Journal of Fourier Analysis and Applications (2025) 31:49
https://doi.org/10.1007/s00041-025-10182-7

Frame Quantization of Neural Networks

Wojciech Czaja[1] · **Sanghoon Na**[1]

Received: 14 April 2024 / Revised: 30 May 2025 / Accepted: 2 June 2025 /
Published online: 25 June 2025

Abstract
We present a post-training quantization algorithm with error estimates relying on ideas originating from frame theory. Specifically, we use first-order Sigma-Delta ($\Sigma\Delta$) quantization for finite unit-norm tight frames to quantize weight matrices and biases in a neural network. In our scenario, we derive an error bound between the original neural network and the quantized neural network in terms of step size and the number of frame elements. We also demonstrate how to leverage the redundancy of frames to achieve a quantized neural network with higher accuracy.

Keywords Neural Network Quantization · Post-Training Quantization · Sigma-Delta Quantization · Finite Frames

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, pulse-code modulation (PCM) and sigma-delta ($\Sigma\Delta$) quantization. Among them, $\Sigma\Delta$ algorithms stand out due to their theoretically guaranteed robustness. Mathematical foundations were developed in several seminal works [4–6, 12, 17], and have been carefully studied since, e.g., [20, 21, 26, 35].

On a different level, the notion of quantization also led to the development of quantum mechanics from the classical understanding of the physical phenomena, through ideas such as Weyl quantization, e.g., [3, 15]. This is a different, but not entirely unrelated, quantization concept, which has been brought closer to the analog-

Dedicated to Professor Karlheinz Gröchenig on the occasion of his 65th birthday.

Communicated by Gabriel Peyre.

✉ Sanghoon Na
shna2020@umd.edu

[1] Department of Mathematics, University of Maryland, College Park, MD 20742, USA

to-digital quantization in the work of Karlheinz Gröchenig and his collaborators via the use of localization operators arising in the time-frequency analysis [2, 10]. Gröchenig then extended the concept of localization to frame theory [14], laying a foundation for further joint exploration of the theory of quantization and the frame theory [15].

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 [13]. 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 [11, 24, 27, 28, 34, 39] 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 [1, 22, 29–32, 37, 38, 40, 41] 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 [13] and references therein.

Despite the popularity of PTQ methods, unfortunately, most of them lack theoretical error analysis. While some algorithms [29, 30, 40, 41] 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 [23], 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 [12, 17] focus on $\Sigma\Delta$ quantization for bandlimited functions, [4–6] 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 [4–6], we are able to provide a new PTQ algorithm with error estimates. Our contributions are threefold:

1. In Section 4 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 [42] 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 5. These results demonstrate how to control the error using the step size and the number of frame elements.
3. Finally, in Section 6, we present numerical results for quantizing neural networks using 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 5. Experiments and detailed explanations for the most extreme case, 1-bit quantization, are also provided.

2 Preliminaries

In this paper, all vectors are column vectors in $\mathbb{R}^d$, $d \in \mathbb{N}$. For a vector x, $\|x\|$ denotes the Euclidean norm of x. For a matrix A, $\|A\|$ denotes the matrix 2-norm of A, which is the largest singular value of A.

We introduce the mathematical formulation of neural networks used in this paper. 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

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

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 σ 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 $x \in \mathbb{R}$. Note that ReLU is a 1-Lipchitz continuous function attaining 0 at $x = 0$. Throughout this paper we shall assume activation function σ is an L-Lipschitz continuous function with $\sigma(0) = 0$, which makes ReLU a special case. In addition, we will also assume that $m_i \geq 3$ for all i's. This is a natural assumption, since typical neural network architectures set the number of neurons at each layer to be greater than 2.

Next, we introduce the mathematical formulation for residual blocks and residual neural networks used in this paper. We adopt a similar mathematical formulation from Chapter 3.8.2 in [16]. A *residual neural network* with n residual blocks is a function $g : \mathbb{R}^k \to \mathbb{R}^k$ defined as

$$g(x) = z^{[n]} \circ \sigma \circ z^{[n-1]} \circ \cdots \circ \sigma \circ z^{[1]}(x), \tag{2}$$

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.

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 (1), 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,

$$f_Q(x) = h_Q^{[n]} \circ \sigma \circ h_Q^{[n-1]} \circ \sigma \cdots \circ \sigma \circ h_Q^{[1]}(x), \tag{3}$$

where $h_Q^{[i]}(x) = Q_i x + c_i$ for $i = 1, \cdots, n$. Similarly for a residual network $g(x)$ in (2),

$$g_Q(x) = z_Q^{[n]} \circ \sigma \circ z_Q^{[n-1]} \circ \cdots \circ \sigma \circ z_Q^{[1]}(x), \tag{4}$$

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.

Note that if we write $\widetilde{W}_l = (W_l, b_l)$ and $\tilde{x} = (x^T, 1)^T$, 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 (1) can be regarded as $h^{[i]}(x) = W_i x$ and $z^{[i]}(x)$'s in (2) can be regarded as $z^{[i]}(x) = W_{i,2}\sigma(W_{i,1}x) + x$.

Next, we recall some basics from frame theory. In this paper, we only discuss finite frames for $\mathbb{R}^d$. For a discussion of finite frames for general Hilbert spaces, see [9]. We say a set $\{e_1, \cdots, e_N\}$ in $\mathbb{R}^d$ is a *finite frame* for $\mathbb{R}^d$ if there exist $0 < A \le B < \infty$, such that $A\|x\|^2 \le \sum_{i=1}^N |\langle x, e_i\rangle|^2 \le B\|x\|^2$ holds for all $x \in \mathbb{R}^d$. 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). For a finite frame $\{e_1, \cdots, e_N\}$ in $\mathbb{R}^d$, the linear function $S : \mathbb{R}^d \to \mathbb{R}^d$ defined by

$$Sx = \sum_{i=1}^N \langle x, e_i\rangle e_i, \tag{5}$$

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 $AI_d \le S \le BI_d$, where I_d is the identity matrix of $\mathbb{R}^d$. Therefore, its inverse S^{-1} exists and it satisfies $\frac{1}{B}I_d \le S^{-1} \le \frac{1}{A}I_d$. This inverse operator S^{-1} is called the *dual frame operator*. Multiplying (5) by S^{-1} gives us an atomic decomposition of an arbitrary $x \in \mathbb{R}^d$:

$$x = S^{-1}Sx = S^{-1}\Big\{\sum_{i=1}^{N}\langle x, e_i\rangle e_i\Big\} = \sum_{i=1}^{N}\langle x, e_i\rangle S^{-1}e_i. \tag{6}$$

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 [9]. 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, (6) gives us the frame expansion $x = \frac{1}{A}\sum_{i=1}^{N}\langle x, e_i\rangle e_i$, for any $x \in \mathbb{R}^d$. Moreover, if F is a FUNTF, then it is known that $A = N/d$ holds [6]. Therefore, we have the frame expansion $x = \frac{d}{N}\sum_{i=1}^{N}\langle x, e_i\rangle e_i$ for any $x \in \mathbb{R}^d$.

3 Frame Quantization

In vector quantization, first-order $\Sigma\Delta$ quantization algorithm has a uniform upper bound on the error in the case of finite frames [5] and it is known to outperform PCM most of the time for realistic settings [7]. Since a matrix can be interpreted as a stack of column vectors, $\Sigma\Delta$ becomes an option for quantizing a weight matrix. In this section, we introduce first-order $\Sigma\Delta$ quantization algorithms for finite frames for $\mathbb{R}^d$. We begin with some common definitions in [4–6, 9, 41]. Given $K \in \mathbb{N}$ and $\delta > 0$, the *midrise quantization alphabet* A_K^δ is defined as

$$A_K^\delta = \Big\{(-K + \frac{1}{2})\delta, (-K + \frac{3}{2})\delta, \cdots, -\frac{1}{2}\delta, \frac{1}{2}\delta, \cdots, (K - \frac{1}{2})\delta\Big\}. \tag{7}$$

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 which is closest to u.

Definition 3.1 [5] Given $K \in \mathbb{N}$ and $\delta > 0$, define the midrise quantization alphabet A_K^δ and the $2K$-level midrise uniform scalar quantizer Q with stepsize δ . 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 $\{1, 2, \cdots, N\}$. For a given input sequence $\{x_1, x_2, \cdots, x_N\}$, the associated first-order $\Sigma\Delta$ *quantization* is defined by the iteration:

$$\begin{aligned} u_n &= u_{n-1} + x_{p(n)} - q_n, \\ q_n &= Q(u_{n-1} + x_{p(n)}), \end{aligned}$$

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.

In the above 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 [5].

We now describe the vector quantization process. Let $F = \{e_1, \cdots, e_N\}$ be a finite frame for $\mathbb{R}^d$ and let p be a permutation of $\{1, 2, \cdots, N\}$. Choose an arbitrary vector $x \in \mathbb{R}^d$ and represent it as $x = \sum_{i=1}^N x_i S^{-1} e_i$ with frame expansion in (6). We say $\bar{x} = \sum_{i=1}^N q_i S^{-1} e_{p(i)}$ is the *quantized expansion* of x, where $\{q_1, \cdots, q_N\}$ is the quantized sequence from the first-order $\Sigma\Delta$ quantization in Definition 3.1. The first-order $\Sigma\Delta$ scheme in Definition 3.1 is an iterative scheme that depends heavily on the choice of permutation p. 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 $\sigma(F, p) := \sum_{i=1}^{N-1} \|e_{p(i)} - e_{p(i+1)}\|$. Frame variation is a measurement that captures the interdependencies between the frame elements resulting from the choice of p. It is an important quantity that reflects the role of permutation p in the error estimates for first-order $\Sigma\Delta$ quantization.

We provide 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 [5] and [6].

Theorem 3.1 [5] *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*

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

To obtain a quantized expansion with a small error, it is desirable to choose a permutation p that makes frame variation $\sigma(F, p)$ small. In [36], the author proved that for a set $\{e_1, \cdots, e_N\} \subset [-\frac{1}{2}, \frac{1}{2}]^d$, with $d \geq 3$, there exists a permutation p of $\{1, 2, \cdots, N\}$ which satisfies $\sum_{i=1}^{N-1} \|e_{p(i)} - e_{p(i+1)}\| \leq 2\sqrt{d+3}N^{1-\frac{1}{d}} - 2\sqrt{d+3}$. 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 [6], the authors combined these results to obtain the following result.

Theorem 3.2 [6] *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*

$$\sigma(F, p) \leq 4\sqrt{d+3}N^{1-\frac{1}{d}} - 4\sqrt{d+3}. \tag{8}$$

Next, we estimate the approximation error in terms of δ, d, and N.

Corollary 3.3 *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 (8). Let $\bar{x}$ be the quantized expansion. Then, the approximation error satisfies*

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

Proof Apply Theorem 3.2 to Theorem 3.1. □

Corollary 3.3 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 5.

While Theorem 3.2 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 for identity permutation.

Definition 3.2 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. We omit the proof.

Lemma 3.4 [5] *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 FUNTFs with uniformly bounded frame variations, it is straightforward to check that the approximation error in Corollary 8 can be improved to $\|x - \bar{x}\| \leq C_d \delta / N$. For more details on finite frames with uniformly bounded frame variation, see [8].

4 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

$$\max_{j=1,\cdots,m_{i-1}} \|w_j^i\| \leq (K_i - \frac{1}{2})\delta_i. \tag{9}$$

Next, we find a permutation p_i of $\{1, \cdots, N_i\}$ that satisfies (8). The algorithm to find such permutations is described in [6, 36], 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

$$q_j^i = \sum_{k=1}^{N_i} q_{j,k}^i S^{-1} e_{p_i(k)}^i = \frac{m_i}{N_i} \sum_{k=1}^{N_i} q_{j,k}^i e_{p_i(k)}^i. \tag{10}$$

Note that the first-order $\Sigma\Delta$ quantization in Definition 3.1 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 $C_i = [q_{j,k}^i]_{1 \le j \le m_{i-1}, 1 \le k \le N_i} \in (A_{K_i}^{\delta_i})^{m_{i-1} \times N_i}$. 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 (10).

Current neural network quantization methods convert a weight matrix $W = [w_{ij}]_{1 \le i \le m, 1 \le j \le n} \in \mathbb{R}^{m \times n}$ into a quantized matrix $Q = [q_{ij}]_{1 \le i \le m, 1 \le j \le n}$, 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 $\max_{j=1,\cdots,m_{i-1}} \|w_j^i\| \le (K_i - \frac{1}{2})\delta_i$.
2. Find a permutation p_i of $\{1, \cdots, N_i\}$ that satisfies (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 3.1 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 $C_i = [q_{j,k}^i]_{1 \le j \le m_{i-1}, 1 \le k \le N_i} \in (A_{K_i}^{\delta_i})^{m_{i-1} \times N_i}$.

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^T. This framework is applied for neural networks with residual blocks in Section 5.2.

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.

5 Error Estimates

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 5.1 *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,*

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

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 (8), Corollary 3.3 gives us the error estimate

$$\|w_j^i - q_j^i\| \le \frac{\delta_i m_i}{2N_i}(4\sqrt{m_i + 3}N_i^{1-\frac{1}{m_i}} - 4\sqrt{m_i + 3} + 1) \tag{11}$$

for all $j = 1, \cdots, m_{i-1}$. Now, using (11) and the assumption $m_i \ge 3$ from Section 2, we have

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

□

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 5.2 *Let Q_i be the quantized matrix of W_i. Let $\sigma_i = \|W_i\|$. Then we have*

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

Proof Using Lemma 5.1 and the triangle inequality of matrix norms, we have

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

□

5.1 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 (1) and $f_Q(x)$ its quantized neural network (3). For convenience, we adopt the assumptions from Section 2 and omit the biases in our analysis. Thus, we can write $f(x) = W_n(\sigma(\cdots\sigma(W_1x)\cdots))$ and $f_Q(x) = Q_n(\sigma(\cdots\sigma(Q_1x)\cdots))$. As in Lemma 5.2, we let $\sigma_i = \|W_i\|$ for $i = 1, \cdots, n$.

Theorem 5.3 *Let $f(x) = W_n(\sigma(\cdots\sigma(W_1x)\cdots))$ be a feed-forward neural network with n layers. Let $f_Q(x) = Q_n(\sigma(\cdots\sigma(Q_1x)\cdots))$ be the quantized neural network obtained by means of Algorithm 1. Then, for any input $X \in \mathbb{R}^{m_0}$, we have*

$$\begin{aligned}\|f(X) - f_Q(X)\| \leq L^{n-1}\|X\| \times \sum_{j=1}^{n} \Big\{ & 2\sqrt{2}\delta_j m_j \sqrt{m_j m_{j-1}} N_j^{-\frac{1}{m_j}} \times \prod_{i=j+1}^{n} \sigma_i \\ & \times \prod_{l=1}^{j-1} (2\sqrt{2}\delta_l m_l \sqrt{m_l m_{l-1}} N_l^{-\frac{1}{m_l}} + \sigma_l) \Big\}. \end{aligned} \tag{12}$$

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

$$\begin{aligned} & \|W_n\sigma \cdots W_j\sigma Q_{j-1}\sigma Q_{j-2} \cdots Q_1X - W_n\sigma \cdots W_{j+1}\sigma Q_j\sigma Q_{j-1}\sigma Q_{j-2} \cdots Q_1X\| \\ & \leq L^{n-j}\|W_n\| \cdots \|W_{j+1}\| \times \|W_j - Q_j\| \|\sigma Q_{j-1}\sigma Q_{j-2} \cdots Q_1X\| \\ & \leq L^{n-1}\|W_n\| \cdots \|W_{j+1}\| \times \|W_j - Q_j\| \times \|Q_{j-1}\| \cdots \|Q_1\| \|X\|. \end{aligned}$$

Next, using Lemma 5.1, Lemma 5.2, and triangle inequality, we get

$$\begin{aligned}
&\|f(X) - f_Q(X)\| = \|W_n(\sigma(\cdots\sigma(W_1x)\cdots)) - Q_n(\sigma(\cdots\sigma(Q_1x)\cdots))\| \\
&\le \sum_{j=1}^{n} \|W_n\sigma\cdots W_j\sigma Q_{j-1}\sigma Q_{j-2}\cdots Q_1X - W_n\sigma\cdots \\
&W_{j+1}\sigma Q_j\sigma Q_{j-1}\sigma Q_{j-2}\cdots Q_1X\| \\
&\le \sum_{j=1}^{n} L^{n-1}\|W_n\|\cdots\|W_{j+1}\| \times \|W_j - Q_j\| \times \|Q_{j-1}\|\cdots\|Q_1\|\|X\| \\
&\le L^{n-1}\|X\| \times \sum_{j=1}^{n} \left\{ 2\sqrt{2}\delta_j m_j \sqrt{m_j m_{j-1}} N_j^{-\frac{1}{m_j}} \times \prod_{i=j+1}^{n} \sigma_i \right. \\
&\left. \times \prod_{l=1}^{j-1} (2\sqrt{2}\delta_l m_l \sqrt{m_l m_{l-1}} N_l^{-\frac{1}{m_l}} + \sigma_l) \right\}. \qquad (13)
\end{aligned}$$

□

Note that $\{\sigma_i\}_{i=1}^n$, $\{m_i\}_{i=0}^n$, and L are constants determined by the given FNN $f(x)$. Therefore, the upper bound in (12) 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 (12) 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 (12) 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 5.3.

Corollary 5.4 *Let $f(x) = W_n(\sigma(\cdots\sigma(W_1x)\cdots))$ be a feed-forward neural network with n layer 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 (8). Now, use Algorithm 1 to quantize $W_1, \cdots, W_{n-1}$ and W_n^T 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*

$$\begin{aligned}
&\|f(X) - f_Q(X)\| \\
&\le 2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} L^{n-1}\|X\| \sum_{j=1}^{n} \left\{ \prod_{i=j+1}^{n} \sigma_i \prod_{l=1}^{j-1} (2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} + \sigma_l) \right\} \qquad (14)
\end{aligned}$$

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

Proof By Lemma 5.1, 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 $\|W_n - Q_n\| =$

$\|W_n^T - Q_n^T\| \leq 2\sqrt{2}\delta_n m_{n-1}\sqrt{m_{n-1}m_n}N_n^{-\frac{1}{m_{n-1}}} \leq 2\sqrt{2}\delta M^2 N^{-\frac{1}{m}}$. These estimates yield $\|Q_i\| \leq \|W_i - Q_i\| + \|W_i\| \leq 2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} + \sigma_i$. Therefore, from the proof of Theorem 5.3, we get

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

□

If we use a FUNTF for $\mathbb{R}^m$ with N elements where $N \geq \left(\frac{2\sqrt{2}\delta M^2}{\min\{\sigma_1,\cdots,\sigma_n\}}\right)^m$, then we have the following simplification.

Corollary 5.5 *If we have* $N \geq \left(\frac{2\sqrt{2}\delta M^2}{\min\{\sigma_1,\cdots,\sigma_n\}}\right)^m$ *in Corollary 5.4, then*

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

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

Proof $N \geq \left(\frac{2\sqrt{2}\delta M^2}{\min\{\sigma_1,\cdots,\sigma_n\}}\right)^m$ implies $2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} \leq \min\{\sigma_1,\cdots,\sigma_n\}$. Hence we have

$$\prod_{i=j+1}^{n}\sigma_i\prod_{l=1}^{j-1}(2\sqrt{2}\delta M^2 N^{-\frac{1}{m}} + \sigma_l) \leq \prod_{i=j+1}^{n}\sigma_i\prod_{l=1}^{j-1}(\sigma_l + \sigma_l) = \frac{2^{j-1}}{\sigma_j}\prod_{i=1}^{n}\sigma_i.$$

Applying this to (14) yields the result. □

Note that the above results apply to general FUNTFs. However, certain families of FUNTFs discussed in Section 3, 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 5.1:

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

The last inequality holds since we assume $m_i \geq 3$ for all i's. Using this, we can show (12) can be rewritten as an explicit upper bound for harmonic frames, which is a stronger upper bound in terms of N_i's:

$$\|f(X) - f_Q(X)\| \leq L^{n-1}\|X\| \times \sum_{j=1}^{n} \Big\{ \frac{(8\pi + \sqrt{3})}{6\sqrt{3}} \delta_j m_j \sqrt{m_j m_{j-1}} N_j^{-1} \times \prod_{i=j+1}^{n} \sigma_i \times \prod_{l=1}^{j-1} \Big(\frac{(8\pi + \sqrt{3})}{6\sqrt{3}} \delta_l m_l \sqrt{m_l m_{l-1}} N_l^{-1} + \sigma_l \Big) \Big\}. \tag{15}$$

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.

5.2 Neural Networks with Residual Blocks

We now establish error estimates for residual networks. Following the assumptions in Section 2, we 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 (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 $W_{i,1}^T$ 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 5.6 *Let $g(x)$ be a residual neural network with n residual blocks (2). 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 $g(x)$ (4) from Algorithm 1. Then for any input $X \in \mathbb{R}^k$, we have

$$\|g(X) - g_Q(X)\| \leq 4\delta k\sqrt{r(k+3)}(\delta k\sqrt{r(k+3)} + \lambda)N^{-\frac{1}{k}}\|X\| \times \sum_{j=0}^{n-1} (\lambda^2 + 1)^j \Big((2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}} + \lambda)^2 + 1 \Big)^{n-1-j}. \tag{16}$$

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

$$\|y_i(X) - y_{i,Q}(X)\| \le A\|X\| \sum_{j=0}^{i-1} (\lambda^2+1)^j B^{i-1-j}, \tag{17}$$

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

$$y_i(X) = W_{i,2}(\sigma(W_{i,1}(\sigma(y_{i-1}(X))))) + \sigma(y_{i-1}(X)), \tag{18}$$
$$y_{i,Q}(X) = Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1,Q}(X))))) + \sigma(y_{i-1,Q}(X)). \tag{19}$$

By using the triangle inequality repeatedly, we obtain:

$$\begin{aligned}
&\|y_i(X) - y_{i,Q}(X)\| \\
&\quad \le \|W_{i,2}(\sigma(W_{i,1}(\sigma(y_{i-1}(X))))) - Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1,Q}(X)))))\| \\
&\qquad + \|\sigma(y_{i-1}(X)) - \sigma(y_{i-1,Q}(X))\| \\
&\quad \le \|W_{i,2}(\sigma(W_{i,1}(\sigma(y_{i-1}(X))))) - W_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X)))))\| \\
&\qquad + \|W_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X))))) - Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X)))))\| \\
&\qquad + \|Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1}(X))))) - Q_{i,2}(\sigma(Q_{i,1}(\sigma(y_{i-1,Q}(X)))))\| \\
&\qquad + \|\sigma(y_{i-1}(X)) - \sigma(y_{i-1,Q}(X))\|.
\end{aligned} \tag{20}$$

Since σ is ReLU, we have $\|\sigma(x) - \sigma(y)\| \le \|x - y\|$ and $\|\sigma(x)\| \le \|x\|$ for any vectors x, $y \in \mathbb{R}^k$. Using this fact, the first term in (20) is bounded by:

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

The second term in (20) is bounded by:

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

The third term in (20) is bounded by:

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

Finally, the fourth term in (20) is bounded by:

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

From the four inequalities above, we obtain:

$$\|y_i(X) - y_{i,Q}(X)\| \le \{\lambda\|W_{i,1} - Q_{i,1}\| + \|W_{i,2} - Q_{i,2}\|\|Q_{i,1}\|\} \times \|y_{i-1}(X)\|$$

$$+ \{\|Q_{i,2}\|\|Q_{i,1}\| + 1\} \times \|y_{i-1}(X) - y_{i-1,Q}(X)\|. \tag{21}$$

From the proof of Lemma 5.1 and Lemma 5.2, using $\|A\| = \|A^T\|$ 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 (21), we have:

$$\begin{aligned}\|y_i(X) - y_{i,Q}(X)\| &\leq 2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}}(\lambda + 2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}} + \lambda)\|y_{i-1}(X)\| \\ &\quad + \left((2\delta k\sqrt{r(k+3)}N^{-\frac{1}{k}} + \lambda)^2 + 1\right)\|y_{i-1}(X) - y_{i-1,Q}(X)\| \\ &= A\|y_{i-1}(X)\| + B\|y_{i-1}(X) - y_{i-1,Q}(X)\|. \end{aligned} \tag{22}$$

From the recurrence relations (18) and (19), one can easily check that

$$\begin{aligned}\|y_i(X)\| &= \|W_{i,2}(\sigma(W_{i,1}(\sigma(y_{i-1}(X))))) + \sigma(y_{i-1}(X))\| \\ &\leq \|W_{i,2}\|\|W_{i,1}\|\|\|y_{i-1}(X)\| + \|y_{i-1}(X)\| \leq (\lambda^2+1)\|y_{i-1}(X)\|. \end{aligned} \tag{23}$$

Using (23) repeatedly, we obtain $\|y_i(X)\| \leq (\lambda^2+1)^i\|y_0(X)\| = (\lambda^2+1)^i\|X\|$, which together with (22) implies:

$$\begin{aligned}\|y_i(X) - y_{i,Q}(X)\| &\leq A\|y_{i-1}(X)\| + B\|y_{i-1}(X) - y_{i-1,Q}(X)\| \\ &\leq A(\lambda^2+1)^{i-1}\|X\| + B\|y_{i-1}(X) - y_{i-1,Q}(X)\|. \end{aligned} \tag{24}$$

To prove (17), we proceed by induction. When $i = 1$, (24) 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, (17) holds when $i = 1$. Now, let's assume that (17) holds for $i = t$. When $i = t+1$, by (24) and 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, (17) holds. Note that $g(x) = y_n(x)$ and $g_Q(x) = y_{n,Q}(x)$. Taking $i = n$ in (17) 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 [16, 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 5.7 *Let $g(x)$ be a residual neural network with n residual blocks (2) 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 $g(x)$ (4) from Algorithm 1. Then for any input $X \in \mathbb{R}^k$, we have

$$\begin{aligned} \|g(X) - g_Q(X)\| \leq\ & 4\delta k\sqrt{k(k+3)}(\delta k\sqrt{k(k+3)} + \lambda)N^{-\frac{1}{k}}\|X\| \\ & \times \sum_{j=0}^{n-1}(\lambda^2+1)^j\Big((2\delta k\sqrt{k(k+3)}N^{-\frac{1}{k}} + \lambda)^2 + 1\Big)^{n-1-j}. \end{aligned} \tag{25}$$

Theorem 5.6 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 (16) as $O(N^{-1/k})\|X\|$, similar to the interpretation in Section 5.1. This result applies to general FUNTFs. However, for FUNTFs with uniformly bounded variation, the terms $N^{-\frac{1}{k}}$ in (16) can be replaced by N^{-1}, further reducing the reconstruction error.

6 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 5. 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.

6.1 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 denotes 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 (1) without bias terms. The MNIST images were normalized by dividing each pixel value by 255. Training was carried

Table 1 Frame Quantization for FNN with 3 layers. The test accuracy for the pretrained neural network is $97.72 \pm 0.21\%$.

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	$97.68 \pm 0.25\%$	$97.57 \pm 0.25\%$	$97.20 \pm 0.30\%$	$95.71 \pm 1.36\%$	$86.97 \pm 1.92\%$

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 1. As expected, quantizing with smaller δ and larger N generally led to higher test accuracy.

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 (15) 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 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 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.

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

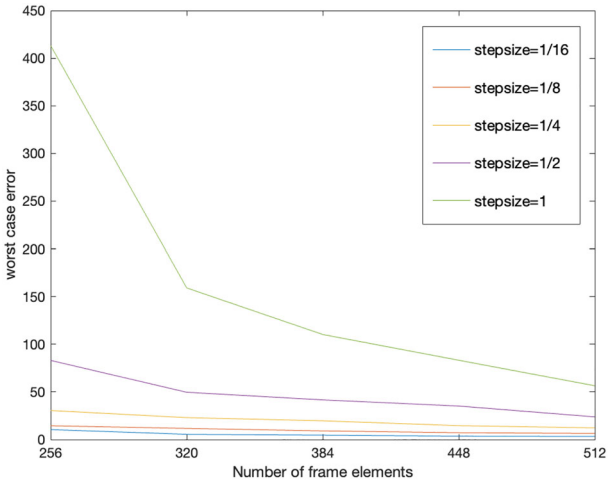

Fig. 1 Worst-case error $\|f(X) - f_Q(X)\|$ for FNN with 3 layers.

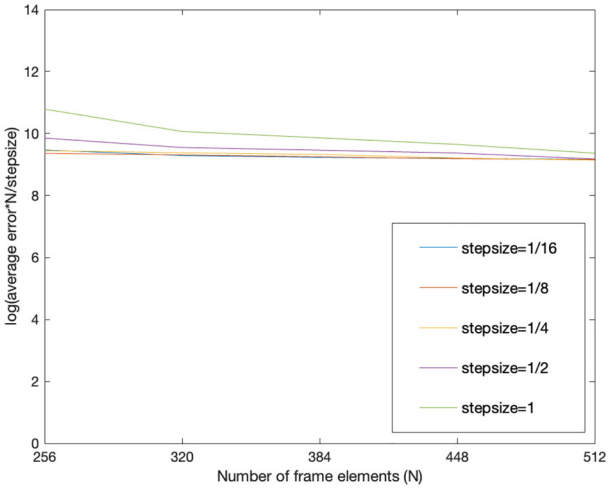

Fig. 2 $\log \mathrm{E}_X[\|f(X) - f_Q(X)\| \times N/\delta]$ for FNN with 3 layers

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 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 5.

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

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 1-Bit Frame Quantization for FNN with 3 layers

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

6.3 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, [11] quantized the weights of the neural networks to $\{-1, 1\}$. Mathematical theories were also developed in 1-bit $\Sigma\Delta$ quantization for bandlimited signals [18] and approximation capabilities of 1-bit neural networks [19].

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 (7). 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 (9) 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 and Table 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.

Table 4 1-Bit Frame Quantization for Neural Network with 2 Residual Blocks

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 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 $c \in \{128, 256\}$.

	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%

6.4 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 [33], 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) [29], 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 [25]. The test accuracy results for both Frame Quantization and GPFQ are presented in Table 5. The pretrained ResNet-18 model achieved a baseline test accuracy of 90.46% prior to quantization.

Table 6 1-Bit Frame Quantization for ResNet-18 with CIFAR-10 dataset

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

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 6.

The numerical results in Table 5 and Table 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 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.

7 Conclusions and Future Work

In this paper, 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.

1. In this paper, we only focus on using FUNTFs for $\mathbb{R}^d$. Designing a quantization method using different types of frames would be a new task for the future.
2. Our method uses first-order $\Sigma\Delta$ quantization, but one may consider using higher-order $\Sigma\Delta$ quantization. In this case, how much can we benefit from the error bound in terms of the number of frame elements? Also, what quantization rule should be used for higher-order schemes?
3. To the best of our knowledge, this is the first attempt to adopt frame theory to neural network quantization. We were able to explain the accuracy and storage benefits,

but we must admit that we do not immediately see a connection between using frames and saving inference time. To develop a new frame quantization algorithm that reduces inference time, we may need to use some specific frame.

4. Our method considers a uniformly spaced symmetric quantization alphabet. Alternatively, one can consider a nonuniform asymmetric quantization alphabet, whose design would depend on the distributions of both the dataset and the pretrained weight matrices.
5. While our method quantizes without data, which benefits in preserving privacy, other PTQ methods commonly use a small batch of data to improve accuracy. Therefore, developing a PTQ method based on data-adapted finite frames could be an interesting direction.

References

1. Banner, R., Nahshan, Y., Soudry, D.: Post training 4-bit quantization of convolutional networks for rapid-deployment. Adv. Neural Inf. Process. Syst. 32 (2019)
2. Bayer, D., Gröchenig, K.: Time–frequency localization operators and a Berezin transform. Integral Equ. Oper. Theory **82**, 95–117 (2015)
3. Bayer, D., Cordero, E., Gröchenig, K., Trapasso, S.I.: Linear perturbations of the Wigner transform and the Weyl quantization. Advances in Microlocal and Time-Frequency Analysis, 79-120 (2020)
4. Benedetto, J.J., Powell, A.M., Yılmaz, Ö.: Second-order Sigma-Delta ($\Sigma\Delta$) quantization of finite frame expansions. Appl. Comput. Harmon. Anal. **20**(1), 126–148 (2006)
5. Benedetto, J.J., Powell, A.M., Yılmaz, Ö.: Sigma-delta ($\Sigma\Delta$) quantization and finite frames. IEEE Trans. Inf. Theory **52**(5), 1990–2005 (2006)
6. Benedetto, J.J., Oktay, O., Tangboondouangjit, A.: Complex Sigma-Delta quantization algorithms for finite frames. Radon transforms, geometry, and wavelets **464**, 27–49 (2008)
7. Benedetto, J.J., Oktay, O.: Pointwise comparison of PCM and $\Sigma\Delta$ quantization. Constr. Approx. **32**, 131–158 (2010)
8. Bodmann, B.G., Paulsen, V.I.: Frame paths and error bounds for sigma-delta quantization. Appl. Comput. Harmon. Anal. **22**(2), 176–197 (2007)
9. Casazza, P.G., Kutyniok, G.: Finite frames: Theory and applications. Springer Science & Business Media (2012)
10. Cordero, E., Gröchenig, K.: Time–frequency analysis of localization operators. J. Funct. Anal. **205**(1), 107–131 (2003)
11. Courbariaux, M., Bengio, Y., David, Y-P.: Binaryconnect: Training deep neural networks with binary weights during propagations. Adv. Neural. Inf. Process. Syst. 28 (2015)
12. Daubechies, I., DeVore, R.: Approximating a bandlimited function using very coarsely quantized data: A family of stable sigma-delta modulators of arbitrary order. Ann. Math. **158**(2), 679–710 (2003)
13. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. Low-Power Computer Vision. Chapman and Hall/CRC, 291-326 (2022)

14. Gröchenig, K.: Localization of frames, Banach frames, and the invertibility of the frame operator. J. Fourier Anal. Appl. **10**, 105–132 (2004)
15. Gröchenig, K.: Foundations of time-frequency analysis. Springer Science & Business Media (2013)
16. Grohs, P., Kutyniok, G.: Mathematical aspects of deep learning. Cambridge University Press (2022)
17. Güntürk, C.S.: Approximating a bandlimited function using very coarsely quantized data: improved error estimates in sigma-delta modulation. J. Am. Math. Soc. **17**(1), 229–242 (2004)
18. Güntürk, C.S.: One-bit sigma-delta quantization with exponential accuracy. Commun. Pure Appl. Math. **56**(11), 1608–1630 (2003)
19. Güntürk, C. S., Li, W.: Approximation of functions with one-bit neural networks. arXiv preprint arXiv:2112.09181 (2021)
20. Güntürk, C. S., Lammers, M., Powell, A., Saab, R., Yılmaz, ö.: Sigma delta quantization for compressed sensing. 2010 44th Annual Conference on Information Sciences and Systems (CISS). IEEE. (2010)
21. Güntürk, C.S., Lammers, M., Powell, A., Saab, R., Yılmaz, Ö.: Sobolev duals for random frames and $\Sigma\Delta$ quantization of compressed sensing measurements. Found. Comput. Math. **13**, 1–36 (2013)
22. Guo, C., et al.: SQuant: On-the-fly data-free quantization via diagonal hessian approximation. arXiv preprint arXiv:2202.07471 (2022)
23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 770-778 (2016)
24. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. Adv. Neural Inf. Process. Syst. 29 (2016)
25. Hubara, I., Nahshan, Y., Hanani, Y., Banner, R., Soudry, D. Accurate post training quantization with small calibration sets. Int. Conf. Mach. Learn. PMLR, 4466-4475 (2021)
26. Krahmer, F., Saab, R., Yılmaz, Ö.: 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)
27. Kummer, L., Sidak, K., Reichmann, T., Gansterer, W.: Adaptive Precision Training (AdaPT): A dynamic quantized training approach for DNNs. Proc. SIAM Int. Conf. Data Mining, 559-567 (2023)
28. Long, Z., Yin, P., Xin, J.: Recurrence of optimum for training weight and activation quantized networks. Appl. Comput. Harmon. Anal. **62**, 41–65 (2023)
29. Lybrand, E., Saab, R.: A greedy algorithm for quantizing neural networks. J. Mach. Learn. Res. **22**(1), 7007–7044 (2021)
30. Maly, J., Saab, R.: A simple approach for quantizing neural networks. Appl. Comput. Harmon. Anal. **66**, 138–150 (2023)
31. Nagel, M., Amjad, R. A., Van Baalen, M., Louizos, C., Blankevoort, T.: Up or down? adaptive rounding for post-training quantization. Proc. 37th Int. Conf. Mach. Learn. PMLR, 7197-7206 (2020)
32. Nahshan, Y., Chmiel, B., Baskin, C., Zheltonozhskii, E., Banner, R., Bronstein, A.M., Mendelson, A.: Loss aware post-training quantization. Mach. Learn. **110**(11), 3245–3262 (2021)
33. Paszke, A., et al.: Pytorch: An imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. **32**, 8026–8037 (2019)
34. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks, pp. 525–542. Eur. Conf. Comput. Vis, Springer (2016)
35. Saab, R., Wang, R., Yılmaz, Ö.: Quantization of compressive samples with stable and robust recovery. Appl. Comput. Harmon. Anal. **44**(1), 123–143 (2018)
36. Wang, Y.: Sigma-Delta quantization errors and the traveling salesman problem. Adv. Comput. Math. **28**, 101–118 (2008)
37. Wei, X., Gong, R., Li, Y., Liu, X., Yu, F.: Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. arXiv preprint arXiv:2203.05740 (2022)
38. Yao, Z., Yazdani Aminabadi, R., Zhang, M., Wu, X., Li, C., He, Y.: Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. Adv. Neural Inf. Process. Syst. **35**, 27168–27183 (2022)
39. Yin, P., Zhang, S., Lyu, J., Osher, S., Qi, Y., Xin, J.: Binaryrelax: A relaxation approach for training deep neural networks with quantized weights. SIAM J. Imaging Sci. **11**(4), 2205–2223 (2018)
40. Zhang, J., Saab, R.: SPFQ: A Stochastic Algorithm and Its Error Analysis for Neural Network Quantization. arXiv preprint arXiv:2309.10975 (2023)
41. Zhang, J., Zhou, Y., Saab, R.: Post-training quantization for neural networks with provable guarantees. SIAM J. Math. Data Sci. **5**(2), 373–399 (2023)

42. Zhang, J., Kannan, H., Cloninger, A., Saab, R.: Sigma-Delta and distributed noise-shaping quantization methods for random Fourier features. Information and Inference: A Journal of the IMA 13.1, iaad052 (2024)