A SIMPLE APPROACH FOR QUANTIZING NEURAL NETWORKS

JOHANNES MALY AND RAYAN SAAB

Abstract. In this short note, we propose a new method for quantizing the weights of a fully trained neural network. A simple deterministic pre-processing step allows us to quantize network layers via *memoryless scalar quantization* while preserving the network performance on given training data. On one hand, the computational complexity of this pre-processing slightly exceeds that of state-of-the-art algorithms in the literature. On the other hand, our approach does not require any hyper-parameter tuning and, in contrast to previous methods, allows a plain analysis. We provide rigorous theoretical guarantees in the case of quantizing single network layers and show that the relative error decays with the number of parameters in the network if the training data behaves well, e.g., if it is sampled from suitable random distributions. The developed method also readily allows the quantization of deep networks by consecutive application to single layers.

1. Introduction

An *L-layer feedforward neural network*, $\Phi : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ is a function whose action on a vector $\mathbf{x} \in \mathbb{R}^{N_0}$ is given by

$$\Phi(\mathbf{x}) := \varphi \circ A^{(L)}. \circ \cdots \circ \varphi \circ A^{(1)}(\mathbf{x}), \tag{1}$$

where the *activation function* $\varphi : \mathbb{R} \to \mathbb{R}$ acts entry-wise on vectors, and $A^{(\ell)} : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$ are affine maps given by $A^{(\ell)}(\mathbf{z}) := \mathbf{W}^{(\ell)\top}\mathbf{z} + \mathbf{b}^{(\ell)}$. We call $\mathbf{W}^{(\ell)} \in \mathbb{R}^{N_{\ell-1}\times N_\ell}$ and $\mathbf{b}^{(\ell)} \in \mathbb{R}^{N_\ell}$ the *weight matrix* and *bias vector* associated with the ℓ-th layer of Φ. The i-th *neuron* (without activation) of the ℓ-th layer is then the map $\mathbf{z} \mapsto (\mathbf{w}_i^{(\ell)})^\top \mathbf{z} + b_i^{(\ell)}$, where $\mathbf{w}_i^{(\ell)}$ denotes the i-th column of $\mathbf{W}^{(\ell)}$. In modern machine learning, neural networks have become the state-of-the-art tool for various tasks like speech recognition, autonomous driving, and games [9, 12, 5]. Nevertheless, such networks tend to require a large number of layers, and a large number of parameters N_ℓ per layer. As a result, they are associated with high computational costs, both in storage/memory and in power usage. In order to reduce these costs, one approach is to use coarsly quantized parameters, i.e., quantized weights of the neural network (see [7, 3, 4]). This can be achieved either by restricting the elements of $\mathbf{W}^{(\ell)}$ and $\mathbf{b}^{(\ell)}$ at training time to take on values from a discrete finite set, or by replacing them with elements from such a set after training [8]. The first approach entails *quantization-aware training*, whereas the second involves *post-training quantization* and is the focus of our work. In this context, quantization consists of replacing the, e.g., 32-bit floating point numbers that constitute the weights of an already trained neural network with coarsly quantized counterparts that can be represented with many fewer bits. The challenge lies in not degrading the performance of the network by doing so.

To accomplish this task, one can progressively approach the problem one layer at a time, quantizing each neuron (column of $\mathbf{W}^{(\ell)}$) in the layer before advancing to the next layer. Ignoring the bias terms for the moment and considering, for example, the first layer of the neural network of

width N_1, one can select an appropriate *alphabet* $\mathcal{A}$ and devise a map

$$\mathcal{Q}: \quad \mathbb{R}^{N_0} \to \mathcal{A}^{N_0}$$
$$\mathbf{w} \mapsto \mathbf{q}$$

that respects $\mathbf{w}^\top \mathbf{X} \approx \mathbf{q}^\top \mathbf{X}$ or equivalently $\mathbf{X}^\top \mathbf{w} \approx \mathbf{X}^\top \mathbf{q}$, where $\mathbf{X} \in \mathbb{R}^{N_0 \times m}$ is a matrix with m training samples as its columns. Defining $B := \log_2(|\mathcal{A}|)$, each quantized neuron $\mathbf{q}$ from among the N_1 neurons in the first layer can now be represented using BN_0 bits. Variants of this approach have been explored recently (e.g., [1, 2, 18, 10, 11, 17]), including in the nascent literature that seeks rigorous theoretical guarantees for neural network quantization (e.g., [11, 17, 16, 6]).

While this general approach seems to work reasonably well in practice, and includes recent algorithms with theoretical guarantees, there are some important challenges associated with it. First, an appropriate alphabet $\mathcal{A}$ must be chosen for each layer so that there even exists $\mathbf{q} \in \mathcal{A}^{N_0}$ with $\mathbf{X}^\top \mathbf{w} \approx \mathbf{X}^\top \mathbf{q}$. Second, the obvious approach once such an $\mathcal{A}$ is chosen, consists in finding $\mathbf{q} \in \mathcal{A}^{N_0}$ that minimizes the objective function $\|\mathbf{X}^\top(\mathbf{w} - \mathbf{q})\|_2$. However, this constitutes an integer program, so it is generally NP-hard, and remains so for other objective functions. Despite these challenges, various ad-hoc computationally feasible approaches have been proposed, including [1, 2, 18, 10].

1.1. **Related work.** As already alluded to, there has been recent progress in developing computationally efficient algorithms with rigorous theoretical guarantees [11, 17]. The authors of [11] propose a greedy quantization algorithm based on *noise-shaping* and analyze its performance in the case of a single layer neural network with Gaussian random training data, and they restrict their analysis to the case of the alphabet $\{-1, 0, 1\}$. Notably, the algorithm proposed in [11] has computational complexity $\mathcal{O}(mN_0)$ per neuron, which is near optimal, as the size of the training data is mN_0. Subsequently [17] extends the analysis to more general distributions and alphabets. For example, if $\mathbf{X}$ is uniformly distributed in the ball of $\mathbb{R}^m$ of radius r, [17] shows that the error of quantizing a neuron $\mathbf{w} \in \mathbb{R}^{N_0}$ satisfies

$$\|\mathbf{X}^\top \mathbf{w} - \mathbf{X}^\top \mathbf{q}\|_2^2 \lesssim mr^2 \log N_0, \tag{2}$$

with high probability, where $\mathbf{q} \in \mathcal{A}^{N_0}$ contains the quantized weights. As a corollary, one can see that for generic vectors $\mathbf{w}$ that are independent of $\mathbf{X}$,

$$\frac{\|\mathbf{X}^\top \mathbf{w} - \mathbf{X}^\top \mathbf{q}\|_2^2}{\|\mathbf{X}^\top \mathbf{w}\|_2^2} \lesssim \frac{m \log N_0}{N_0} \tag{3}$$

with high probability. One issue with the theory in [11, 17] is that it requires the largest element in $\mathcal{A}$ to be at least as large as $\|\mathbf{w}\|_\infty$. While this may seem innocuous for a single neuron $\mathbf{w}$, in practice the different columns in a weight matrix $\mathbf{W}^{(\ell)}$ may be bounded differently. As a result, one must either use different alphabets for each neuron, or accept a potentially large error bound. In practice, however, the numerical experiments in [11, 17] use a single alphabet with a carefully tuned range to optimize the performance of the algorithm. In other words, they introduced an additional hyper-parameter that needs to be set a priori.

1.2. **Contribution.** In this work, we examine how one can reliably quantize a fully trained network Φ via *memoryless scalar quantization*. Like [11, 17] before, we restrict our analysis to quantizing a single network layer. We show that, surprisingly, a simple pre-processing step on $\mathbf{w}$ allows us to quantize the weights in a naive way and obtain theoretical guarantees as in (3). In contrast to the noise-shaping approaches in [11, 17], the analysis is however remarkably simple. Moreover, even when quantizing full network layers no additional hyper-parameter tuning is required to boost the performance. The price we pay is that the pre-processing step is slightly more expensive in computation time. Let us mention that while our analysis is restricted to a single layer, the developed method readily lends itself to the quantization of deep networks as well by consecutive application to single layers.

1.3. **Notation.** We abbreviate $[n] := \{1, \dots, n\}$, for $n \in \mathbb{N}$. We use C, c, to denote absolute constants, while $a \lesssim b$ denotes $a \leq Cb$. Similarly, $a \gtrsim b$ means $a \geq Cb$ and $a \simeq b$ denotes $a \lesssim b \lesssim a$. Henceforth, as justified by the observation $\mathbf{w}^\top \mathbf{x} + b = \langle (\mathbf{w}, b), (\mathbf{x}, 1) \rangle$, we will ignore the bias term $\mathbf{b}^{(\ell)}$ in (1) as it can simply be treated as an extra column of $\mathbf{W}^{(\ell)}$. Under this prerequisite, the i-th neuron of the ℓ-th layer is defined as the map $\mathbf{z} \mapsto (\mathbf{w}_i^{(\ell)})^\top \mathbf{z}$, where $\mathbf{w}_i^{(\ell)}$ denotes the i-th column of $\mathbf{W}^{(\ell)}$. We define $\mathbf{0}$ and $\mathbf{1}$ to be the vector/matrix of zeros and ones, respectively (the dimensions will always be clear from the context). For a matrix $\mathbf{W}$, we denote the operator norm by $\|\mathbf{W}\|$ and the maximum entry in absolute value by $\|\mathbf{W}\|_\infty$. If $\mathbf{A}$ is an $n_1 \times n_2$ matrix and $J \subset [n_2]$, we define the restricted kernel

$$\ker_J(\mathbf{A}) := \{\mathbf{b} \in \ker(\mathbf{A}) \colon b_i = 0 \text{ if } i \in J^c\}. \tag{4}$$

For $K \geq 1$, we define *midrise* alphabets having $2K$ elements, as sets of the form

$$\mathcal{A} = \{\pm(k - 1/2)\delta : 1 \leq k \leq K, k \in \mathbb{Z}\} \tag{5}$$

and, similarly, *midtread* alphabets with $2K + 1$ elements as sets of the form

$$\mathcal{A} = \{\pm k\delta : 0 \leq k \leq K, k \in \mathbb{Z}\} \tag{6}$$

where $\delta > 0$ denotes the quantization step size. The simplest examples of such alphabets are the 1-bit alphabet $\{-1, 1\}$, and the ternary alphabet $\{-1, 0, 1\}$. The *memoryless scalar quantizer* (MSQ) associated with an alphabet $\mathcal{A}$ is given by $Q : \mathbb{R} \to \mathcal{A}$ with

$$Q(z) := \arg\min_{p \in \mathcal{A}} |z - p|. \tag{7}$$

For instance, the MSQ map Q with $\mathcal{A} = \{-1, 1\}$ is given by the two-valued sign-function

$$\operatorname{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0. \end{cases}$$

In the following, we apply the quantizer Q entry-wise to vectors and matrices.

2. Quantizing a network layer

In this work, we consider uniform memoryless quantization.

Algorithm 1 : Neuron Preprocessing

Given: $\mathbf{A}_0 \in \mathbb{R}^{m\times n}$ $(n > m)$, $\mathbf{z}_0 \in \mathbb{R}^n$, and $c \geq \|\mathbf{z}_0\|_\infty$

Initialize $J_0 = \{i \in [n]\colon \text{the } i\text{-th column of } \mathbf{A}_0 \text{ is zero}\}$
Define $\mathbf{b} \in \mathbb{R}^n$ via $\mathbf{b}_{J_0^c} = \mathbf{0}$ and $b_i = c - (z_0)_i$, for $i \in J_0$. Then $\mathbf{b} \in \ker(\mathbf{A}_0)$ and $|(z_0)_i + b_i| = c$ for $i \in J_0$
Replace $\mathbf{z}_0$ with $\mathbf{z}_0 + \mathbf{b}$
Initialize $k = 0$
while $\||\mathbf{z}_k| - c\mathbf{1}\|_0 > m$ (which implies that $\ker_{J_k}(\mathbf{A}_k) \neq \{\mathbf{0}\}$, cf. Equation (4)) **do**
    Compute $\mathbf{b} \in \ker_{J_k}(\mathbf{A}_k)$, $\mathbf{b} \neq \mathbf{0}$
    Compute $\alpha \in \mathbb{R}$ with $\|\mathbf{z}_k + \alpha\mathbf{b}\|_\infty = c$
    $\mathbf{z}_{k+1}$=$\mathbf{z}_k + \alpha\mathbf{b} \in \mathbb{R}^n$
    J_{k+1}=$J_k \cup \{i \in [n]\colon |(\mathbf{z}_{k+1})_i| = c\}$
    $\mathbf{A}_{k+1}$=$(\mathbf{A}_k)_{J_{k+1}^c} \in \mathbb{R}^{m\times n}$ (Matrix in which all columns indexed by J_{k+1} are set to zero.)
    $k \leftarrow k + 1$
end while
$k_{\text{final}} = k$

Return: $\mathbf{z}_{k_{\text{final}}}$ for which $\mathbf{A}\mathbf{z}_{k_{\text{final}}} = \mathbf{A}\mathbf{z}_0$, $\|\mathbf{z}_{k_{\text{final}}}\|_\infty = c$, and $\||\mathbf{z}_{k_{\text{final}}}| - c\mathbf{1}\|_0 \leq m$

Definition 2.1 (Uniform B-bit quantizer)**.** For any midrise or midtread alphabet $\mathcal{A}$ with

$$\max_{q\in\mathcal{A}} |q| \;=\; 1,$$

we define the quantization alphabet as $\mathcal{A}_c = c \cdot \mathcal{A} = \{-c, \dots, c\}$, for some suitable $c > 0$.

If $|\mathcal{A}_c| = 2^B$, for $B \in \mathbb{N}$, then $\mathcal{A}_c$ can be encoded in B bits, the worst-case distortion of $\mathcal{A}_c$ on $[-c, c]$

$$\delta_{\mathcal{A}_c} = \max_{z\in[-c,c]} |z - Q(z)|$$

satisfies $\delta_{\mathcal{A}_c} = c2^{-B}$, and we call the associated MSQ map Q_c defined in (7), a *uniform B-bit quantizer*.

We focus on quantization of single layer networks, i.e., the network $\Phi\colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_1}$, $\Phi = \varphi \circ A$ consists of one layer. It is thus determined by the weight matrix $\mathbf{W} \in \mathbb{R}^{N_1\times N_0}$ of A. We furthermore assume that we have access to training data $\mathbf{x}_i, i \in [m]$ for which we have $\Phi(\mathbf{x}_i) = \mathbf{y}_i$. We consider the overparametrized setting $N_0, N_1 \gg m$, i.e., there are far more trainable parameters than training samples. For convenience, we define the matrix of input data $\mathbf{X} = \big(\mathbf{x}_1, \dots, \mathbf{x}_m\big) \in \mathbb{R}^{N_0\times m}$. For fixed $B \in \mathbb{N}$, i.e., $|\mathcal{A}_c| = 2^B$, our goal is thus to find a constant $c > 0$ and a matrix $\mathbf{Q} \in \mathcal{A}_c^{N_0\times N_1}$ such that $\mathbf{Q}^\top\mathbf{X} \approx \mathbf{W}^\top\mathbf{X}$. Lipschitz-continuity of the activation function φ then guarantees $\varphi(\mathbf{Q}^\top\mathbf{X}) \;\approx\; \varphi(\mathbf{W}^\top\mathbf{X})$.

2.1. **Quantizing a single neuron.** As a proof of concept, let us begin with the simpler case of quantizing one single neuron, i.e., the map $\mathbf{z} \mapsto \mathbf{w}^\top\mathbf{z}$. Given the data $\mathbf{X}$ we wish to construct $\mathbf{q} \in \mathcal{A}_c^{N_0}$ such that $\mathbf{q}^\top\mathbf{X} \approx \mathbf{w}^\top\mathbf{X}$, or equivalently, $\mathbf{X}^\top\mathbf{q} \approx \mathbf{X}^\top\mathbf{w}$. To this end, we define $\widehat{c} = \|\mathbf{w}\|_\infty$

and

$$\mathbf{w}^\sharp \in \arg\min_{\mathbf{z}\in\mathbb{R}^{N_0}} \||\mathbf{z}| - \widehat{c}\mathbf{1}\|_0, \qquad \text{s.t. } \mathbf{X}^\top\mathbf{z} = \mathbf{X}^\top\mathbf{w} \text{ and } \|\mathbf{z}\|_\infty \leq \widehat{c}, \tag{8}$$

where $\|\cdot\|_0$ is not a norm but counts the number of non-zero entries and $|\cdot|$ is applied entry-wise. The idea behind (8) is to find a vector $\mathbf{w}^\sharp$ that mimics the action of $\mathbf{w}$ on the data, while at the same time having most of its entries exactly take on the values $\pm\widehat{c}$. Depending on the quantizer alphabet, the remaining entries can then be quantized more finely and the error can be easily bounded well. Unfortunately, the objective in (8) is discrete and renders the optimization problem hard to solve.

As a work-around, we propose Algorithm 1 as an efficient procedure to compute substitute solutions. It is straight-forward to check that the algorithm (applied to $\mathbf{A}_0 = \mathbf{X}^\top$, $\mathbf{z}_0 = \mathbf{w}$, and $c = \widehat{c}$) stops after at most $N_0 - m$ iterations and produces a vector $\widehat{\mathbf{w}}$ with $\mathbf{X}^\top\widehat{\mathbf{w}} = \mathbf{X}^\top\mathbf{w}$, $\|\widehat{\mathbf{w}}\|_\infty = \widehat{c}$, and $\||\widehat{\mathbf{w}}| - \widehat{c}\mathbf{1}\|_0 \leq m$: indeed, Algorithm 1 changes the input only along the kernel of $\mathbf{X}^\top$, keeps the ℓ_∞-norm of the iterates constant, and reduces the quantity $\||\mathbf{z}_k| - c\mathbf{1}\|_0$ by at least one in each iteration. Although the computed solution $\widehat{\mathbf{w}}$ is not necessarily optimal in the sense of (8), it suffices for our purpose. We now set

$$\mathbf{q} = Q_{\widehat{c}}(\widehat{\mathbf{w}}), \tag{9}$$

where the MSQ $Q_{\widehat{c}}$ is applied entry-wise. We can deduce the following result.

Theorem 2.2. *Let $N_0 > m$, $\mathbf{w} \in \mathbb{R}^{N_0}$, and let $\mathbf{X} \in \mathbb{R}^{N_0\times m}$. Define the data complexity parameter*

$$\Gamma(\mathbf{X}) = \sup_{\substack{T\subset[N_0]\\|T|=m}} \|\mathbf{X}^\top|_T\|,$$

where $\mathbf{X}^\top|_T$ denotes $\mathbf{X}^\top$ with all columns not indexed by T set to zero. Let $\mathbf{Q}_{\widehat{c}}$ be a uniform B-bit quantizer as in Definition 2.1, for $\widehat{c} = \|\mathbf{w}\|_\infty$ and $B \in \mathbb{N}$. Then, if $\mathbf{q}$ is constructed via (9), where $\widehat{\mathbf{w}}$ is the output of Algorithm 1, we have that

$$\frac{\|\mathbf{X}^\top\mathbf{w} - \mathbf{X}^\top\mathbf{q}\|_2}{\|\mathbf{X}^\top\mathbf{w}\|_2} \leq 2^{-B}\cdot\Gamma(\mathbf{X})\cdot\frac{\sqrt{m}\|\mathbf{w}\|_\infty}{\|\mathbf{X}^\top\mathbf{w}\|_2}. \tag{10}$$

Proof. First note that, for any matrix $\mathbf{X} \in \mathbb{R}^{N_0\times m}$ with $N_0 > m$, the approximate solution $\widehat{\mathbf{w}}$ of (8) computed by Algorithm 1 consists of $N_0 - m$ entries that are of magnitude $\widehat{c} = \|\mathbf{w}\|_\infty$ and has m remaining entries of (possibly) smaller magnitude. Let us denote the set of these m indices by $T \subset [N_0]$. Recall that $\mathbf{X}^\top\mathbf{w} = \mathbf{X}^\top\widehat{\mathbf{w}}$ and that the B-bit quantizer $Q_{\widehat{c}}$ has an entry-wise worst-case distortion of $2^{-B}\widehat{c} = 2^{-B}\|\mathbf{w}\|_\infty$ on the cube $[-\widehat{c},\widehat{c}]^{N_0}$. By the definition of $\mathbf{q}$, it then follows that

$$\begin{aligned}\|\mathbf{X}^\top\mathbf{w} - \mathbf{X}^\top\mathbf{q}\|_2 &= \|\mathbf{X}^\top\widehat{\mathbf{w}} - \mathbf{X}^\top\mathbf{q}\|_2 = \|\mathbf{X}^\top|_T\cdot(\widehat{\mathbf{w}} - \mathbf{q})\|_2 \leq \Gamma(\mathbf{X})\cdot\sqrt{m}\|\widehat{\mathbf{w}} - \mathbf{q}\|_\infty\\ &\leq \Gamma(\mathbf{X})\cdot 2^{-B}\sqrt{m}\|\widehat{\mathbf{w}}\|_\infty,\end{aligned}$$

where $\mathbf{X}^\top|_T$ denotes the matrix $\mathbf{X}^\top$ restricted to the columns indexed in T. We thus have that

$$\frac{\|\mathbf{X}^\top\mathbf{w} - \mathbf{X}^\top\mathbf{q}\|_2}{\|\mathbf{X}^\top\mathbf{w}\|_2} \lesssim 2^{-B}\cdot\Gamma(\mathbf{X})\cdot\frac{\sqrt{m}\|\widehat{\mathbf{w}}\|_\infty}{\|\mathbf{X}^\top\mathbf{w}\|_2}.$$

The desired result follows trivially from the fact that $\|\widehat{\mathbf{w}}\|_\infty = \|\mathbf{w}\|_\infty$, cf. Algorithm 1. □

If the data $\mathbf{X}$ is, e.g., Gaussian, Theorem 2.2 shows that the quantized neuron defined via $\mathbf{q}$ behaves similarly to the original neuron.

Corollary 2.3. *Let $N_0 > m$, $\mathbf{w} \in \mathbb{R}^{N_0}$, and let $\mathbf{X} \in \mathbb{R}^{N_0 \times m}$ have i.i.d. entries $X_{i,j} \sim \mathcal{N}(0,1)$. Let $\mathbf{Q}_{\widehat{c}}$ be a uniform B-bit quantizer as defined in Definition 2.1, for $\widehat{c} = \|\mathbf{w}\|_\infty$ and $B \in \mathbb{N}$. Then, with probability at least $1 - 4e^{-m \log(N_0)}$ on the draw of $\mathbf{X}$, if $\mathbf{q}$ is constructed via (9), where $\widehat{\mathbf{w}}$ is the output of Algorithm 1, we have that*

$$\frac{\|\mathbf{X}^\top \mathbf{w} - \mathbf{X}^\top \mathbf{q}\|_2}{\|\mathbf{X}^\top \mathbf{w}\|_2} \lesssim 2^{-B} \frac{\sqrt{m \log(N_0)} \|\mathbf{w}\|_\infty}{\|\mathbf{w}\|_2}. \tag{11}$$

Proof. Since the non-zero entries of $\mathbf{X}^\top|_T$ form an $m \times m$ sub-matrix of an $m \times N_0$ Gaussian matrix, we get from [14, Theorem 4.4.5], and a union bound over the $\binom{N_0}{m}$ submatrices of size $m \times m$, that

$$\Gamma(\mathbf{X}) \lesssim \sqrt{m \log(N_0)} \tag{12}$$

with probability at least $1 - 2e^{-m \log(N_0)}$. At the same time, the vector $\mathbf{X}^\top \mathbf{w}$ is Gaussian, so Lemma A.1 yields that with probability at least $1 - 2e^{-m}$

$$\|\mathbf{X}^\top \mathbf{w}\|_2 \gtrsim \sqrt{m} \|\mathbf{w}\|_2. \tag{13}$$

Combining (12) and (13) by a union bound and inserting them into (10), we obtain that (11) holds with probability at least $1 - 4e^{-m \log(N_0)}$. □

A couple of comments are in order. First, the assumption that the entries of $\mathbf{X}$ in Corollary 2.3 are standard Gaussian is only for ease of exposition. Indeed, the conclusions of the corollary hold for any (e.g., subgaussian) distribution for which (12) and (13) are satisfied with appropriately high probability. Second, Corollary 2.3 strongly resembles the state-of-the-art results [11, Theorem 2] and [17, Section 2]. Its proof is, however, remarkably simpler since our quantization technique is not adaptive but relies on the single pre-processing step in (8). For a generic weight vector $\mathbf{w} \in \mathbb{R}^{N_0}$, i.e., $\|\mathbf{w}\|_2 \simeq \sqrt{N_0} \|\mathbf{w}\|_\infty$, the bound in (11) becomes

$$\frac{\|\mathbf{X}^\top \mathbf{w} - \mathbf{X}^\top \mathbf{q}\|_2}{\|\mathbf{X}^\top \mathbf{w}\|_2} \lesssim 2^{-B} \sqrt{\frac{m \log(N_0)}{N_0}}.$$

Being of the same form as the just mentioned results, cf. Equation (3) above, this is a meaningful estimate in the overparametrized regime where the number of parameters exceeds the number of training data points, i.e., $N_0 \gg m$. Let us also emphasize that if the activation function φ is L-Lipschitz continuous, the bound in (11) directly extends to the concatenation of neuron and activation function and becomes

$$\frac{\|\varphi(\mathbf{X}^\top \mathbf{w}) - \varphi(\mathbf{X}^\top \mathbf{q})\|_2}{\|\mathbf{X}^\top \mathbf{w}\|_2} \lesssim L 2^{-B} \frac{\sqrt{m \log(N_0)} \|\mathbf{w}\|_\infty}{\|\mathbf{w}\|_2}. \tag{14}$$

Remark 2.4. Computing the complexity parameter $\Gamma(\mathbf{X})$ that appears in Theorem 2.2 is challenging in general. However, it can trivially be bounded by $\|\mathbf{X}\|$. If $\mathbf{X}$ is a frame with upper frame bound C, this implies that $\Gamma(\mathbf{X}) \leq C$. In particular, if $\mathbf{X} \in \mathbb{R}^{m \times N_0}$ is a concatenation of N_0/m frames with upper frame bound C (assuming for simplicity that m divides N_0), it is easy to check that $\Gamma(\mathbf{X}) \leq C\sqrt{N_0/m}$. For $N_0 = \mathcal{O}(m^2)$, this leads to $\Gamma(\mathbf{X}) \lesssim \sqrt{m}$ and thus to the same bound as in (11).

Remark 2.5. In order to improve the bound in Theorem 2.2 and Corollary 2.3, we can find a $\mathbf{z}$ that minimizes the ℓ_∞-norm among all vectors satisfying $\||\mathbf{z}| - \|\mathbf{z}\|_\infty \mathbf{1}\|_0 \leq m$ and $\mathbf{X}^\top \mathbf{z} = \mathbf{X}^\top \mathbf{w}$. Indeed, if the rows of $\mathbf{X}$ are in general position, Lemma A.2 shows that any solution

$$\widehat{\mathbf{w}} \in \arg\min_{\mathbf{z}\in\mathbb{R}^{N_0}} \|\mathbf{z}\|_\infty, \qquad \text{s.t. } \mathbf{X}^\top \mathbf{z} = \mathbf{X}^\top \mathbf{w} \tag{15}$$

fulfills $\||\widehat{\mathbf{w}}| - \widehat{c}\mathbf{1}\|_0 \leq m$, for $\widehat{c} := \|\widehat{\mathbf{w}}\|_\infty \leq \|\mathbf{w}\|_\infty$. This means we can solve (15) instead of using Algorithm 1, which, can be more efficient, depending on the ratio between N_0 and m, cf. Section 2.3. We emphasize, however, that Algorithm 1 does not require the rows of $\mathbf{X}$ to be in general position.

One may wonder how the quantized neuron performs on data from outside of the training set. The following theorem is an improved version of [11, Theorem 3] and answers this question in the case of new data drawn from the span of the training set.

Theorem 2.6. *Let* $\mathbf{X}, \mathbf{w}$ *and* $\mathbf{q}$ *be as in Corollary 2.3 and suppose that* $N_0 > m$. *Then with probability at least* $1 - 4e^{-2m\log(N_0)}$ *we have for any data point* $\mathbf{z}$ *that lies in the span of* $\mathbf{X}$ *that*

$$|\mathbf{z}^\top(\mathbf{w} - \mathbf{q})| \lesssim 2^{-B} \left(\frac{m\sqrt{\log(N_0)}}{\sqrt{N_0} - \sqrt{m}} \right) \|\mathbf{z}\|_2 \|\mathbf{w}\|_\infty. \tag{16}$$

Proof. Define the set $\mathbf{X}(B_2^{N_0}) = \{\boldsymbol{\zeta} \in \mathbb{R}^{N_0} : \boldsymbol{\zeta} = \mathbf{X}\mathbf{h} \text{ with } \|\mathbf{h}\|_2 \leq 1\}$ which is a bounded subset of the span of the data points. Then, for any $\boldsymbol{\zeta} \in \mathbf{X}(B_2^{N_0})$ one has

$$|\boldsymbol{\zeta}^\top(\mathbf{w} - \mathbf{q})| = \Big| \sum_{i=1}^m h_i \mathbf{x}_i^\top (\mathbf{w} - \mathbf{q}) \Big| \leq \|\mathbf{h}\|_2 \|\mathbf{X}^\top(\mathbf{w} - \mathbf{q})\|_2 \lesssim 2^{-B} m\sqrt{\log(N_0)} \|\mathbf{w}\|_\infty,$$

where we first used the Cauchy-Schwarz inequality, then the bound for the numerator in Corollary 2.3 in the second inequality, which holds with probability at least $1 - 2e^{-m\log(N_0)}$. For $\mathbf{z}$ defined as in the statement, let $\mathbf{p} = \alpha_\star \mathbf{z}$ with

$$\alpha_\star = \max_{\alpha \geq 0} \alpha, \qquad \text{s.t. } \alpha\mathbf{z} \in \mathbf{X}(B_2^{N_0}).$$

By using that $\mathbf{p} \in \mathbf{X}(B_2^{N_0})$, any strictly positive lower bound on $\alpha_\star$ would then yield a bound on our quantity of interest in (16) via

$$|\mathbf{z}^\top(\mathbf{w} - \mathbf{q})| = \frac{1}{\alpha_\star} |\mathbf{p}^\top(\mathbf{w} - \mathbf{q})| \lesssim \frac{2^{-B} m\sqrt{\log(N_0)} \|\mathbf{w}\|_\infty}{\alpha_\star}.$$

All that remains is to find a suitable lower bound for $\alpha_\star$. Since $\mathbf{z}$ is in the span of $\mathbf{X}$, there exists $\mathbf{h}_\mathbf{z} \in \mathbb{R}^m$ with $\mathbf{z} = \mathbf{X}\mathbf{h}_\mathbf{z}$. Since $N_0 > m$ and $\mathbf{X}$ is Gaussian, the embedding is almost surely injective and $\mathbf{h}_\mathbf{z}$ is unique. Setting $\bar{\mathbf{h}}_\mathbf{z} = \frac{\mathbf{h}_\mathbf{z}}{\|\mathbf{h}_\mathbf{z}\|_2}$, we have that $\mathbf{X}\bar{\mathbf{h}}_\mathbf{z} = \frac{\mathbf{z}}{\|\mathbf{h}_\mathbf{z}\|_2}$ and $\|\bar{\mathbf{h}}_\mathbf{z}\|_2 = 1$ which implies that $\alpha_\star \geq \|\bar{\mathbf{h}}_\mathbf{z}\|_2^{-1}$. We can now estimate that

$$\frac{\|\mathbf{z}\|_2}{\|\mathbf{h}_\mathbf{z}\|_2} = \|\mathbf{X}\bar{\mathbf{h}}_\mathbf{z}\|_2 \geq \min_{\|\boldsymbol{\zeta}\|_2 = 1} \|\mathbf{X}\boldsymbol{\zeta}\|_2 \gtrsim \sqrt{N_0} - \sqrt{m},$$

where the last inequality holds with probability at least $1 - 2e^{-m}$ (over the draw of $\mathbf{X}$) and follows from standard bounds on the singular values of Gaussian matrices, e.g., [14, Theorem 4.6.1]. Consequently, we obtain with the same probability that $\alpha_\star \geq \|\bar{\mathbf{h}}_\mathbf{z}\|_2^{-1} \gtrsim \frac{\sqrt{N_0} - \sqrt{m}}{\|\mathbf{z}\|_2}$. The claim follows from a union bound over both events. ∎

2.2. **Quantizing a network layer.** The main challenge in generalizing (8)-(9) to a whole layer is that each neuron of the layer has a different upper bound on the magnitude of its entries, i.e., a different $\widehat{c}$. There is, however, a simple way to deal with this. First, define $\widehat{C} = \|\mathbf{W}\|_\infty$ where $\mathbf{W}$ is the weight-matrix with columns $\mathbf{w}_i \in \mathbb{R}^{N_0}$ corresponding to single neurons, for $i \in [N_1]$. The value of $\widehat{C}$ corresponds to the maximum $\widehat{c}_i = \|\mathbf{w}_i\|_\infty$ of all single neurons $\mathbf{w}_i$. We now solve

$$\mathbf{W}^\sharp_{\widehat{C}} \in \arg\min_{\mathbf{Z}\in\mathbb{R}^{N_0\times N_1}} \||\mathbf{Z}| - \widehat{C}\mathbf{1}\|_0, \qquad \text{s.t. } \mathbf{X}^\top\mathbf{Z} = \mathbf{X}^\top\mathbf{W} \text{ and } \|\mathbf{Z}\|_\infty \le \widehat{C}. \tag{17}$$

Since the optimization in (17) decouples in the single neurons, the columns $\mathbf{w}^\sharp_{i,\widehat{C}}$ of $\mathbf{W}^\sharp_{\widehat{C}}$ can be computed separately via

$$\mathbf{w}^\sharp_{i,\widehat{C}} \in \arg\min_{\mathbf{z}\in\mathbb{R}^{N_0}} \||\mathbf{z}| - \widehat{C}\mathbf{1}\|_0, \qquad \text{s.t. } \mathbf{X}^\top\mathbf{z} = \mathbf{X}^\top\mathbf{w}_i \text{ and } \|\mathbf{z}\|_\infty \le \widehat{C}. \tag{18}$$

Algorithm 1 applied to $\mathbf{A}_0 = \mathbf{X}^\top$, $\mathbf{z}_0 = \mathbf{w}_i$, and $c = \widehat{C}$ can be used to get approximate solutions $\widehat{\mathbf{w}}_{i,\widehat{C}}$ of (18). Having obtained a matrix $\widehat{\mathbf{W}}_{\widehat{C}}$ with columns $\widehat{\mathbf{w}}_{i,\widehat{C}}$ by consecutively applying Algorithm 1, we can now define

$$\mathbf{Q} = Q_{\widehat{C}}(\widehat{\mathbf{W}}_{\widehat{C}}). \tag{19}$$

Since each column of $\widehat{\mathbf{W}}_{\widehat{C}}$ has at most $N_0 - m$ entries that are smaller than $\widehat{C}$ in magnitude, it is straight-forward to extend Theorem 2.2 and Corollary 2.3 to the following results.

Theorem 2.7. *Let $N_0 > m$, $\mathbf{W} \in \mathbb{R}^{N_0\times N_1}$, and let $\mathbf{X} \in \mathbb{R}^{N_0\times m}$. Let $\mathbf{Q}_{\widehat{C}}$ be a uniform B-bit quantizer as in Definition 2.1 with $\widehat{C} = \|\mathbf{W}\|_\infty$ and $B \in \mathbb{N}$. Then, if $\mathbf{Q}$ is constructed via (19), where the columns of $\widehat{\mathbf{W}}_{\widehat{C}}$ are computed by Algorithm 1, we have that*

$$\frac{\|\mathbf{X}^\top\mathbf{W} - \mathbf{X}^\top\mathbf{Q}\|_F}{\|\mathbf{X}^\top\mathbf{W}\|_F} \le 2^{-B}\cdot\Gamma(\mathbf{X})\cdot\frac{\sqrt{N_1 m}\|\mathbf{W}\|_\infty}{\|\mathbf{X}^\top\mathbf{W}\|_F}, \tag{20}$$

where $\Gamma(\mathbf{X})$ is the data complexity parameter from Theorem 2.2.

Proof. The result follows by applying the same reasoning as in the proof of Theorem 2.2 to each of the columns $\mathbf{q}_i$ of $\mathbf{Q}$ independently, i.e.,

$$\|\mathbf{X}^\top\mathbf{w}_i - \mathbf{X}^\top\mathbf{q}_i\|_2 \le \Gamma(\mathbf{X})\cdot 2^{-B}\sqrt{m}\|\widehat{\mathbf{w}}_i\|_\infty,$$

for any $i \in [N_1]$. This yields

$$\frac{\|\mathbf{X}^\top\mathbf{W} - \mathbf{X}^\top\mathbf{Q}\|_F^2}{\|\mathbf{X}^\top\mathbf{W}\|_F^2} = \frac{\sum_{i=1}^{N_1}\|\mathbf{X}^\top\mathbf{w}_i - \mathbf{X}^\top\mathbf{q}_i\|_2^2}{\|\mathbf{X}^\top\mathbf{W}\|_F^2} \le 2^{-2B}\cdot\Gamma(\mathbf{X})^2\cdot\frac{N_1 m\|\widehat{\mathbf{W}}_{\widehat{C}}\|_\infty^2}{\|\mathbf{X}^\top\mathbf{W}\|_F^2},$$

and thus the claim since $\|\widehat{\mathbf{W}}_{\widehat{C}}\|_\infty = \|\mathbf{W}\|_\infty$. □

Along the lines of Corollary 2.3 one obtains then the following.

Theorem 2.8. *Let $N_0 > m \ge \log(N_1)$, $\mathbf{W} \in \mathbb{R}^{N_0\times N_1}$, and let $\mathbf{X} \in \mathbb{R}^{N_0\times m}$ have i.i.d. entries $X_{i,j} \sim \mathcal{N}(0,1)$. Let $\mathbf{Q}_{\widehat{C}}$ be a uniform B-bit quantizer as defined in Definition 2.1, for $\widehat{C} = \|\mathbf{W}\|_\infty$ and $B \in \mathbb{N}$. Then, with probability at least $1 - 4e^{\log(N_1)-m}$ on the draw of $\mathbf{X}$, if $\mathbf{Q}$ is constructed*

via (19), *where the columns of* $\widehat{\mathbf{W}}_{\widehat{C}}$ *are computed by Algorithm 1, we have that*

$$\frac{\|\mathbf{X}^\top \mathbf{W} - \mathbf{X}^\top \mathbf{Q}\|_F}{\|\mathbf{X}^\top \mathbf{W}\|_F} \lesssim 2^{-B} \frac{\sqrt{N_1 m \log(N_0)} \|\mathbf{W}\|_\infty}{\|\mathbf{W}\|_F}. \tag{21}$$

Proof. Since $\mathbf{X}^\top|_T$ is a Gaussian $m \times m$-submatrix, Theorem 2.7 and (12) yield with probability at least $1 - 2e^{-m \log(N_0)}$ that

$$\frac{\|\mathbf{X}^\top \mathbf{W} - \mathbf{X}^\top \mathbf{Q}\|_F}{\|\mathbf{X}^\top \mathbf{W}\|_F} \leq 2^{-B} \cdot \Gamma(\mathbf{X}) \cdot \frac{\sqrt{N_1 m} \|\mathbf{W}\|_\infty}{\|\mathbf{X}^\top \mathbf{W}\|_F} \leq 2^{-B} \cdot \frac{m \sqrt{N_1 \log(N_0)} \|\mathbf{W}\|_\infty}{\|\mathbf{X}^\top \mathbf{W}\|_F}. \tag{22}$$

Moreover, by applying Lemma A.1 for each $\mathbf{X}^\top \mathbf{w}_i$ and using a union bound, we obtain with probability at least $1 - 2N_1 e^{-m}$ that

$$\|\mathbf{X}^\top \mathbf{w}_i\|_2 \gtrsim \sqrt{m} \|\mathbf{w}_i\|_2, \tag{23}$$

for all $i \in [N_1]$. Combining (22) and (23) by another union bound, we thus have with probability at least $1 - 4e^{\log(N_1) - m}$ that

$$\frac{\|\mathbf{X}^\top \mathbf{W} - \mathbf{X}^\top \mathbf{Q}\|_F}{\|\mathbf{X}^\top \mathbf{W}\|_F} \lesssim 2^{-B} \cdot \frac{m \sqrt{N_1 \log(N_0)} \|\mathbf{W}\|_\infty}{\sqrt{m} \|\mathbf{W}\|_F} \leq 2^{-B} \frac{\sqrt{N_1 m \log(N_0)} \|\mathbf{W}\|_\infty}{\|\mathbf{W}\|_F}.$$

□

A similar discussion as in the single neuron case applies. If the activation function φ is L-Lipschitz continuous, then for any generic weight matrix $\mathbf{W} \in \mathbb{R}^{N_0 \times N_1}$, i.e., $\|\mathbf{W}\|_F \simeq \sqrt{N_0 N_1} \|\mathbf{W}\|_\infty$, the bound in (21) becomes

$$\frac{\|\varphi(\mathbf{X}^\top \mathbf{W}) - \varphi(\mathbf{X}^\top \mathbf{Q})\|_2}{\|\mathbf{X}^\top \mathbf{W}\|_2} \lesssim L 2^{-B} \sqrt{\frac{m \log(N_0)}{N_0}}. \tag{24}$$

As soon as $m \ll N_0$ this guarantees a small quantization error of the network when evaluated on the available data.

2.3. **Computational complexity.** As Lemma A.3 in the appendix shows, Algorithm 1 requires a run time of $\mathcal{O}(m^3 N_0)$ per single neuron. This is, by a factor of m^2, more computationally intensive than the near-optimal guarantees $\mathcal{O}(mN_0)$ provided in [11, 17]. Meanwhile, Algorithm 1 has only one hyper-parameter, namely the bit-budget B, since the required quantizer range c is automatically determined by $\mathbf{w}$ resp. $\mathbf{W}$. Moreover, we will now present two algorithmic modifications to reduce our computational complexity when $\mathbf{X}$ is in general position.

2.3.1. *First variation.* One can slightly adapt Algorithm 1 as follows:

(1) Define in the beginning $\mathbf{A}^{(0)} \in \mathbb{R}^{m \times (m+1)}$ and $\tilde{\mathbf{A}}^{(0)} \in \mathbb{R}^{m \times m}$ as

$$\mathbf{A}^{(0)} = \begin{pmatrix} | & & | \\ \mathbf{a}_1 & \cdots & \mathbf{a}_{m+1} \\ | & & | \end{pmatrix} = \left(\begin{array}{c|c} & | \\ \tilde{\mathbf{A}}^{(0)} & \mathbf{a}_{m+1} \\ & | \end{array} \right).$$

By computing $(\tilde{\mathbf{A}}^{(0)})^{-1}$ and $\tilde{\mathbf{b}} = (\tilde{\mathbf{A}}^{(0)})^{-1} \mathbf{a}_{m+1}$, the first kernel vector can be obtained via $\mathbf{b} = (\tilde{\mathbf{b}}^\top, -1)^\top \in \ker(\mathbf{A}^{(0)})$.

(2) Compute α as in Algorithm 1 but reduced to the first $(m+1)$ entries of $\mathbf{z}_0$.

(3) Choose $j' \in [m+1]$ as the smallest index j with $|(\mathbf{z}_0 + \alpha \mathbf{b})_j| = c$ and define $J_1 = J_0 \cup \{j'\}$. Note that only the first $(m+1)$ entries of $\mathbf{z}_0$ are updated to get $\mathbf{z}_1$.

(4) Generate $\mathbf{A}^{(1)}$ from $\mathbf{A}^{(0)}$ by replacing the j'-th column with $\mathbf{a}_{m+2}$.

(5) If $j' = m+1$, set $(\tilde{\mathbf{A}}^{(1)}) = (\tilde{\mathbf{A}}^{(0)})$, compute $\tilde{\mathbf{b}} = (\tilde{\mathbf{A}}^{(1)})^{-1}\mathbf{a}_{m+2}$, and obtain $\mathbf{b} = (\tilde{\mathbf{b}}^\top, -1)^\top \in \ker(\mathbf{A}^{(1)})$. If $j' < m+1$, abbreviate $\mathbf{a} = \mathbf{a}_{m+2} - \mathbf{a}_{j'}$ and note that $\tilde{\mathbf{A}}^{(1)} = \tilde{\mathbf{A}}^{(0)} + \mathbf{a}\mathbf{e}_{j'}^\top$, where $\mathbf{e}_{j'} \in \mathbb{R}^m$ denotes the j'-th unit vector. The Woodbury identity then yields

$$(\tilde{\mathbf{A}}^{(1)})^{-1} = (\tilde{\mathbf{A}}^{(0)})^{-1} + \frac{1}{1 + \mathbf{e}_{j'}^\top (\tilde{\mathbf{A}}^{(0)})^{-1}\mathbf{a}} (\tilde{\mathbf{A}}^{(0)})^{-1}\mathbf{a}\, \mathbf{e}_{j'}^\top (\tilde{\mathbf{A}}^{(0)})^{-1}$$

and $\tilde{\mathbf{b}} = (\tilde{\mathbf{A}}^{(1)})^{-1}\mathbf{a}_{m+1}$. (The matrix $(\tilde{\mathbf{A}}^{(1)})$ is invertible since the columns of $\mathbf{A}_0$ are in general position.)

(6) While keeping thorough track of index switches, repeat Steps (2)-(5) until all columns $\mathbf{a}_{m+2}, \dots, \mathbf{a}_n$ have been used.

As Lemma A.4 shows, this accelerated procedure requires a run time of $\mathcal{O}(m^2 N_0)$ which differs from [11, 17] only by a factor m.

2.3.2. *Second variation.* One can use ℓ_∞-minimization, as per (15) in Remark 2.5 instead of applying Algorithm 1. Lemma A.5 shows that, if the rows of $\mathbf{X}$ are in general position, (15) can be solved by interior point methods up to accuracy $\delta > 0$ in $\mathcal{O}(N_0^{2.5} \log(N_0/\delta))$ time. If m is of the same order as N_0, i.e., $m = \theta N_0$ for some $\theta \in (0,1)$, this run time differs from [11, 17] only by a factor $N_0^{\frac{1}{2}} \simeq m^{\frac{1}{2}}$ (up to log-factors). Note, however, that some adaptions are necessary when pre-processing a whole layer $\mathbf{W}$ via ℓ_∞-minimization. Indeed, to obtain one $\widehat{C}$ for $\mathbf{W}$ one would solve

$$\widehat{\mathbf{W}} \in \arg \min_{\mathbf{Z} \in \mathbb{R}^{N_0 \times N_1}} \|\mathbf{Z}\|_\infty, \qquad \text{s.t. } \mathbf{X}^\top \mathbf{Z} = \mathbf{X}^\top \mathbf{W}. \tag{25}$$

However, as (25) entails minimizing the infinity norm for each neuron, it follows that for several neurons the strict inequality $\|\widehat{\mathbf{w}}_i\|_\infty < \widehat{C} := \|\widehat{\mathbf{W}}\|_\infty$ may hold. This implies that we cannot quantize these neurons using $Q_{\widehat{C}}$ and still use Lemma A.2 to control the error. To resolve this issue, after solving (25) one can find for each of the N_1 neurons,

$$\mathbf{w}_i^\star \in \arg \min_{\mathbf{z} \in \mathbb{R}^{N_0}} \mathbf{a}^\top \mathbf{z} \quad \text{subject to} \quad \begin{cases} \|\mathbf{z}\|_\infty & \le \widehat{C} \\ \mathbf{X}^\top \mathbf{z} & = \mathbf{X}^\top \mathbf{w} \end{cases}, \tag{26}$$

where $\mathbf{a} \in \mathbb{R}^{N_0}$ is an arbitrary vector such that $\begin{pmatrix} \mathbf{X} \mid \mathbf{a} \end{pmatrix} \in \mathbb{R}^{N_0 \times (m+1)}$ is still in general position. As (26) is also a linear program it can be solved in $\mathcal{O}(N_0^{2.5} \log(N_0/\delta))$ time (by [13, Theorem 1.1]). Surprisingly, however, the minimizers $\mathbf{w}_i^\star$ of (26) all satisfy $\|\mathbf{w}_i^\star\|_\infty = \widehat{C}$ and $|\{i \in [N_0] \colon |w_i^\star| = \widehat{C}\}| \ge N_0 - m$ as we will now argue.

To see that $\|\mathbf{w}_i^\star\|_\infty = \widehat{C}$, suppose by way of contradiction that $\|\mathbf{w}_i^\star\|_\infty < \widehat{C}$. Then, we can select $\mathbf{h}$ with $\mathbf{X}^\top \mathbf{h} = 0$, and $\mathbf{a}^\top \mathbf{h} < 0$. There exists an $\alpha > 0$, small enough such that $\|\mathbf{w}_i^\star + \alpha \mathbf{h}\|_\infty \leq \widehat{C}$, with $\mathbf{X}^\top(\mathbf{z}^* + \alpha \mathbf{h}) = \mathbf{X}^\top \mathbf{w}$, and $\mathbf{a}^\top(\mathbf{w}_i^\star + \alpha \mathbf{h}) < \mathbf{a}^\top \mathbf{w}_i^\star$. This contradicts the optimality of $\mathbf{w}_i^\star$.

Now, consider the following auxiliary optimization problem, which we only use to prove that $\mathbf{w}_i^\star$ of (26) satisfies $|\{i \in [N_0] \colon |w_i^*| = \widehat{C}\}| \geq N_0 - m$:

$$\bar{\mathbf{w}} \in \arg\min_{\mathbf{z} \in \mathbb{R}^n} \|\mathbf{z}\|_\infty \quad \text{subject to} \quad \begin{cases} \mathbf{X}^\top \mathbf{z} & = \mathbf{X}^\top \mathbf{w}_i^\star \\ \mathbf{a}^\top \mathbf{z} & = \mathbf{a}^\top \mathbf{w}_i^\star \end{cases}. \tag{27}$$

Notice that $\|\bar{\mathbf{w}}\|_\infty \leq \widehat{C}$ since $\mathbf{w}_i^\star$, which satisfies $\|\mathbf{w}_i^\star\|_\infty = \widehat{C}$, satisfies the constraints of (27). In turn, this means that $\bar{\mathbf{w}}$ satisfies the constraints of (26). Moreover, as $\mathbf{a}^\top \bar{\mathbf{w}} = \mathbf{a}^\top \mathbf{w}_i^\star$ is the optimal value for (26), it follows that $\bar{\mathbf{w}}$ minimizes (26) and, as such, must satisfy $\|\bar{\mathbf{w}}\|_\infty = \widehat{C}$, which is also achieved by $\mathbf{w}_i^\star$. Thus the optimal value for (27) is also $\|\bar{\mathbf{w}}\|_\infty = \widehat{C}$. Collecting these results we see that (26) and (27) have the same minimizers. Now apply Lemma A.2 to (27), noting that the concatenated matrix consisting of $\mathbf{X}^T$ and $\mathbf{a}^\top$ is of size $(m+1) \times N_0$, to conclude that $|\{i \in [N_0] : |w_i^\star| = \widehat{C}\}| \geq N_0 - m$.

Appendix A. Technical addendum

It is well-known that the norm of n-dimensional Gaussian vectors strongly concentrates around $\sqrt{n}$. For the reader's convenience we recall this fact in the following lemma.

Lemma A.1 ([15, Ch. 2]). *Let $\mathbf{g} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{n\times n})$ be an n-dimensional standard Gaussian vector. Then, for any $\theta > 0$,*

$$\mathrm{P}\left[\left|\|\mathbf{g}\|_2 - \sqrt{n}\right| \geq \theta\right] \leq 2e^{-\frac{\theta^2}{8}}.$$

The next lemma proves the claim made in Remark 2.5, namely that ℓ_∞-minimization provides the same properties as Algorithm 1 if the columns of $\mathbf{X}$ are in general position.

Lemma A.2. *Let $m \leq n$, let $\mathbf{A} \in \mathbb{R}^{m\times n}$ have columns in general position, i.e., any m columns of $\mathbf{A}$ span $\mathbb{R}^m$, and let $\mathbf{b} \in \mathbb{R}^m$. Then any*

$$\mathbf{z}^\star \in \arg\min_{\mathbf{z} \in \mathbb{R}^n} \|\mathbf{z}\|_\infty, \qquad s.t.\ \mathbf{b} = \mathbf{A}\mathbf{z} \tag{28}$$

has the property that $|\{i \in [n] : |z_i^\star| = \|\mathbf{z}^\star\|_\infty\}| \geq n - m + 1$.

Proof. Suppose that any collection of m columns of $\mathbf{A}$ spans $\mathbb{R}^m$. Suppose further that $\mathbf{z}^\star$ solves (28) and that $T := \{i \in [n] : |z_i^\star| = \|\mathbf{z}^\star\|_\infty\}$ has $|T| < n - m + 1$, thus $|T^c| \geq m$. Then there exists a non-zero vector $\boldsymbol{\eta}^\varepsilon \in \ker(\mathbf{A})$ parametrized by $\varepsilon > 0$ with

$$\boldsymbol{\eta}_T^\varepsilon = -\varepsilon \cdot \mathbf{z}_T^\star = -\varepsilon \cdot \operatorname{sign}(\mathbf{z}_T^\star) \cdot \|\mathbf{z}^\star\|_\infty \quad \text{and} \quad \mathbf{A}_T \boldsymbol{\eta}_T^\varepsilon = -\mathbf{A}_{T^c} \boldsymbol{\eta}_{T^c}^\varepsilon,$$

where $\mathbf{A}_T \in \mathbb{R}^{m\times|T|}$ and $\mathbf{A}_{T^c} \in \mathbb{R}^{m\times|T^c|}$ are the submatrices of $\mathbf{A}$ formed by the columns indexed by T and T^c. Indeed, to construct such a vector, simply pick $\boldsymbol{\eta}_T^\varepsilon$ to satisfy the first equation above and pick $S \subset T^c$ with $|S| = m$, then set

$$\boldsymbol{\eta}_S^\varepsilon = -\mathbf{A}_S^{-1}(\mathbf{A}_T \boldsymbol{\eta}_T^\varepsilon) \quad \text{and} \quad \boldsymbol{\eta}_{T^c \setminus S}^\varepsilon = 0$$

Now, notice that $\mathbf{A}(\mathbf{z}^\star + \boldsymbol{\eta}^\varepsilon) = \mathbf{A}\mathbf{z}^\star = \mathbf{b}$, i.e., $\mathbf{z}^\star + \boldsymbol{\eta}^\varepsilon$ is feasible to (28) and

$$\begin{aligned}\|\mathbf{z}^\star + \boldsymbol{\eta}^\varepsilon\|_\infty &= \max\{\|\mathbf{z}^\star_T + \boldsymbol{\eta}^\varepsilon_T\|_\infty, \|\mathbf{z}^\star_S + \boldsymbol{\eta}^\varepsilon_S\|_\infty, \|\mathbf{z}^\star_{T^c\setminus S}\|_\infty\}.\\ &= \max\{(1-\varepsilon)\|\mathbf{z}^\star_T\|_\infty, \|\mathbf{z}^\star_S + \varepsilon\mathbf{A}_S^{-1}\mathbf{A}_T\mathbf{z}^\star_T\|_\infty, \|\mathbf{z}^\star_{T^c\setminus S}\|_\infty\}.\end{aligned}$$

Since there is a non-zero gap between the magnitude of entries of $\mathbf{z}^\star$ on T and T^c respectively, by continuity there is an $\varepsilon > 0$ small enough so that

$$(1-\varepsilon)\|\mathbf{z}^\star_T\|_\infty \geq \max\{\|\mathbf{z}^\star_S + \varepsilon\mathbf{A}_S^{-1}\mathbf{A}_T\boldsymbol{\eta}^\varepsilon_T\|_\infty, \|\mathbf{z}^\star_{T^c\setminus S}\|_\infty\},$$

and thus $\|\mathbf{z}^\star + \boldsymbol{\eta}^\varepsilon\|_\infty < \|\mathbf{z}^\star\|_\infty$. However, $\mathbf{z}^\star$ was defined as a minimizer of (28), which is a contradiction. It follows that T must satisfy $|T| \geq n - m + 1$. □

The final three lemmas formalize the claims made in Section 2.3 by analyzing the computational complexity of Algorithm 1 (Lemma A.3), of the accelerated version of Algorithm 1 described in Section 2.3 (Lemma A.4), and of the ℓ_∞-minimization described in Remark 2.5 (Lemma A.5).

Lemma A.3. *Let $m \leq n$. For $\mathbf{A}_0 \in \mathbb{R}^{m\times n}$ and $\mathbf{z}_0 \in \mathbb{R}^n$, Algorithm 1 computes an output $\mathbf{z}_{k_{final}}$ in $\mathcal{O}(m^3 n)$ time.*

Proof. The steps before the while-loop require $\mathcal{O}(n)$ time since they only involve adding n-dimensional vectors.

The only steps in the while-loop that are relevant for determining the computational complexity are (i) determining $\mathbf{b} \in \ker_{J_k}(\mathbf{A})$ and (ii) computing α.

First note that in (i) an arbitrary kernel element of the restricted matrix A_{J_k} is needed. One thus can reduce $\mathbf{A}_k$ to $(m+1)$ non-zero columns before computing $\mathbf{b}$, which then requires $\mathcal{O}(m^3)$ time. Let us denote the subset of indices of these $(m+1)$ columns by $I \subset [n]$.

Furthermore, it is straight-forward to check that (ii) can be computed in $\mathcal{O}(m)$ time. One just determines $\alpha_i \in \mathbb{R}$ with $|(z_k)_i + \alpha_i b_i| = c$ and $|\alpha_i|$ minimal, for all $i \in I$, and then sets $\alpha = \operatorname{argmin}_{i\in I} |\alpha_i|$. Here it is important to note that the only relevant coordinates of $\mathbf{b}$ (and $\mathbf{z}_k$) are the entries indexed by I.

Since these computations are performed $(n-m)$-times in the worst case (in each iteration the quantity $\||\mathbf{z}_k| - c\mathbf{1}\|_0$ is reduced by at least one), we obtain the claimed time complexity. □

Lemma A.4. *Let $m \leq n$. For any $\mathbf{z}_0 \in \mathbb{R}^n$ and $\mathbf{A}_0 \in \mathbb{R}^{m\times n}$ with columns in general position, i.e., any m columns of $\mathbf{A}$ span $\mathbb{R}^m$, the accelerated version of Algorithm 1 described in Section 2.3 outputs $\mathbf{z}_{k_{final}}$ in $\mathcal{O}(m^2 n)$ time.*

Proof. Note that there is only one full matrix inversion in Step (1) which costs $\mathcal{O}(m^3)$. In Steps (2)-(5) only the computation of α — complexity $\mathcal{O}(m)$ — and matrix-vector multiplications of dimension m — complexity $\mathcal{O}(m^2)$ — take place. Since these are repeated $(n-m)$-times, the overall complexity is $\mathcal{O}(m^3) + \mathcal{O}(m^2(n-m)) = \mathcal{O}(m^2 n)$. □

Lemma A.5. *Let $m \leq n$. For $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{y} \in \mathbb{R}^n$, the minimization*

$$\min_{\mathbf{z}\in\mathbb{R}^n} \|\mathbf{z}\|_\infty, \qquad \text{s.t. } \mathbf{A}\mathbf{z} = \mathbf{y} \tag{29}$$

can be solved by interior point methods up to accuracy $\delta > 0$ in $\mathcal{O}(n^\omega \log(n/\delta))$ time. Here $\mathcal{O}(n^\omega)$ is the time required to multiply two $n \times n$-matrices, with the best ω known to satisfy $\omega < 2.5$.

Proof. Note that (29) is equivalent to the linear program

$$\min_{\mathbf{z}\in\mathbb{R}^n, u\in\mathbb{R}_{\geq 0}} u \qquad \text{s.t.} \quad \begin{cases} \mathbf{A}\mathbf{z} = \mathbf{y} \\ u\mathbf{1} - \mathbf{z} \geq \mathbf{0} \\ \mathbf{z} + u\mathbf{1} \geq \mathbf{0} \end{cases}. \tag{30}$$

By introducing the auxiliary variables $\mathbf{w}_+ = u\mathbf{1} - \mathbf{z} \in \mathbb{R}^n$ and $\mathbf{w}_- = u\mathbf{1} + \mathbf{z} \in \mathbb{R}^n$, and denoting $\mathbf{w} = (\mathbf{w}_+^\top, \mathbf{w}_-^\top, u)^\top \in \mathbb{R}^{2n+1}$, we can re-write (30) as

$$\min_{\mathbf{w}\in\mathbb{R}^{2n+1}} \mathbf{e}_{2n+1}^\top \mathbf{w} \qquad \text{s.t.} \quad \begin{cases} \tilde{\mathbf{A}}\mathbf{w} = \mathbf{y} \\ \mathbf{w} \geq \mathbf{0} \end{cases}, \tag{31}$$

where $\mathbf{e}_{2n+1}$ is the $(2n+1)$-th unit vector and

$$\tilde{\mathbf{A}} = \begin{pmatrix} -\mathbf{A} & \mathbf{A} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{m\times(2n+1)}.$$

The claim now follows by applying [13, Theorem 1.1] to (31). □

Acknowledgments

RS was supported in part by National Science Foundation Grant DMS-2012546, and by a Simons Fellowship.

References

[1] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems*, 2019.

[2] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCV Workshops*, pages 3009–3018, 2019.

[3] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.

[4] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[6] C Sinan Güntürk and Weilin Li. Approximation of functions with one-bit neural networks. *arXiv preprint arXiv:2112.09181*, 2021.

[7] Yunhui Guo. A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*, 2018.

[8] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[10] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.

[11] Eric Lybrand and Rayan Saab. A greedy algorithm for quantizing neural networks. *Journal of Machine Learning Research*, 22(156):1–38, 2021.

[12] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[13] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 259–278. SIAM, 2020.

[14] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[15] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge university press, 2019.

[16] P Yin, J Lyu, S Zhang, S Osher, YY Qi, and J Xin. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Learning Representations*, 2019.

[17] Jinjie Zhang, Yixuan Zhou, and Rayan Saab. Post-training quantization for neural networks with provable guarantees. *arXiv preprint arXiv:2201.11113*, 2022.

[18] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *International conference on machine learning*, pages 7543–7552. PMLR, 2019.

Department of Mathematics, LMU Munich
and Munich Center for Machine Learning (MCML)
Email address: maly@math.lmu.de

Department of Mathematics and Halıcıoğlu Data Science Institute, University of California San Diego
Email address: rsaab@ucsd.edu