the emergence of NC confers important benefits-including enhanced generalization performance and robustness [87]-it is essential to extend its study to the general case of $n > d + 1$. To this end, the authors in [57] proposed generalized neural collapse (GNC), which extends NC framework to arbitrary d and n by introducing the concept of *softmax codes*. A formal statement of GNC is provided in Section 6.3.

While GNC was theoretically motivated, its initial justification relied on some strong geometric assumptions about the softmax codes. In this chapter, we provide a proof of GNC without such assumptions, in the regime $d+2 \leq n \leq 2d$, hereafter referred to as *orthoplex regime*. The rationale for focusing on this specific regime is further detailed in Remark 6.2.1. The contents of this chapter are based on Sections 2 and 3 of the recent preprint [1][1], co-authored with James Alcala, Rayna Andreeva, Vladimir A Kobzar, Dustin G Mixon, Shashank Sule, and Yangxinyu Xie.

[1]Theorem 6.2.2 was conjectured by Professor Dustin Mixon. The author of this thesis proved Theorem 6.2.2 and justified generalized neural collapse in $d + 2 \leq n \leq 2d$, which correspond to Sections 2 and 3 of [1].

6.2 Softmax codes in the Orthoplex regime

Before introducing GNC, we begin by investigating the equivalence between spherical codes and softmax codes in the orthoplex regime. Given a configuration $X = \{x_i\}_{i\in[n]}$ in the sphere $S^{d-1} \subseteq \mathbb{R}^d$, we denote

$$\alpha(X) := \max_{\substack{i,j\in[n]\\ i\neq j}} \langle x_i, x_j\rangle, \qquad \delta(X) := \min_{j\in[n]} \operatorname{dist}(x_j, \operatorname{conv}\{x_i\}_{i\in[n]\setminus\{j\}}).$$

We call X a **spherical code** if it minimizes $\alpha(X)$, and a **softmax code** if it maximizes $\delta(X)$. Note that both types of codes necessarily exist by a compactness argument.

Both types of codes are well understood in the 2-*dimensional regime* where $d = 2$, in which case they form the vertices of an origin-centered regular n-gon [57]. They are also well understood in the *simplex regime* where $n \leq d + 1$, in which case they form the vertices of an origin-centered regular simplex [57, 92]. In this chapter, we focus on the *orthoplex regime* where $n \in [d + 2, 2d]$. Spherical codes are already well understood in this regime:

Proposition 6.2.1 (Rankin's orthoplex bound; see Theorem 1 in [92])**.** *Fix $d \geq 2$ and $n \geq d + 2$. For every configuration $X = \{x_i\}_{i\in[n]}$ in S^{d-1}, it holds that*

$$\alpha(X) \geq 0.$$

Furthermore, equality is achievable if and only if $n \leq 2d$.

Remark 6.2.1. *Proposition 6.2.1 establishes that for $n \in [d + 2, 2d]$, a configuration X satisfying $\alpha(X) = 0$ is an optimal on the sphere, in the sense that its elements are maximally separated. On the*

other hand, when $n > 2d$, an optimal lower bound for $\alpha(X)$ remains generally unknown, which poses the difficulty in the analysis. We therefore focus our investigation on the orthoplex regime.

The main result of this section is the equivalence of softmax codes and spherical codes.

Theorem 6.2.2. *Fix $d \geq 2$ and $n \geq d+2$. For every configuration $X = \{x_i\}_{i\in[n]}$ in S^{d-1}, it holds that*

$$\delta(X) \leq 1.$$

Furthermore, $\delta(X) = 1$ if and only if $\alpha(X) = 0$.

We present Radon's theorem on convex sets, which is pivotal to the proof of Theorem 6.2.2.

Theorem 6.2.3 (Radon's theorem [90])**.** *Any set of $d+2$ points in $\mathbb{R}^d$ can be partitioned into two sets, where the intersection of their convex hulls is nonempty.*

Our proof of Theorem 6.2.2 uses the following lemma.

Lemma 6.2.4. *Fix $d \geq 2$ and $n \geq d+2$. For every configuration $X = \{x_i\}_{i\in[n]}$ in S^{d-1}, it holds that*

$$\alpha(X) > 0 \qquad \Longrightarrow \qquad \delta(X) < 1.$$

Proof. By Radon's theorem, there exists a partition $[n] = A \sqcup B$ with both A and B nonempty such that the intersection

$$K := \operatorname{conv}\{x_i\}_{i\in A} \cap \operatorname{conv}\{x_i\}_{i\in B}$$

is also nonempty. We proceed by casing on whether K contains the origin.

Case I: $0 \in K$. Fix $j, k \in [n]$ such that $\langle x_j, x_k\rangle > 0$, and denote $\theta := \arccos\langle x_j, x_k\rangle \in [0, \frac{\pi}{2})$. Without

loss of generality, we have $j \in A$, and so

$$0 \in K \subseteq \operatorname{conv}\{x_i\}_{i\in B} \subseteq \operatorname{conv}\{x_i\}_{i\in[n]\setminus\{j\}}.$$

Since $\operatorname{conv}\{x_i\}_{i\in[n]\setminus\{j\}}$ contains both x_k and 0, by convexity, it also contains the entire line segment $\operatorname{conv}\{x_k, 0\}$. Thus,

$$\delta(X) \leq \operatorname{dist}(x_j, \operatorname{conv}\{x_i\}_{i\in[n]\setminus\{j\}}) \leq \operatorname{dist}(x_j, \operatorname{conv}\{x_k, 0\}) = \sin\theta < 1.$$

Case II: $0 \notin K$. Let v denote the projection of 0 onto K. (Notably, v is nonzero since $0 \notin K$.) Since $v \in K \subseteq \operatorname{conv}\{x_i\}_{i\in[n]}$, we may express v as a convex combination:

$$v = \sum_{i=1}^{n} a_i x_i.$$

It follows that

$$\|v\|^2 = \langle v, v\rangle = \left\langle \sum_{i=1}^{n} a_i x_i, v \right\rangle = \sum_{i=1}^{n} a_i \langle x_i, v\rangle \leq \max_{i\in[n]} \langle x_i, v\rangle,$$

i.e., there exists $j \in [n]$ such that $\langle x_j, v\rangle \geq \|v\|^2$. This in turn implies

$$\|x_j - v\|^2 = 1 - 2\langle x_j, v\rangle + \|v\|^2 \leq 1 - 2\|v\|^2 + \|v\|^2 = 1 - \|v\|^2.$$

Without loss of generality, we have $j \in A$, and so

$$v \in K \subseteq \operatorname{conv}\{x_i\}_{i\in B} \subseteq \operatorname{conv}\{x_i\}_{i\in[n]\setminus\{j\}}.$$

As such,

$$\delta(X) \leq \operatorname{dist}(x_j, \operatorname{conv}\{x_i\}_{i\in[n]\setminus\{j\}}) \leq \|x_j - v\| \leq \sqrt{1 - \|v\|^2} < 1,$$

where the last step uses the fact that v is nonzero. □

Now we present the proof of Theorem 6.2.2.

Proof of Theorem 6.2.2. The bound $\delta(X) \leq 1$ is given by Lemma C.9 in [57]. Meanwhile, the "only if" direction of the "Furthermore" statement follows from combining Proposition 6.2.1 with Lemma 6.2.4. It remains to prove the "if" direction. To this end, suppose $\alpha(X) = 0$. Then for each $j \in [n]$, it holds that every x_i with $i \neq j$ resides in the halfspace

$$H_j := \{z \in \mathbb{R}^d : \langle z, x_j\rangle \leq 0\}.$$

Since H_j is convex, it follows that $\operatorname{conv}\{x_i\}_{i\in[n]\setminus\{j\}} \subseteq H_j$, and so

$$\operatorname{dist}(x_j, \operatorname{conv}\{x_i\}_{i\in[n]\setminus\{j\}}) \geq \operatorname{dist}(x_j, H_j) = 1,$$

where the last step follows from the projection theorem. Minimizing over $j \in [n]$ then gives $\delta(X) \geq 1$, but since $\delta(X) \leq 1$ in general, we conclude that $\delta(X) = 1$. □

6.3 Generalized Neural Collapse

This section completes and extends several theorems and conjectures presented in [57] for the orthoplex regime $n \in [d+2, 2d]$. Consistent with the notation from earlier sections, we interpret n as the number of classes (denoted as K in [57]) and d as the dimension of the feature space.

6.3.1 HardMax problem and Neural Collapse

While substantial literature exists on neural collapse for the regime $n \leq d+1$, the scenario where $n \geq d+2$ remains largely unexplored. In [57], the authors consider a realistic setup for the general case. [57] studies neural collapse by optimizing the cross-entropy loss with a finite temperature τ, under the UFM [79] with spherical constraints on classifier weights $W \in \mathbb{R}^{d\times n}$ and on unconstrained features $H \in \mathbb{R}^{d\times mn}$:

$$\min_{W,H} \frac{1}{mn} \sum_{k=1}^{n} \sum_{i=1}^{m} \mathcal{L}_{CE}(W^\top T h_{k,i}, Y, \tau), \quad \mathcal{L}_{CE}(W^\top h_{k,i}, Y, \tau) = -\sum_{k=1}^{n} \log \left(\frac{\exp(\langle w_k, h_{k,i}\rangle/\tau)}{\sum_j \exp(\langle w_j, h_{k,i}\rangle/\tau)} \right), \tag{6.1}$$

where $Y = \{\delta_k\}_{k=1}^n$ denotes the one-hot true labels, m denotes the number of training samples in each class, $w_k \in S^{d-1}$ denotes the kth column of W, and $h_{k,i} \in S^{d-1}$ denotes the feature of the ith sample in the kth class. The feature matrix H is written as

$$H = [H_1 \quad H_2 \cdots \quad H_n] \in \mathbb{R}^{d\times mn}$$

with $H_k = [h_{k,1} \quad \cdots \quad h_{k,m}] \in \mathbb{R}^{d\times m}$. Motivated by the convention that the cross-entropy is often optimized under small τ, [57] proves the following convergence result:

Lemma 6.3.1. *[57, Lemma 3.1]*

$$\limsup_{\tau\to 0} \left(\arg\min_{W,H} \frac{1}{mn} \sum_{k=1}^{n} \sum_{i=1}^{m} \mathcal{L}_{CE}(W^\top h_{k,i}, Y, \tau) \right) \subseteq \arg\min_{W,H} \mathcal{L}_{HM}(W, H),$$

where the HardMax problem is defined as

$$\mathcal{L}_{HM}(W,H) := \max_{k\in[n]} \max_{i\in[m]} \max_{k'\neq k} \langle w_k - w_{k'}, h_{k,i}\rangle. \tag{6.2}$$

Inspired by the convergence to the HarMax problem, [57] analyzes (6.2) to establish generalized neural collapse (GNC). One of the key findings is that the set of softmax codes is exactly the same as the set of solution W^* to (6.2).

Theorem 6.3.2 (GNC_2, Theorem C.7 in [57])**.** *If (W^*, H^*) is an optimal solution to (6.2), then W^* is a softmax code. Conversely, for any softmax code W', there exists H' such that (W', H') is an optimal solution to (6.2).*

6.3.2 Tammes problem, Softmax codes, and Rattlers

The other results for GNC hold under some assumptions on the Tammes problem and softmax codes. In this section, we begin by introducing the Tammes problem and *rattlers*, and then prove several useful properties needed for GNC in the orthoplex regime.

Definition 6.3.3. *(Rattler of softmax code) A rattler associated with a softmax code $X = \{x_i\}_{i\in[n]} \subset S^{d-1}$ is an index $j \in [n]$ for which*

$$\delta(X) \neq \operatorname{dist}(x_j, \operatorname{conv}\{x_i\}_{i\in[n]\setminus\{j\}})$$

In other words, the rattlers are points in a softmax code with no neighbors at the minimum one-to-rest distance.

Definition 6.3.4. *(Tammes problem and its rattler) The Tammes problem is*

$$\max_{|X|=n, X\subset S^{d-1}} \min_{i\in[n]} \min_{k\in[n]\setminus\{i\}} \|x_i - x_k\|.$$

A rattler of the Tammes problem for an optimal solution $X^* = \{x_1^*, \cdots, x_n^*\}$ is an index $j \in [n]$ for which

$$\min_{i\in[n]} \min_{k\in[n]\setminus\{i\}} \|x_i^* - x_k^*\| \neq \min_{k\in[n]\setminus\{j\}} \|x_j^* - x_k^*\|.$$

In other words, a rattler is a point where the minimum distance between other points is not maximized.

Geometrically, the Tammes problem requires a configuration $X = \{x_i\}_{i\in[n]} \subset S^{d-1}$ in which the minimum distance between any pair of points is maximized. While this is a different optimization compared to the problem maximizing $\delta(X)$, they share the same solution in the 2-dimensional regime and the simplex regime [57, Theorem 3.8]. Here, we present a new result that they are also equivalent in the orthoplex regime.

Theorem 6.3.5. *Let $d+2 \leq n \leq 2d$. Then, the Tammes problem and softmax codes are equivalent.*

Proof. Since $2\langle x_i, x_j\rangle = \|x_i\|^2 + \|x_j\|^2 - \|x_i - x_j\|^2 = 2 - \|x_i - x_j\|^2$, it is easy to see that the Tammes problem is equal to finding a minimizer of $\alpha(X)$. Hence the solution set of the Tammes problem is equal to the set of spherical codes. On the other hand, spherical codes are softmax codes and vice verca in the orthoplex regime by Theorem 6.2.2. □

Now we prove that a softmax code does not have any rattler in the orthoplex regime.

Proposition 6.3.6. *Let $d+2 \leq n \leq 2d$ and $X = \{x_1, \cdots, x_n\} \subset S^{d-1}$ be a softmax code. Then there exists no rattler associated with this softmax code X.*

Proof. By Radon's theorem [90], there exists a partition $[n] = A \sqcup B$ with both A and B nonempty such that the intersection

$$K := \mathrm{conv}\{x_i\}_{i\in A} \cap \mathrm{conv}\{x_i\}_{i\in B}$$

is also nonempty. Note that by [57, Theorem 3.3], we have $\delta(X) = 1$. Therefore, if $0 \notin K$, then from Case II in the proof of Lemma 6.2.4, we get a contradiction. Hence, $0 \in K$ holds.

Now choose an arbitrary $x_i \in X$. Without loss of generality, assume $x_i \in A$. Then $B \subset \{x_j\}_{j\neq i}$ holds and thus $\mathrm{conv}\{x_j\}_{j\in B} \subset \mathrm{conv}\{x_j\}_{j\neq i}$ holds. Since $0 \in K$ and $K \subset \mathrm{conv}\{x_j\}_{j\in B}$, we have $0 \in \mathrm{conv}\{x_j\}_{j\neq i}$. This implies

$$\mathrm{dist}(x_i, \mathrm{conv}\{x_j\}_{j\neq i}) \leq \|x_i - 0\| = 1.$$

On the other hand, since $\delta(X) = 1$, we have $\mathrm{dist}(x_i, \mathrm{conv}\{x_j\}_{j\neq i}) \geq \delta(X) = 1$. Therefore we conclude that $\mathrm{dist}(x_i, \mathrm{conv}\{x_j\}_{j\neq i}) = 1 = \delta(X)$. Since this holds for any choice $i \in [n]$, X has no rattler. □

Using the nonexistence of rattlers, we can also characterize the projections of each point onto the convex hull of others.

Lemma 6.3.7. *Let $d + 2 \leq n \leq 2d$ and $X = \{x_1, \cdots, x_n\} \subset S^{d-1}$ be a softmax code. Then for any $i \in [n]$, the projection of x_i onto the convex hull of $\{x_j\}_{j\neq i}$ is 0. Moreover, there exists some $j \neq i$ such that $\langle x_i, x_j\rangle = 0$*

Proof. Fix any $i \in [n]$. From Lemma 6.3.6, $\mathrm{dist}(x_i, \mathrm{conv}\{x_j\}_{j\neq i}) = 1$ holds. Let $v_i = \mathrm{proj}_{\mathrm{conv}\{x_j\}_{j\neq i}}(x_i)$ be the projection of x_i onto the convex hull of $\{x_j\}_{j\neq i}$. Then there exist nonnegative coefficients $\{\alpha_j\}_{j\neq i}$ where their sum is equal to 1 and $v_i = \sum_{j\neq i} \alpha_j x_j$. Since $\|x_i - v_i\| = \mathrm{dist}(x_i, \mathrm{conv}\{x_j\}_{j\neq i}) =$

1, we have $1 = \|x_i - v_i\|^2 = \|x_i\|^2 + \|v_i\|^2 - 2\langle x_i, v_i\rangle$. Using $\|x_i\| = 1$, we get

$$\|v_i\|^2 = 2\langle x_i, v_i\rangle = 2\sum_{j\neq i} \alpha_j \langle x_i, x_j\rangle.$$

On the other hand, by Theorem 6.2.2, we have $\langle x_i, x_j\rangle \leq 0$ for $j \neq i$. Therefore,

$$0 \leq \|v_i\|^2 = 2\sum_{j\neq i} \alpha_j \langle x_i, x_j\rangle \leq 0. \tag{6.3}$$

This implies $\mathrm{proj}_{\mathrm{conv}\{x_j\}_{j\neq i}}(x_i) = v_i = 0$.

Since $v_i = 0$, the inequalities in (6.3) must be equalities. Therefore, there exists $j \neq i$ such that $\langle x_i, x_j\rangle = 0$.

□

Finally, we prove that there is no rattler of the Tammes problem for an optimal solution in the orthoplex regime.

Proposition 6.3.8. *Let $d+2 \leq n \leq 2d$. Then there is no rattler of the Tammes problem for an optimal solution.*

Proof. Let $W = \{w_1, \cdots, w_n\} \subset S^{d-1}$ be an optimal solution to the Tammes problem. Then $\alpha(W) = 0$ holds. This implies

$$\min_{k\in[n]\setminus\{i\}} \|w_i - w_k\| = \min_{k\in[n]\setminus\{i\}} \sqrt{2 - 2\langle w_i, w_k\rangle} = \sqrt{2 - \max_{k\in[n]\setminus\{i\}} \langle w_i, w_k\rangle} \geq \sqrt{2 - \alpha(W)} = \sqrt{2}$$

for all $i \in [n]$.

On the other hand, for any $i \in [n]$, there exists $j \neq i$ such that $\langle w_i, w_j\rangle = 0$ by Lemma 6.3.7.

Hence we get

$$\min_{k\in[n]\setminus\{i\}} \|w_i - w_k\| = \min_{k\in[n]\setminus\{i\}} \sqrt{2-2\langle w_i, w_k\rangle} \leq \sqrt{2-\langle w_i, w_j\rangle} = \sqrt{2}.$$

Therefore, $\min_{k\in[n]\setminus\{i\}} \|w_i - w_k\| = \sqrt{2}$ for all $i \in [n]$, and thus W has no rattler associated with the Tammes problem. □

6.3.3 Variability Collapse and Self-Duality

Now we introduce the remaining results of GNC - *variability collapse* (GNC_1) and *self-duality* (GNC_3).

Theorem 6.3.9 (GNC_1, Theorem 3.5 in [57])**.** *Let (W^*, H^*) be an optimal solution to (6.2). For all i that is not a rattler associated with softmax code W^*, it holds that*

$$h^*_{i,1} = \cdots = h^*_{i,m} = \text{proj}_{S^{d-1}}\left(w^*_i - \text{proj}_{\text{conv}(w^*_j)_{j\neq i}}(w^*_i)\right).$$

Theorem 6.3.10 (GNC_3, Theorem 3.7 in [57])**.** *For any K, d such that both Tammes problem and softmax code have no rattler, the following statements are equivalent:*

(a) Any optimal solution (W^*, H^*) to (6.2) satisfies $h^*_{i,l} = w^*_i$ for all $i \in [n], l \in [m]$.

(b) The Tammes problem and softmax codes are equivalent.

GNC_1 can be interpreted as stating that all features belonging to the same class collapse to their corresponding class mean. GNC_3 indicates that the linear classifier weights converge to the class-mean features. However, both statements rely on assumptions regarding rattlers and on the equivalence of

the Tammes problem and softmax codes, which makes them incomplete.

On the other hand, from the results that we have established in the previous sections, we can give a complete characterization about GNC in the orthoplex regime without any additional assumptions.

Theorem 6.3.11 (Self-Duality of the HardMax problem)**.** *Let $d+2 \leq n \leq 2d$. Then the following statements hold.*

(a) Any optimal solution (W^*, H^*) to (6.2) satisfies $h^*_{i,l} = w^*_i$ for all $i \in [n], l \in [m]$.

(b) For a softmax code W, there exists a unique $H \in \mathbb{R}^{d\times mn}$ such that (W, H) is an optimal solution to (6.2) where

$$H = [H_1 \quad H_2 \cdots \quad H_n] \in \mathbb{R}^{d\times mn}$$

with $H_k = [h_{k,1} \quad \cdots \quad h_{k,m}] \in \mathbb{R}^{d\times m}$ and $h^*_{i,l} = w^*_i$ for all $i \in [n], l \in [m]$.

Proof. We first prove (a). Let (W^*, H^*) be an optimal solution to (6.2). By GNC_2, W^* is a softmax code. Then, W^* does not have any rattler by Proposition 6.3.6. Hence by GNC_1, it holds that

$$h^*_{i,1} = \cdots = h^*_{i,m} = \text{proj}_{S^{d-1}}\left(w^*_i - \text{proj}_{\text{conv}(w^*_j)_{j\neq i}}(w^*_i)\right)$$

for any $i \in [n]$. On the other hand, Lemma 6.3.7 gives $\text{proj}_{\text{conv}(w^*_j)_{j\neq i}}(w^*_i) = 0$ for all $i \in [n]$. Therefore,

$$\text{proj}_{S^{d-1}}\left(w^*_i - \text{proj}_{\text{conv}(w^*_j)_{j\neq i}}(w^*_i)\right) = \text{proj}_{S^{d-1}}\left(w^*_i - 0\right) = w^*_i.$$

This completes the proof of (a). The proof for (b) follows directly from GNC_2 and (a). □

6.4 Conclusion and Open Questions

In this chapter, we established the equivalence between softmax codes and spherical codes within the orthoplex regime and theoretically justified GNC independently of the assumptions provided in [57]. Despite these advances, several intriguing and challenging open questions remain regarding both spherical codes and neural collapse. We highlight some of these directions below.

Beyond the Orthoplex Regime : A natural question is whether the equivalence between spherical codes and softmax codes persists when $n > 2d$. Furthermore, the role of "rattlers" in this regime remains to be elucidated. Addressing these questions will be instrumental in understanding neural collapse across arbitrary settings.

Practical Applications of GNC : Neural collapse is a significant area of study, not only for its theoretical elegance but also for its practical utility. For instance, neural collapse has been employed to enhance transfer learning [63] and improve out-of-distribution detection [68]. As the theory of neural collapse for generalized cases matures, leveraging these insights to solve diverse practical problems will be a promising avenue for research.

Analysis Beyond the UFM : Numerous studies have theoretically analyzed neural collapse [113, 126, 136], most of which rely on the UFM to investigate the optimization landscape. While the UFM is valid when a neural network is sufficiently deep and wide to approximate various function classes, it is inapplicable to shallow neural network architectures. Therefore, a compelling research direction involves the emergence of neural collapse in shallow neural networks beyond the UFM framework, such as in [53]. Another critical challenge lies in characterizing this phenomenon through the study of optimization dynamics without relying on the UFM. Recent work [78] has begun this effort by studying the training dynamics of shallow ReLU networks for orthogonally separable data.

Intermediate Layers : While neural collapse describes a geometric symmetry in feature vectors and classifier weights at the penultimate layer, a fundamental question remains regarding the intermediate layers. Does a similar phenomenon emerge in the layers preceding the penultimate layer when a network is fully optimized? If neural collapse is absent in the intermediate layers, what then serves as the primary driving force behind its emergence?

Bibliography

[1] James Alcala, Rayna Andreeva, Vladimir A. Kobzar, Dustin G. Mixon, Sanghoon Na, Shashank Sule, and Yangxinyu Xie. Neural collapse in the orthoplex regime. *arXiv preprint arXiv:2603.20587*, 2026.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *International Conference on Machine Learning*, 97:242–252, 2019.

[3] William F. Ames. *Numerical methods for partial differential equations*. Academic Press, 2014.

[4] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

[5] Francis Bach. *Learning theory from first principles*. MIT Press, 2024.

[6] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019.

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

[8] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[9] John J. Benedetto and Onur Oktay. Pointwise comparison of PCM and $\Sigma\Delta$ quantization. *Constructive Approximation*, 32(1):131–158, 2010.

[10] John J. Benedetto, Onur Oktay, and Aram Tangboondouangjit. Complex Sigma-Delta ($\Sigma\Delta$) quantization algorithms for finite frames. *Radon Transforms, Geometry, and Wavelets*, 464:27–49, 2008.

[11] John J. Benedetto, Alexander M. Powell, and Özgür Yılmaz. Second-order Sigma–Delta ($\Sigma\Delta$) quantization of finite frame expansions. *Applied and Computational Harmonic Analysis*, 20(1):126–148, 2006.

[12] John J. Benedetto, Alexander M. Powell, and Özgür Yılmaz. Sigma-delta ($\Sigma\Delta$) quantization and finite frames. *IEEE Transactions on Information Theory*, 52(5):1990–2005, 2006.

[13] Julius Berner, Philipp Grohs, and Arnulf Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020.

[14] Bernhard G. Bodmann and Vern I. Paulsen. Frame paths and error bounds for sigma–delta quantization. *Applied and Computational Harmonic Analysis*, 22(2):176–197, 2007.

[15] Vivien Cabannes, Loucas Pillaud-Vivien, Francis Bach, and Alessandro Rudi. Overcoming the curse of dimensionality with Laplacian regularization in semi-supervised learning. *Advances in Neural Information Processing Systems*, 34:30439–30451, 2021.

[16] Peter G. Casazza and Gitta Kutyniok, editors. *Finite frames: Theory and applications*. Springer Science & Business Media, 2012.

[17] Ke Chen, Chunmei Wang, and Haizhao Yang. Deep operator learning lessens the curse of dimensionality for PDEs. *Transactions on Machine Learning Research*, 2023.

[18] Ke Chen, Chunmei Wang, and Haizhao Yang. Let data talk: data-regularized operator learning theory for inverse problems. *arXiv preprint arXiv:2310.09854*, 2023.

[19] Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao. Efficient approximation of deep ReLU networks for functions on low dimensional manifolds. *Advances in Neural Information Processing Systems*, 32, 2019.

[20] Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao. Nonparametric regression on low-dimensional manifolds using deep ReLU networks: Function approximation and statistical recovery. *Information and Inference: A Journal of the IMA*, 11(4):1203–1253, 2022.

[21] Zhengdao Chen, Eric Vanden-Eijnden, and Joan Bruna. On feature learning in neural networks with global convergence guarantees. *International Conference on Learning Representations*, 2022.

[22] Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in Neural Information Processing Systems*, 31, 2018.

[23] Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. *Conference on Learning Theory*, 125:1305–1338, 2020.

[24] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[25] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in Neural Information Processing Systems*, 28, 2015.

[26] Wojciech Czaja and Martin Ehler. Schroedinger eigenmaps for the analysis of biomedical data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1274–1280, 2012.

[27] Wojciech Czaja and Sanghoon Na. Frame quantization of neural networks. *Journal of Fourier Analysis and Applications*, 31(4):49, 2025.

[28] Ingrid Daubechies and Ron DeVore. Approximating a bandlimited function using very coarsely quantized data: A family of stable sigma-delta modulators of arbitrary order. *Annals of Mathematics*, 158(2):679–710, 2003.

[29] Ronald A. DeVore, Ralph Howard, and Charles Micchelli. Optimal nonlinear approximation. *Manuscripta Mathematica*, 63(4):469–478, 1989.

[30] David L. Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[31] Simon Du and Jason Lee. On the power of over-parametrization in neural networks with quadratic activation. *International Conference on Machine Learning*, 80:1329–1338, 2018.

[32] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *International Conference on Machine Learning*, 97:1675–1685, 2019.

[33] Simon Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *International Conference on Learning Representations*, 2019.

[34] Weinan E, Chao Ma, and Lei Wu. The Barron Space and the Flow-Induced Function Spaces for Neural Network Models. *Constructive Approximation*, 55:369–406, 2022.

[35] Weinan E and Stephan Wojtowytsch. On the Banach Spaces Associated with Multi-Layer ReLU Networks: Function Representation, Approximation Theory and Gradient Descent Dynamics. *CSIAM Transactions on Applied Mathematics*, 1(3):387–440, 2020.

[36] Weinan E and Stephan Wojtowytsch. Kolmogorov width decay and poor approximators in machine learning: Shallow neural networks, random feature models and neural tangent kernels. *Research in the Mathematical Sciences*, 8(1):1–28, 2021.

[37] Weinan E and Stephan Wojtowytsch. Representation formulas and pointwise properties for Barron functions. *Calculus of Variations and Partial Differential Equations*, 61(2):46, 2022.

[38] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[39] Philipp Grohs, Shokhrukh Ibragimov, Arnulf Jentzen, and Sarah Koppensteiner. Lower bounds for artificial neural network approximations: A proof that shallow neural networks fail to overcome the curse of dimensionality. *Journal of Complexity*, 77:101746, 2023.

[40] Philipp Grohs and Gitta Kutyniok, editors. *Mathematical aspects of deep learning*. Cambridge University Press, 2022.

[41] Yiqi Gu, Haizhao Yang, and Chao Zhou. Selectnet: Self-paced learning for high-dimensional partial differential equations. *Journal of Computational Physics*, 441:110444, 2021.

[42] C. Sinan Güntürk. One-bit sigma-delta quantization with exponential accuracy. *Communications on Pure and Applied Mathematics*, 56(11):1608–1630, 2003.

[43] C. Sinan Güntürk. Approximating a bandlimited function using very coarsely quantized data: improved error estimates in sigma-delta modulation. *Journal of the American Mathematical Society*, 17(1):229–242, 2004.

[44] C. Sinan Güntürk, Mark Lammers, Alex Powell, Rayan Saab, and Özgür Yılmaz. Sigma delta quantization for compressed sensing. *44th Annual Conference on Information Sciences and Systems (CISS)*, 2010.

[45] C. Sinan Güntürk, Mark Lammers, Alex Powell, Rayan Saab, and Özgür Yılmaz. Sobolev duals for random frames and $\Sigma\Delta$ quantization of compressed sensing measurements. *Foundations of Computational Mathematics*, 13:1–36, 2013.

[46] C. Sinan Güntürk and Weilin Li. Approximation of functions with one-bit neural networks. *arXiv preprint arXiv:2112.09181*, 2021.

[47] Cong Guo, Yuxian Qiu, Jingwen Leng, Xiaotian Gao, Chen Zhang, Yunxin Liu, Fan Yang, Yuhao Zhu, and Minyi Guo. SQuant: On-the-fly data-free quantization via diagonal Hessian approximation. *International Conference on Learning Representations*, 2022.

[48] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[50] Tjeerd Jan Heeringa, Len Spek, Felix L. Schwenninger, and Christoph Brune. Embeddings between Barron spaces with higher-order activation functions. *Applied and Computational Harmonic Analysis*, 73:101691, 2024.

[51] Aicke Hinrichs, Erich Novak, Mario Ullrich, and H. Woźniakowski. The curse of dimensionality for numerical integration of smooth functions. *Mathematics of Computation*, 83(290):2853–2863, 2014.

[52] Aicke Hinrichs, Erich Novak, Mario Ullrich, and Henryk Woźniakowski. Product rules are optimal for numerical integration in classical smoothness spaces. *Journal of Complexity*, 38:39–49, 2017.

[53] Wanli Hong and Shuyang Ling. Beyond unconstrained features: Neural collapse for shallow neural networks with general data. *arXiv preprint arXiv:2409.01832*, 2024.

[54] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[55] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. *International Conference on Machine Learning*, 139:4466–4475, 2021.

[56] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[57] Jiachen Jiang, Jinxin Zhou, Peng Wang, Qing Qu, Dustin Mixon, Chong You, and Zhihui Zhu. Generalized neural collapse for a large number of classes. *International Conference on Machine Learning*, 235:22010–22041, 2024.

[58] Joseph Dennis Kolesar. $\Sigma\Delta$ *modulation and correlation criteria for the construction of finite frames arising in communication theory*. PhD thesis, University of Maryland, College Park, 2004.

[59] Felix Krahmer, Rayan Saab, and Özgür Yilmaz. Sigma–delta quantization of sub-gaussian frame expansions and its application to compressed sensing. *Information and Inference: A Journal of the IMA*, 3(1):40–58, 2014.

[60] Lorenz Kummer, Kevin Sidak, Tabea Reichmann, and Wilfried Gansterer. Adaptive Precision Training (AdaPT): A dynamic quantized training approach for DNNs. *SIAM International Conference on Data Mining*, pages 559–567, 2023.

[61] John M. Lee. *Introduction to smooth manifolds*. Springer New York, New York, NY, 2003.

[62] Binghui Li, Jikai Jin, Han Zhong, John Hopcroft, and Liwei Wang. Why robust generalization in deep learning is difficult: Perspective of expressive power. *Advances in Neural Information Processing Systems*, 35:4370–4384, 2022.

[63] Xiao Li, Sheng Liu, Jinxin Zhou, Xinyu Lu, Carlos Fernandez-Granda, Zhihui Zhu, and Qing Qu. Understanding and Improving Transfer Learning of Deep Models via Neural Collapse. *Transactions on Machine Learning Research*, 2024.

[64] Yuanzhi Li, Tengyu Ma, and Hongyang R. Zhang. Learning over-parametrized two-layer neural networks beyond NTK. *Conference on Learning Theory*, 125:2613–2682, 2020.

[65] Zhong Li, Chao Ma, and Lei Wu. Complexity measures for neural networks with general activation functions using path-based norms. *arXiv preprint arXiv:2009.06132*, 2020.

[66] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

[67] Hao Liu, Haizhao Yang, Minshuo Chen, Tuo Zhao, and Wenjing Liao. Deep nonparametric estimation of operators between infinite dimensional spaces. *Journal of Machine Learning Research*, 25(24):1–67, 2024.

[68] Litian Liu and Yao Qin. Detecting out-of-distribution through the lens of neural collapse. *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15424–15433, 2025.

[69] Ziang Long, Penghang Yin, and Jack Xin. Recurrence of optimum for training weight and activation quantized networks. *Applied and Computational Harmonic Analysis*, 62:41–65, 2023.

[70] Jianfeng Lu, Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5):5465–5506, 2021.

[71] Jianfeng Lu and Stefan Steinerberger. Neural collapse under cross-entropy loss. *Applied and Computational Harmonic Analysis*, 59:224–241, 2022.

[72] Yiping Lu, Chao Ma, Yulong Lu, Jianfeng Lu, and Lexing Ying. A mean field analysis of deep ResNet and beyond: Towards provably optimization via overparameterization from depth. *International Conference on Machine Learning*, 119:6426–6436, 2020.

[73] Tao Luo and Haizhao Yang. Two-layer neural networks for partial differential equations: Optimization and generalization theory. *Handbook of Numerical Analysis*, 25:515–554. Elsevier, 2024.

[74] Eric Lybrand and Rayan Saab. A greedy algorithm for quantizing neural networks. *Journal of Machine Learning Research*, 22(156):1–38, 2021.

[75] Johannes Maly and Rayan Saab. A simple approach for quantizing neural networks. *Applied and Computational Harmonic Analysis*, 66:138–150, 2023.

[76] Song Mei, Andrea Montanari, and Phan-Minh 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.

[77] Hrushikesh N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, 8(1):164–177, 1996.

[78] Hancheng Min, Zhihui Zhu, and Rene Vidal. Neural Collapse under Gradient Flow on Shallow ReLU Networks for Orthogonally Separable Data. *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[79] Dustin G. Mixon, Hans Parshall, and Jianzong Pi. Neural collapse with unconstrained features. *Sampling Theory, Signal Processing, and Data Analysis*, 20(2):11, 2022.

[80] Hadrien Montanelli, Haizhao Yang, and Qiang Du. Deep ReLU Networks Overcome the Curse of Dimensionality for Generalized Bandlimited Functions. *Journal of Computational Mathematics*, 39(6):801–815, 2021.

[81] Sanghoon Na and Haizhao Yang. Curse of dimensionality in neural network optimization. *arXiv preprint arXiv:2502.05360*, 2025.

[82] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. *International Conference on Machine Learning*, 119:7197–7206, 2020.

[83] Yury Nahshan, Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Alex M. Bronstein, and Avi Mendelson. Loss aware post-training quantization. *Machine Learning*, 110(11):3245–3262, 2021.

[84] Atsushi Nitanda, Denny Wu, and Taiji Suzuki. Particle dual averaging: Optimization of mean field neural network with global convergence rate analysis. *Advances in Neural Information Processing Systems*, 34:19608–19621, 2021.

[85] Onur Oktay. *Frame quantization theory and equiangular tight frames*. PhD thesis, University of Maryland, College Park, 2007.

[86] Samet Oymak and Mahdi Soltanolkotabi. Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1):84–105, 2020.

[87] Vardan Papyan, XY Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[88] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and others. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[89] Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

[90] Johann Radon. Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Mathematische Annalen*, 83(1):113–115, 1921.

[91] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[92] Robert Alexander Rankin. The closest packing of spherical caps in n dimensions. *Glasgow Mathematical Journal*, 2(3):139–144, 1955.

[93] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *European Conference on Computer Vision*, pages 525–542, 2016.

[94] Grant M. Rotskoff and Eric Vanden-Eijnden. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *stat*, 1050:22, 2018.

[95] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[96] Rayan Saab, Rongrong Wang, and Özgür Yılmaz. Quantization of compressive samples with stable and robust recovery. *Applied and Computational Harmonic Analysis*, 44(1):123–143, 2018.

[97] Stefano Sarao Mannelli, Eric Vanden-Eijnden, and Lenka Zdeborová. Optimization and generalization of shallow neural networks with quadratic activation functions. *Advances in Neural Information Processing Systems*, 33:13445–13455, 2020.

[98] Uri Shaham, Alexander Cloninger, and Ronald R. Coifman. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537–557, 2018.

[99] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[100] Ohad Shamir. Exponential convergence time of gradient descent for one-dimensional deep linear neural networks. *Conference on Learning Theory*, 99:2691–2713, 2019.

[101] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep Network Approximation Characterized by Number of Neurons. *Communications in Computational Physics*, 28(5):1768–1811, 2020.

[102] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network with approximation error being reciprocal of width to power of square root of depth. *Neural Computation*, 33(4):1005–1036, 2021.

[103] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Neural network approximation: Three hidden layers are enough. *Neural Networks*, 141:160–173, 2021.

[104] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Optimal approximation rate of ReLU networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, 157:101–135, 2022.

[105] Jonathan W. Siegel and Jinchao Xu. Approximation rates for neural networks with general activation functions. *Neural Networks*, 128:313–321, 2020.

[106] Jonathan W. Siegel and Jinchao Xu. High-order approximation rates for shallow neural networks with cosine and ReLUk activation functions. *Applied and Computational Harmonic Analysis*, 58:1–26, 2022.

[107] Jonathan W. Siegel and Jinchao Xu. Sharp bounds on the approximation rates, metric entropy, and n-widths of shallow neural networks. *Foundations of Computational Mathematics*, 24(2):481–537, 2024.

[108] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[109] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D. Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018.

[110] Taiji Suzuki. Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality. *International Conference on Learning Representations*, 2019.

[111] Aram Tangboondouangjit. *Sigma-delta quantization: Number theoretic aspects of refining quantization error*. PhD thesis, University of Maryland, College Park, 2006.

[112] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[113] Tom Tirer and Joan Bruna. Extended unconstrained features model for exploring deep neural collapse. *International Conference on Machine Learning*, pages 21478–21505, 2022.

[114] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

[115] Cédric Villani. *Optimal Transport: Old and New*. Vol. 338. Springer, 2009.

[116] Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with Lipschitz functions. *Journal of Machine Learning Research*, 5:669–695, 2004.

[117] Hongyu Wang, Shuming Ma, Lingxiao Ma, Lei Wang, Wenhui Wang, Li Dong, Shaohan Huang, Huaijie Wang, Jilong Xue, Ruiping Wang, and others. BitNet: 1-bit pre-training for large language models. *Journal of Machine Learning Research*, 26(125):1–29, 2025.

[118] Yang Wang. Sigma–Delta quantization errors and the traveling salesman problem. *Advances in Computational Mathematics*, 28:101–118, 2008.

[119] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. QDrop: Randomly Dropping Quantization for Extremely Low-bit Post-Training Quantization. *International Conference on Learning Representations*, 2022.

[120] Stephan Wojtowytsch. On the convergence of gradient descent training for two-layer ReLU-networks in the mean field regime. *arXiv preprint arXiv:2005.13530*, 2020.

[121] Stephan Wojtowytsch and Weinan E. Can shallow neural networks beat the curse of dimensionality? A mean field training perspective. *IEEE Transactions on Artificial Intelligence*, 1(2):121–129, 2021.

[122] Yahong Yang, Haizhao Yang, and Yang Xiang. Nearly optimal VC-dimension and pseudo-dimension bounds for deep neural network derivatives. *Advances in Neural Information Processing Systems*, 36:21721–21756, 2023.

[123] Yahong Yang, Yue Wu, Haizhao Yang, and Yang Xiang. Nearly optimal approximation rates for deep super ReLU networks on Sobolev spaces. *arXiv preprint arXiv:2310.10766*, 2023.

[124] Yunfei Yang and Ding-Xuan Zhou. Optimal rates of approximation by shallow ReLU^k neural networks and applications to nonparametric regression. *Constructive Approximation*, 62(2):329–360, 2025.

[125] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

[126] Can Yaras, Peng Wang, Zhihui Zhu, Laura Balzano, and Qing Qu. Neural collapse with normalized features: A geometric analysis over the riemannian manifold. *Advances in Neural Information Processing Systems*, 35:11547–11560, 2022.

[127] Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

[128] Dmitry Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. *Conference on Learning Theory*, 639–649, 2018.

[129] Dmitry Yarotsky and Anton Zhevnerchuk. The phase diagram of approximation rates for deep neural networks. *Advances in Neural Information Processing Systems*, 33:13005–13015, 2020.

[130] Penghang Yin, Shuai Zhang, Jiancheng Lyu, Stanley Osher, Yingyong Qi, and Jack Xin. BinaryRelax: A relaxation approach for training deep neural networks with quantized weights. *SIAM Journal on Imaging Sciences*, 11(4):2205–2223, 2018.

[131] Lijia Yu, Xiao-Shan Gao, Lijun Zhang, and Yibo Miao. Generalizablity of memorization neural network. *Advances in Neural Information Processing Systems*, 37:113311–113359, 2024.

[132] Jinjie Zhang, Harish Kannan, Alexander Cloninger, and Rayan Saab. Sigma-Delta and distributed noise-shaping quantization methods for random Fourier features. *Information and Inference: A Journal of the IMA*, 13(1):iaad052, 2024.

[133] Jinjie Zhang and Rayan Saab. Spfq: A stochastic algorithm and its error analysis for neural network quantization. *arXiv preprint arXiv:2309.10975*, 2023.

[134] Jinjie Zhang, Yixuan Zhou, and Rayan Saab. Post-training quantization for neural networks with provable guarantees. *SIAM Journal on Mathematics of Data Science*, 5(2):373–399, 2023.

[135] Mo Zhou, Rong Ge, and Chi Jin. A local convergence theory for mildly over-parameterized two-layer neural network. *Conference on Learning Theory*, 4577–4632, 2021.

[136] Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. *Advances in Neural Information Processing Systems*, 34:29820–29834, 2021.

[137] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep ReLU networks. *Machine Learning*, 109(3):467–492, 2020.

[138] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.