

On the impact of the parametrization of deep convolutional neural networks on post-training quantization

Anonymous authors

Paper under double-blind review

Abstract

This paper introduces novel theoretical approximation bounds for the output of quantized neural networks, with a focus on convolutional neural networks (CNN). By considering layerwise parametrization and focusing on the quantization of weights, we provide bounds that gain several orders of magnitude compared to state-of-the-art results on classical deep convolutional neural networks such as MobileNetV2 or ResNets. These gains are achieved by improving the behaviour of the approximation bounds with respect to the depth parameter, which has the most impact on the approximation error induced by quantization. To complement our theoretical result, we provide a numerical exploration of our bounds on MobileNetV2 and ResNets.

1 Introduction

Neural networks have become central to modern machine learning, driving significant advancements across a wide range of applications, including computer vision, natural language processing, and robotics LeCun et al. (2015); Goodfellow et al. (2016). Due to the size of state-of-the-art models, deploying these in resource-constrained environments, such as mobile devices or embedded systems, requires model compression techniques like pruning Han et al. (2016), low-rank approximations Denton et al. (2014) or quantization Gholami et al. (2022). Post-training quantization, in particular, reduces the bit-width of parameters, enabling faster inference and reduced energy consumption without retraining the model. In this paper, we focus on quantized convolutional neural networks because they provide state-of-the-art performances while remaining lightweight, thus, they remain models of choice for embedded applications. Despite the empirical success of quantized models, one concern is the potential performance degradation introduced by quantization. Establishing theoretical bounds on this degradation is therefore crucial, especially for safety-critical applications where robust guarantees are required Forsberg et al. (2020).

For a neural network R_θ with parameters θ (i.e. a function parametrized by θ), given an approximation θ' of θ , we look for bounds of the form

$$\sup_{x \in \Omega} \|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq C \|\theta - \theta'\|_\infty, \quad (1)$$

where Ω is the domain of the considered inputs of the networks and C is a constant (that must be explicit) depending on the network’s architecture. Such bounds quantify the stability of an architecture with respect to parameter perturbations which is essential for understanding the impact of quantization on performance.

Recent works, such as Gonon et al. (2023), provide insightful approximation bounds for quantization. However, these bounds are often pessimistic for practical use and come with strict assumptions. For example, Gonon et al. (2023) imposes a condition that the maximum parameter norm r must be larger than 1, which limits the applicability of their results, particularly when using post-training quantization where the weights are fixed. In practice, for larger networks, we often regularize weights to prevent overfitting Bejani & Ghatte (2021); Santos & Papa (2022). Techniques such as L2 regularization DropConnect Wan et al. (2013) and weight decay Krogh & Hertz (1991) actively encourage the parameters to remain small, potentially making r smaller than 1. Moreover, when removing the impact of weights and input distribution, the constant C in

equation 1 exhibits a dependency $O(NL^2)$, where L is the depth of the network and N is the width, making this upper bound of little use for modern deep architectures (which have a large L). This opens the following question: is it possible to quantize *deep* neural networks with a practical approximation bound ?

In this work, we provide new theoretical approximation bounds for neural networks, as illustrated in Figure 1. This bound improves several dependencies in the approximation constant C of equation 1 giving a better analysis of the performance of quantized *deep* neural networks in practical cases. We focus on theoretical guarantees for the worst-case quantization error under the *infinity norm* metric, which captures the maximum deviation in network outputs. Our key contributions can be summarized as follows.

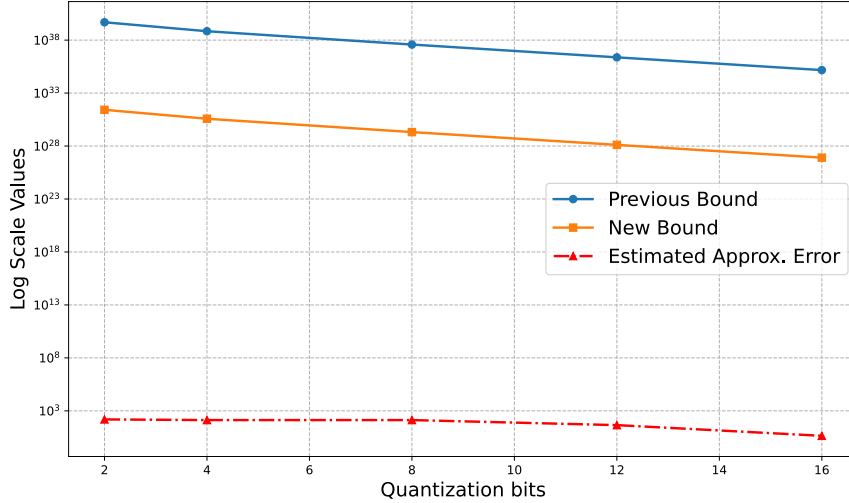


Figure 1: Illustration of the improvement, in log scale, over the previous bound equation 4 on ResNet18 without BatchNorm and without biases, with respect to the number of quantization bits, showing a 10^8 times tighter error estimation.

Tighter approximation bounds: As weights typically require much higher memory usage than biases and many convolutional architectures do not use biases (for the convolutional part), we provide approximation bounds for quantization, when only weights are quantized. In this case, our result enhances the state-of-the-art theorem by Gonon et al. (2023), where we replace their factor of NL^2 with the sum $\sum_{\ell=1}^L N_{\ell-1}$ where N_{ℓ} is the width of the ℓ -th layer, which simplifies to NL when the network has a constant width N . We further improve this constant for convolutional networks where only the filter sizes and the number of channels of each layer replaces the width in the approximation bound.

Relaxation of norm constraints: We weaken the assumptions on the operator norm constraints by considering arbitrary positive values of r_{ℓ} , the ℓ -th layer operator norm. This generalization makes our result applicable to networks where smaller parameter norms are present due to regularization or sparsity constraints. Also, instead of taking the maximum parameter norm r , as in Gonon et al. (2023) and [Corollary F.1]Gonon et al. (2024), our approach replaces it with the less pessimistic expression r_{mean} , which (geometrically) averages the parameter norms across layers.

Practical validation: We validate our theoretical improvements by applying them to classical pretrained CNN models, demonstrating that our approach is orders of magnitude closer to a practical application compared to previous works.

2 Related Works

Approximation bounds have been studied from various perspectives, including results on the approximation capacity of neural networks DeVore et al. (2021); Ding et al. (2019); Csáji et al. (2001); Barron (1993) and the topological properties of the realization map Petersen et al. (2021), particularly focusing on the fact that

this realization is Lipschitz continuous with the constant depending on the network architecture Petersen et al. (2021); DeVore et al. (2021).

There also have been analytical works directly quantifying the value of C in Equation 1. For example, Neyshabur et al. (2018) studied this constant in the context of a particular case where θ' is obtained through controlled perturbation (i.e., perturbations that do not significantly modify the norm of the initial weights). They derived, for the L^2 norm, a constant that depends on the network depth, the norm of the weights, the data and the perturbation. However, their results do not generalize to any θ' and are not applicable to arbitrary quantization. Similarly, Berner et al. (2020) expressed the constant in the case of the L^∞ norm but with uniform parameter bounds and, specifically for $d_{\text{out}} = 1$, which cannot be applied to every neural network tasks.

More recently, Gonon et al. (2023) formalized a framework for approximation bounds of ReLU neural networks, providing a general upper-bound for the constant of a neural network in terms of its architecture, weight norms, and other properties. Specifically, their result applies to neural networks defined over general L^p -spaces, and under general constraints on the weight parameters. The generalization provided by their upper-bound generalizes prior results, which were often limited to specific cases, such as Neyshabur et al. (2018) for spectrally-normalized networks. By the same authors, in Gonon et al. (2024), there is another approach that generalizes the notion of approximation bounds to Directed Acyclic Graphs (DAGs) using ℓ_1 -path norms. This formulation as graphs allows for more flexibility on the network architecture (pooling, skip connection...). This work provides general bounds with the notable feature of being invariant under parameter rescaling. They improve their previous paper results, relaxing some assumptions, notably the condition $r \geq 1$, but introducing new conditions such as $\theta_i \theta'_i \geq 0$, which is not always satisfied for general distinct parameters (θ, θ') . We further discuss these bounds in relation to our work in Section 3.

A direct application of approximation bounds is quantization. Quantization significantly reduces the storage and computational requirements of deep models Gholami et al. (2022). The impact of quantization on the approximation capabilities of neural networks has been studied in Ding et al. (2019) and Hubara et al. (2018), where the authors provided empirical evidence for maintaining high performance even at low precision. However, formal guarantees are still limited, and existing bounds often assume either uniform quantization or specific activation functions, which limit their applicability. Recent works have introduced strategies for low-bit quantization to retain high predictive accuracy in practical implementations. For instance, Choukroun et al. (2019) explored methods for efficient inference with low-bit quantization, while Courbariaux et al. (2015) demonstrated the effectiveness of binary weight quantization during training, opening a way for training and deploying models with significantly reduced memory and computational demands. Another work that explores methods that optimize weight rounding is AdaRound Nagel et al. (2020) which introduces a data-driven adaptive rounding, which learns rounding offsets to preserve layer outputs post-quantization. This approach can be used to further reduce approximation errors. While our theoretical bounds are designed for general quantization mappings, Adaround can be integrated to tighten the term $\|\theta - \theta'\|_\infty$ in practice. Another important aspect in controlling the accuracy of quantized networks involves understanding singular values in convolutional layers Sedghi et al. (2019) which help inform layer-specific quantization strategies.

3 Preliminaries

In this section, we define the key concepts and notations that will be used throughout the paper and we recall reference theoretical approximation bounds from the literature.

Definition 3.1. Neural network architecture. The architecture of a neural network is defined by the tuple (L, \mathbf{N}) , where $L \in \mathbb{N}$ represents the depth of the network, and $\mathbf{N} = (N_0, \dots, N_L) \in \mathbb{N}^{L+1}$ is a sequence specifying the number of neurons in each layer (the width of each). We call N_ℓ the width of the ℓ -th layer. The width of the network is defined as $N := \max_{\ell=0, \dots, L} N_\ell$.

Definition 3.2. Parameters associated with an architecture. Given an architecture (L, \mathbf{N}) , parameters associated with this architecture are $\theta = (\tilde{W}_1, \dots, \tilde{W}_L)$, where $\tilde{W}_\ell \in \mathbb{R}^{N_\ell \times (N_{\ell-1} + 1)}$ is the weight matrix for layer $\ell = 1, \dots, L$ with included bias, i.e. the concatenation of a base weight matrix $W_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ with associated bias $b_\ell \in \mathbb{R}^{N_\ell}$, for layer ℓ . We have that $\theta \in \Theta_{L, \mathbf{N}} := \mathbb{R}^{d(L, \mathbf{N})}$, where the dimension $d(L, \mathbf{N})$ is defined by $d(L, \mathbf{N}) := \sum_{\ell=1}^L N_\ell(N_{\ell-1} + 1)$.

Definition 3.3. ReLU Network. For any vector x , we write $\tilde{x} = \begin{pmatrix} x \\ 1 \end{pmatrix}$. Given an architecture (L, \mathbf{N}) and parameter vector $\theta = (\tilde{W}_1, \dots, \tilde{W}_L)$, we associate the function $R_\theta : \mathbb{R}^{N_0+1} \rightarrow \mathbb{R}^{N_L}$, which is recursively defined for $\ell = 0, \dots, L$ as follows:

$$y_0 = x, \quad y_\ell = \sigma(\tilde{W}_\ell \tilde{y}_{\ell-1}) \quad \text{and} \quad R_\theta(\tilde{x}) = y_L \quad (2)$$

where $\sigma(x) = \max(0, x)$ is the ReLU activation function.

Definition 3.4. Domains for parameters and input vectors. Given an architecture (L, \mathbf{N}) and a parameter space $\Theta_{L, \mathbf{N}}$, for any $r \geq 0$, we define the set of admissible parameters:

$$\Theta_{L, \mathbf{N}}(r) := \{(\tilde{W}_1, \dots, \tilde{W}_L) \in \Theta_{L, \mathbf{N}} : \|\tilde{W}_\ell\|_{\text{op}, \infty} \leq r, \ell = 1, \dots, L\}.$$

where $\|\cdot\|_{\text{op}, \infty}$ denotes the infinity operator norm, defined as follows, for every matrix W in $\mathbb{R}^{m \times n}$:

$$\|W\|_{\text{op}, \infty} := \sup_{x \in \mathbb{R}^n, \|x\|_\infty = 1} \|Wx\|_\infty. \quad (3)$$

In this work, we suppose that the input x belongs to the domain $\Omega = [-D, D]^{N_0}$.

We can now give previous approximation bounds in the ℓ^∞ -norm setting.

Theorem 3.5 (Previous bound from Gonon et al. (2023)). *For any architecture (L, \mathbf{N}) , and any $r \geq 1$, denoting $N := \max_{l=0, \dots, L} N_l$, for any $\theta, \theta' \in \Theta_{L, \mathbf{N}}(r)$, we have :*

$$\sup_{x \in \Omega} \|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq (D+1)NL^2r^{L-1}\|\theta - \theta'\|_\infty. \quad (4)$$

This shows that the quantization error depends on the architecture's depth L , width N , and the maximum norm r of weight matrices. As the network depth L increases, the potential error grows exponentially, increasing the sensitivity of deeper networks to small parameter changes.

In Gonon et al. (2024)[Corollary F.1], a new bound is given as a consequence of a general approach using a path-norm metric (see Section 2):

$$\sup_{x \in \Omega} \|R_\theta(x) - R_{\theta'}(x)\|_1 \leq 2 \max(D, 1)LN^2r^{L-1}\|\theta - \theta'\|_\infty \quad (5)$$

The dependency in L^2 of bound equation 4 is reduced to a dependency in L at the cost of a dependency in N^2 for the ℓ^1 operator norm. For the sake of clarity in our representations, we have chosen to include only the bound equation 4 in our comparisons. This choice is motivated by the fact that it is generally tighter, particularly because most networks of interest tend to have widths significantly larger than their depths (see Table 1 for an illustration). That is, the condition $2LN^2 \gg NL^2$ often holds, making the Gonon et al. (2023) bound more appropriate for practical comparisons in our ℓ^∞ norm setting.

The convolutional case. In convolutional networks for image processing, a convolutional layer can be represented as a matrix multiplication. Consider an input image x of dimensions $n \times m$ and a convolutional filter h of dimensions $p \times p$. The input image x can be flattened into a column vector \mathbf{x} of size $nm \times 1$: $\mathbf{x} = \text{vec}(X)$ where $\text{vec}(\cdot)$ denotes the vectorization operation. The convolution of the input image x by the filter h of size $p \times p$ followed by a subsampling/upsampling can be expressed as a matrix product $\mathbf{y} = \mathcal{H}\mathbf{x}$ where \mathbf{y} of dimensions $n_\ell \times m_\ell$ is the vectorization of the output image and \mathcal{H} is a matrix representing the 2d convolution by h (hence each row contains at most p^2 coefficients). The weight matrix associated with a given convolutional layer is a collection of c_ℓ convolutions by several filters h_ℓ , followed by a subsampling or an upsampling. Hence the width of the ℓ -th layer is $c_{\ell-1} \times n_{\ell-1} \times m_{\ell-1}$.

4 Theoretical results

In this section, we present our main theoretical results, which extend and improve upon the existing bounds for quantized ReLU networks by relaxing the constraints on the network parameters and considering a quantization of the weight matrices only (and not biases). Indeed, in practice, convolutional neural networks such as MobileNetV2 or Resnets without biases are used successfully for vision tasks. If we consider more general networks with biases (and a constant width for the sake of discussion), the size of the bias vector is NL compared to the size of weight matrix N^2L . If the width N of the network is large compared to the objective in terms of memory requirement, e.g. $N \gg 8$ for a $8\times$ memory reduction from 64 bits to 8 bits, then quantifying biases will add little gain in memory compared to the gain resulting from the quantization of weights (this is seen in practice when the biases often remain in full precision or are only lightly quantized Finkelstein et al. (2019)).

We first give a general approximation bound and then specify to the convolutional case.

Theorem 4.1 (General approximation bound). *For any architecture (L, \mathbf{N}) , define the parameters $\theta = (\tilde{W}_1, \dots, \tilde{W}_L)$ and $\theta' = (\tilde{W}'_1, \dots, \tilde{W}'_L)$, where \tilde{W}_ℓ and \tilde{W}'_ℓ are weight matrices with included bias. Assume that the two networks have same biases. Assume besides that $\forall \ell = 1, \dots, L$:*

$$\|\tilde{W}_\ell\|_{\text{op}, \infty} \leq r_\ell \quad \text{and} \quad \|\tilde{W}'_\ell\|_{\text{op}, \infty} \leq r_\ell. \quad (6)$$

Then:

$$\sup_{x \in \Omega} \|R_\theta(\tilde{x}) - R_{\theta'}(\tilde{x})\|_\infty \leq \max(D, 1) \sum_{\ell=1}^L N_{\ell-1} \times r_{\text{mean}}^{L-1} \|\theta - \theta'\|_\infty, \quad (7)$$

where we define the mean norm parameter

$$r_{\text{mean}} := \sqrt[L-1]{\max_{l=1, \dots, L} \max_{i=1, \dots, l-1} \prod_{\substack{j=i \\ j \neq l}}^L r_j}. \quad (8)$$

Note that the maximum norm parameter r in equation 4, from Gonon et al. (2023), is replaced by the term r_{mean} . This geometric mean-like term, which considers the largest of partial products of layer-wise norm parameters, allows for a better adjustment of the bound to the variability of the norms. This variability can be significant, particularly between r_{max} and the other r_l values. Specifically, in the least favorable case where $r_1 = r_2 = \dots = r_L$, the largest product simplifies to r^{L-1} . Conversely, the most favorable scenario would involve a distribution of r_l with high variance, particularly where $\max(r_l) \gg 1$ and all other $r_l \leq 1$. In such cases, r_{mean}^{L-1} would be much smaller than r^{L-1} . We provide a more in depth discussion about this, in Appendix G, with simulated examples in Figure 10.

Remark 4.2 (Improved factor $\sum_{l=1}^L N_{l-1}$). Our result introduces an improved factor, which takes into account the sum of the layer widths. If the architecture has uniform width across all layers, i.e., all layers have width N , this factor becomes $N \times L$, which is smaller than $N \times L^2$ of Equation equation 4.

Remark 4.3 (Weakened condition for the domain of parameters r_ℓ). In this bound, the constants r_ℓ are allowed to be arbitrary positive numbers. This weakens the condition $r \geq 1$ of previous bound, making the result more general and applicable to a wider range of network architectures.

Theorem 4.4 (Approximation bound for CNN). *With the same settings as in Theorem 4.1 and for a purely convolutional network without biases, where each layer applies c_l filters of size $p_l \times p_l$, we have:*

$$\sup_{x \in \Omega} \|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq D \times \sum_{l=1}^L p_l^2 c_{l-1} \times r_{\text{conv}}^{L-1} \|\theta - \theta'\|_\infty \quad (9)$$

where we define

$$r_{\text{conv}} := \sqrt[L-1]{\max_{l=1, \dots, L} \prod_{\substack{k=1 \\ k \neq l}}^L r_k^{\text{conv}}} \quad (10)$$

with r_k^{conv} a bound on the norm of the convolutional matrix of layer k without bias (i.e. $r_k^{conv} \geq \|\mathcal{H}_k\|_{op,\infty}$).

With this formulation, the term $\sum_{\ell=1}^L p_\ell^2 c_{\ell-1}$ accounts for the sparse structure of the convolution matrices \mathcal{H}_ℓ rather than simply using the number of row elements $N_{l-1} = m_{l-1}n_{l-1}c_{l-1}$ ($m_{l-1}n_{l-1}$ being the size of each channel at each layer, which can be much larger, see Table 1). Then notice that we use r_{conv} in this bound, which corresponds to the larger product omitting one term and starting from the first layer, so r_{conv} is smaller than r_{mean} by construction. Finally, we also have an intermediate result between Theorem 4.1 and Theorem 4.4, which corresponds to the case of MLP without bias, given in Appendix A (Theorem A.4).

5 Experiments

In this section, we describe the methodology and setup used to validate our theoretical findings, followed by the results of our experiments, that involve *pretrained* architectures. We evaluate the performance of larger models such as **ResNet18**, **ResNet50**, and **MobileNetV2**, all pretrained on the ImageNet dataset Deng et al. (2009). These models use deep convolutional architectures without biases (except the last fully connected layer) that are frequently used in real-world applications. Note that a particular care must be taken to manage skip connections (see Lemma B.1 and Lemma B.2 in the Annex) in the calculations. For experiments with these networks, we removed the BatchNormalization of the networks to match the conditions of Theorem 4.4. By doing this, we still obtained similar accuracies to models with BatchNorm. Moreover Batch Normalization is not guaranteed to be 1-Lipschitz in general, since its scale and shift parameters are learned during training. One could assume the ℓ -th BN layer is β_ℓ -Lipschitz with $\beta_\ell \geq 1$, which would introduce a factor $\prod_{\ell=1}^L \beta_\ell$ in the final bound. So this would just complicate the analysis and loosen our bound which is already conservative.

The goal of these experiments is to empirically compare the bounds given by our new theoretical results against those derived from the work of Gonon et al. (2023). The pretrained models are also used to evaluate the effects of post-training quantization on model performance. We conduct experiments on the Tiny ImageNet dataset, which contains a total of 110,000 images resized to 64×64 pixels. It includes 100,000 training images, 10,000 test images, and covers 200 distinct classes selected from the original ImageNet. Other experiments on the MNIST and CIFAR-10 datasets leading to similar conclusion are presented in the Annex E.

5.1 Analysis of weight distribution across layers

The results of Figure 2 highlight the key advantage of our theoretical bound equation 9 compared to prior works. In Gonon et al. (2023), the bound depends on the maximum operator norm r , which in these examples, is significantly larger than the "geometric mean" layer-wise term r_{conv} . For example, in ResNet50, the maximum r is approximately 3 times larger than r_{conv} , and in MobileNetV2, it is more than 11 times larger. Specifically for MobileNetV2 we can see that the weight norm distribution looks like the exponential distribution of Figure 10, that is a favorable case to have a tighter bound. Lots of values are small and the maximum value r is more than 100, while r_{conv} is 9. This shows better consideration of network weight norm distribution. In practice, techniques such as Cross-Layer Equalization (CLE) Nagel et al. (2019) can be used as a preprocessing step to homogenize weight distributions across layers for the quantized network. By doing this the value of r_{conv} decreases and leads to tighter bounds in our framework. For example in Figure 11 (in Appendix H), when applying CLE to a 4-bit quantized ResNet50 using the AIMET library Siddegowda et al. (2022), we observed that r_{conv} went from 6 to 4 after CLE. This shows how our theoretical results can explain the benefits of such transformations.

5.2 Quantization

In our experiments, we consider three types of **post-training quantization**. We define *uniform* quantization as a function $Q^{unif} : \Theta_{L,N} \rightarrow \Theta_{L,N}$, where $Q(\theta)$ represents the quantized parameter. Specifically, we use the following definition of uniform quantization in our experiments:

$$Q^{unif}(\theta) = Q_\eta^{unif}(\theta) = \left\lfloor \frac{\theta}{\eta} \right\rfloor \eta, \quad (11)$$

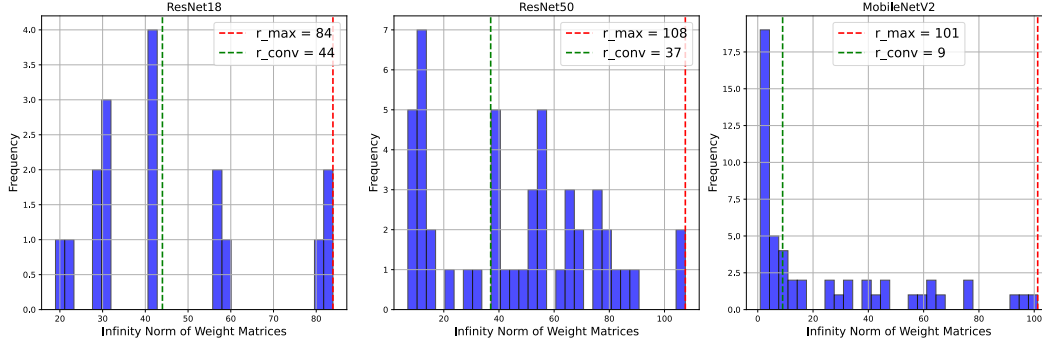


Figure 2: Comparison between the maximum geometric mean term r_{conv} (green) used in equation 9 and the maximum weight norm r (red) used in equation 4, for ResNet18, ResNet50 and MobileNetV2, without BatchNorm, showing a smaller value of r_{conv} for all models.

where $\eta > 0$ is the quantization step size. For each layer, the parameter η is determined by the formula:

$$\eta = \frac{W_{\max}}{2^n - 1},$$

where W_{\max} is the maximum absolute value of the weight values, and n is the bit width.

To highlight the role of the term $\|\theta - Q(\theta)\|_{\infty}$ in the quantization error, as an alternative approach to uniform quantization, we define Q^{round} using a rounding function instead of floor truncation. This form of quantization is defined as:

$$Q^{\text{round}}(\theta) = Q_{\eta}^{\text{round}}(\theta) = \text{round}\left(\frac{\theta}{\eta}\right)\eta, \quad (12)$$

where the function round rounds $\frac{\theta}{\eta}$ to the nearest integer before scaling back by η . This approach may reduce quantization error in cases where uniform quantization leads to excessive loss of information.

In the same way, we also consider an adaptive quantization approach, which is a simplified AdaRound scheme. AdaRound optimizes a per-parameter rounding offset via a differentiable relaxation of the binary rounding decision. More precisely, for a given parameter θ , our adaptive quantization is defined as:

$$Q^{\text{adaround}}(\theta) = \left(\left\lfloor \frac{\theta}{\eta} \right\rfloor + \sigma(\alpha)\right)\eta, \quad (13)$$

where α is a learnable offset, and $\sigma(\cdot)$ denotes the sigmoid function. The offset α is optimized over a calibration dataset by minimizing the mean squared error between the FP32 output and the quantized one, with an additional regularization term that encourages $\sigma(\alpha)$ to converge to 0.5 (i.e., standard rounding behavior). This approach better adjusts the quantization with the distribution of the weights.

5.3 Experimental results on MobileNetV2 and Resnets

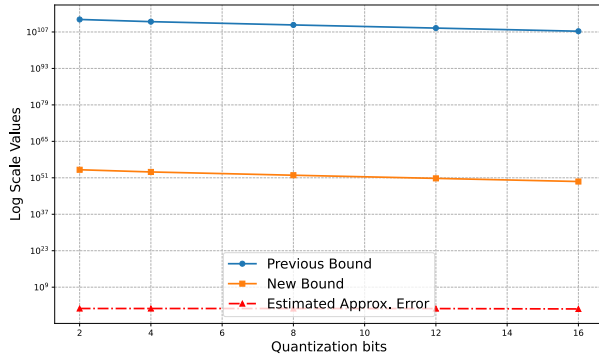
We analyze key parameters involved in our bounds and those from Gonon et al. (2023) in Table 1. The depth L of the considered architectures varies between $L = 18$ for ResNet18 and $L = 53$ for MobileNetV2. This variation in depth is important, as the approximation bound is exponentially dependent on L . An important difference is observed in the width parameter N , where the values obtained using the formulation of Theorem 4.4 bound are orders of magnitude smaller than those from the previous bound. For instance, for MobileNetV2, the previous bound is calculated with a width of 1.2×10^6 , whereas the new bound reduces this to 8641. Similarly, for ResNet50, the width decreases from 8×10^5 to 4609. This significant decrease reflects the tighter characterization provided by the new bound, which avoids overly conservative estimations of N . Another critical parameter is the maximum norm parameter, which is significantly smaller under the new bound, particularly for MobileNetV2 ($r \approx 101$) while r_{mean} only equals to 9. The reduced values of

Table 1: Comparison of parameters between our bound equation 9 and the state-of-the-art Gonon et al. (2023) bound on pre-trained models. The comparison is also expressed in terms of a ratio of the two bounds, where the values of the bounds in this ratio are computed exclusively for the convolutional part of the network.

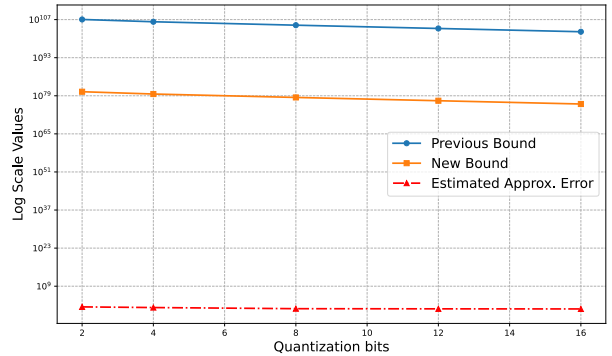
| MODEL | DEPTH (L) | PREVIOUS WIDTH | PREVIOUS NORM PARAM | NEW WIDTH | NEW NORM PARAM | RATIO |
|-------------|---------------|-------------------------|---------------------|---------------------|----------------|---|
| | | $m_{l-1}n_{l-1}c_{l-1}$ | r | $p_{l-1}^2 c_{l-1}$ | r_{conv} | $\frac{\text{PREVIOUS BOUND (2023)}}{\text{NEW BOUND}}$ |
| MOBILENETV2 | 53 | 1.2×10^6 | ≈ 101 | 8641 | ≈ 9 | $\approx 10^{56}$ |
| RESNET18 | 18 | 8×10^5 | ≈ 84 | 4609 | ≈ 44 | $\approx 10^8$ |
| RESNET50 | 50 | 8×10^5 | ≈ 108 | 4609 | ≈ 37 | $\approx 10^{27}$ |

norm parameters reduce the exponential dependency on depth, which is the main pessimistic factor in the bound. These differences between the two bounds is reflected in the ratio value. Even for a shallow network like ResNet18, we notice in Table 1 that our bound is 10^8 times tighter and this observation becomes even more relevant for deeper and wider networks such as MobileNetV2, where the ratio reaches 10^{56} .

In Figure 1 (in the introduction), we compare the approximation error bound and the estimated approximation error for the Resnet18 architecture. The results were obtained by removing the final fully connected layer to keep only the convolutional part of the network. Then we use r_{conv} to calculate the new bound since the network has no biases for the convolutional part. The figure demonstrates an error of approximately 10^{40} , whereas the new bound is only 10^{32} . This highlights a significant improvement, even for a shallow network (with $L = 18$). In Figure 3a, we perform the same experiment for the MobileNetV2 architecture. The effect of weight norm distribution is even more apparent. The previous bound reaches a value of 10^{109} , whereas the new bound is reduced to 10^{53} . For both figures, we can observe that the shape of the bound is accurate as it follows the error approximation trend, up to a constant factor. Nevertheless, the constant is still large even for the new bound, since the output error is at most around 10^3 .



(a) MobileNetV2 without BatchNorm and biases.



(b) ResNet50 without BatchNorm and biases.

Figure 3: Comparison in log scale between our bound equation 9 and the previous bound equation 4, on the convolutional part of (a) MobileNetV2 and (b) ResNet50, with respect to the number of bits. Our bound is approximately 10^{56} times tighter for MobileNetV2 and 10^{27} times tighter for ResNet50.

We can make similar observations from Figure 3b because the previous bound has the same order of magnitude. However, even though MobileNetV2 and ResNet50 have similar depths, our bound has adapted much better to the specific distribution of weight norms in MobileNetV2. The bound value for MobileNetV2 is significantly lower than the ResNet50 one, being around 10^{26} times smaller. This is mainly due to the favorable distribution of weight norms in MobileNetV2, which favors r_{conv} to remain small. Although our new bound significantly improves the previous state of the art bound (e.g., Gonon et al. (2023), [Corollary F.1]Gonon et al. (2024)), the new bound is still orders of magnitude far from what is observed in real outputs of networks (Figures 1, 3a and 3b). One reason is that such bounds are derived under theoretical worst-case scenarios,

which rarely occur in practice. Another reason is the generality of the theorem itself, which leads to a very conservative estimate of the output error.

In Figure 4 we analyze the impact of post-training quantization on the accuracy of MobileNetsV2 and ResNets, across Tiny Imagenet. We notice that the Adaround method gives similar performances to the round method, except for ResNet18 where the behavior for extreme quantization (≤ 4 bits) is better. Indeed for ResNet18 the accuracy with 4 bit quantization is over 30% with the AdaRound method, while it is below 10% accuracy with floor and round methods. Finally, it appears that MobileNetV2 better support quantization with floor method than ResNets. This is probably due to the specific architecture of MobileNetV2 that uses residual bottlenecks and Relu6 activation function ($Relu6(x) := \min(\max(0, x), 6)$) which is known to better support quantization Sandler et al. (2018). Furthermore, the dataset influences precision as well, that is why we conducted similar experiments across MNIST LeCun (1998) and CIFAR-10 Krizhevsky et al. (2009) in which we can also observe the behavior reflected by the form of the bound: depth significantly influences quantization errors (see Figure 9 in the Appendix E).

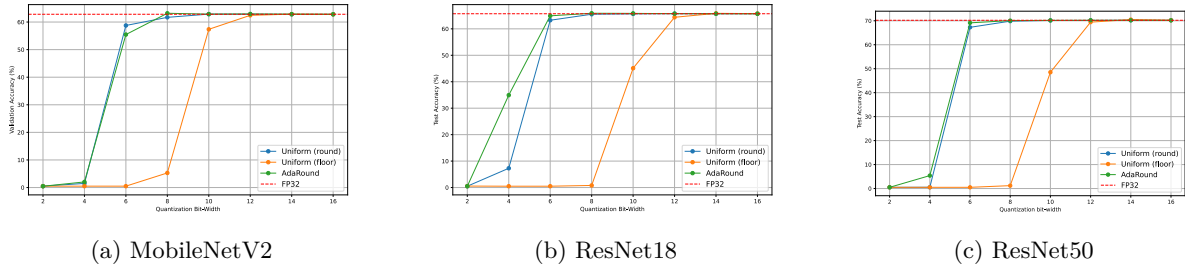


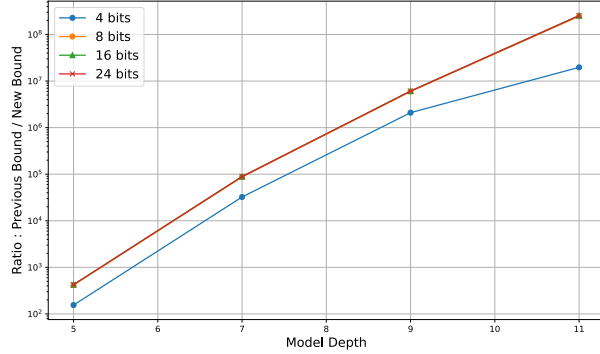
Figure 4: Graphs illustrating the effect of quantization on performance on Tiny ImageNet for three quantization functions (round, uniform and Adaround). The results highlight how quantization reduces memory requirements while maintaining or approaching the base model’s accuracy. The amount of quantization needed to reach the base precision depends on the quantization function used.

5.4 Effect of depth in the quantization of MLP without biases

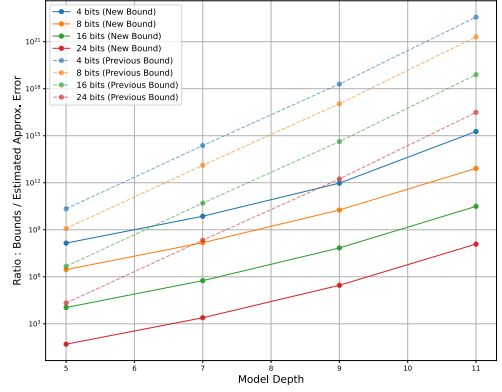
In Figure 5a, we investigate the improvement in accuracy of our bound in Theorem 4.1 with respect to depth. We construct four MLPs of depths: 5, 7, 9, and 11, trained on the MNIST dataset with architectures provided in the Appendix C. Independently of the number of bits, the ratio value is dependent on the depth, starting from $\approx 10^3$ for depth 5 to $\approx 10^8$ for depth 11. Additionally, it is observed that for extreme quantization (4 bits), there is a notable difference compared to other quantizations (8, 16, and 24 bits). Indeed, for 4 bits, the model accuracy is bad ($\leq 10\%$), impacting the norms of the weight matrices and making r_{mean} closer to r_{max} . This results in a slightly lower ratio compared to other quantization levels. Finally, in Figure 5b, we compare the bounds with respect to the approximation error. First, the slope of the curves does not change. Moreover, the exponential dependence on depth is significantly reduced. Indeed, the slope given by r for the previous bound is much steeper than the slope given by r_{mean} in our bound. Thus, the new bound better adapts to the network’s behavior, up to a constant. However, the ideal slope, which would perfectly match the network’s behavior, would be a horizontal curve. We extended our experiments on MLPs by replacing the ReLU activation with a sigmoid function implemented with the `tanh`. This function satisfies our hypotheses and the experimental results show similar trends to those obtained with ReLU, showing that our approach is robust to different activation functions (see Figure 12 in Appendix I).

6 Conclusion and Perspectives

In this work, we introduced a novel theoretical approximation bound for deep (convolutional) neural networks. By generalizing the infinity-norm-based bound and introducing a more flexible approach to operator norm constraints, our bound significantly improves existing ones for a broad range of network architectures. This was validated on popular architectures such as ResNet18, ResNet50, and MobileNetV2, as well as MLPs without



(a) Ratio in log scale between the previous bound equation 4 and our bound equation 7 as a function of model depth for different quantization bit-widths.



(b) Ratio in log scale between bounds equation 4, equation 7 and the estimated error approximation, as a function of model depth for different quantization bit-widths.

Figure 5: Comparison for MLPs of depths 5, 7, 9 and 11 on MNIST. (a) shows how the ratio of our bound over the previous bound grows exponentially with depth, and (b) demonstrates that our bound reduces that exponential dependence across bit-widths.

bias quantization; showcasing improved accuracy in performance predictions for post-training quantized networks. The present study is focused on CNNs, and extensions to other architectures are beyond the scope of this paper. It is nevertheless clear that extending our theoretical framework to Vision Transformers (Dosovitskiy et al., 2020) would be highly interesting for future work. Such an extension would require a careful treatment of specific Transformers block, including layer normalization and multi-head attention.

As for layer normalization, it is not globally Lipschitz due to its input-dependent scaling. However, after training, one can empirically estimate a Lipschitz constant L_{LN} for each layer, which can then be integrated into our framework in the same way as batch normalization, as discussed in Section 5.

Regarding multi-head attention, the main difficulty lies in bounding the softmax operation used in scaled dot-product attention:

$$\text{Att}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V.$$

While softmax is not globally 1-Lipschitz, it is locally Lipschitz over bounded domains. A local Lipschitz constant L_{soft} can be extracted depending on the norm of the inputs, and included in the bound. Recent work (Castin et al., 2023) provides precise estimates for this constant. The concatenation of multiple attention heads can be handled by bounding each head independently and summing their contributions, although this may result in pessimistic bounds. Skip connections can be addressed on a case-by-case basis depending on the model architecture, as we did for CNNs.

Another important and promising lead lies in developing a probabilistic approach that reflects the network’s expected behavior under typical operating conditions, such as average or quantile-based performance, providing a complementary view to our deterministic bound. While a deterministic bound is crucial for critical applications, probabilistic insights can enhance our understanding of the network’s behavior in practical, real-world scenarios.

References

- Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.
- Mohammad Mahdi Bejani and Mehdi Ghaee. A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, 54(8):6391–6438, 2021.

- Julius Berner, Philipp Grohs, and Arnulf Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black–scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020.
- Valérie Castin, Pierre Ablin, and Gabriel Peyré. How smooth is attention? *arXiv preprint arXiv:2312.14820*, 2023.
- Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 3009–3018. IEEE, 2019.
- Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.
- Balázs Csanád Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Eötvös Loránd University, Hungary*, 24(48):7, 2001.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.
- Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pp. 1269–1277, 2014.
- Ronald DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation. *Acta Numerica*, 30: 327–444, 2021.
- Yukun Ding, Jinglan Liu, Jinjun Xiong, and Yiyu Shi. On the universal approximability and complexity bounds of quantized relu neural networks. In *International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, 2019.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Alexander Finkelstein, Uri Almog, and Mark Grobman. Fighting quantization bias with bias. *arXiv preprint arXiv:1906.03193*, 2019.
- Håkan Forsberg, Joakim Lindén, Johan Hjorth, Torbjörn Månefjord, and Masoud Daneshtalab. Challenges in using neural networks in safety-critical applications. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pp. 1–7. IEEE, 2020.
- Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pp. 291–326. Chapman and Hall/CRC, 2022.
- Antoine Gonon, Nicolas Brisebarre, Rémi Gribonval, and Elisa Riccietti. Approximation speed of quantized vs. unquantized relu neural networks and beyond. *IEEE Transactions on Information Theory*, 2023.
- Antoine Gonon, Nicolas Brisebarre, Elisa Riccietti, and Rémi Gribonval. Path-metrics, pruning, and generalization. *arXiv preprint arXiv:2405.15006*, 2024.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations (ICLR)*, 2016.

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30, 2018.
- Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.
- Yann LeCun. The MNIST database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998. Accessed: 2025-05-06.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1325–1334, 2019.
- Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*, pp. 7197–7206. PMLR, 2020.
- Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *International Conference on Learning Representations (ICLR)*, 2018.
- Philipp Petersen, Mones Raslan, and Felix Voigtländer. Topological properties of the set of functions generated by neural networks of fixed size. *Foundations of computational mathematics*, 21:375–444, 2021.
- Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.
- Claudio Filipi Gonçalves Dos Santos and João Paulo Papa. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. *ACM Computing Surveys (CSUR)*, 54(10s):1–25, 2022.
- Hanie Sedghi, Vineet Gupta, and Philip M Long. The singular values of convolutional layers. *International Conference on Learning Representations (ICLR)*, 2019.
- Sangeetha Siddegowda, Marios Fournarakis, Markus Nagel, Tijmen Blankevoort, Chirag Patel, and Abhijit Khobare. Neural network quantization with ai model efficiency toolkit (aimet). *arXiv preprint arXiv:2201.08442*, 2022.
- Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066. PMLR, 2013.

A Proof of main results

For completeness, we begin by recalling the value of the infinity norm of a matrix.

Lemma A.1. *For all matrices $A \in \mathbb{R}^{m \times n}$ and all $x \in \mathbb{R}^n$ we have:*

$$\|A\|_{\text{op},\infty} := \sup_{x \in \mathbb{R}^n, \|x\|_\infty=1} \|Ax\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{i,j}| \quad (14)$$

Proof. Let $A \in \mathbb{R}^{m \times n}$ and $x \in \mathbb{R}^n$ with $\|x\|_\infty = 1$, then

$$\|Ax\|_\infty = \max_{1 \leq i \leq m} |(Ax)_i| = \max_{1 \leq i \leq m} \left| \sum_{j=1}^n a_{i,j} x_j \right| \leq \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{i,j}| \|x\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{i,j}| \quad (15)$$

To show that equality holds, let x such that $x_j = \text{sign}(a_{i^*,j})$ where $i^* \in \arg \max_i \sum_{j=1}^n |a_{i,j}|$. Then $\|x\|_\infty = 1$ and:

$$\|Ax\|_\infty = \sum_{j=1}^n a_{i^*,j} \cdot \text{sign}(a_{i^*,j}) = \sum_{j=1}^n |a_{i^*,j}| = \max_i \sum_{j=1}^n |a_{i,j}|. \quad (16)$$

This shows that:

$$\|A\|_{\text{op},\infty} = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{i,j}|. \quad (17)$$

□

The following Lemma bounds the difference of outputs between a network and its quantization. It is adapted from [Lemma C.1]Gonon et al. (2023) to weight matrices with included identical biases.

Lemma A.2. *Let (L, \mathbf{N}) be an architecture with $L \geq 1$, denoting $\theta = (\tilde{W}_1, \dots, \tilde{W}_L)$, $\theta' = (\tilde{W}'_1, \dots, \tilde{W}'_L) \in \Theta_{L,\mathbf{N}}$ as two sets of parameters associated with this architecture, with each last column of the matrices composed by biases of the corresponding layer. We assume that the two networks have same biases (i.e. we do not quantize the bias) For every $\ell = 1, \dots, L-1$, define θ'_ℓ as the parameter deduced from θ' , associated with the architecture $(\ell, (N_0, \dots, N_\ell))$:*

$$\theta'_\ell = (\tilde{W}'_1, \dots, \tilde{W}'_\ell).$$

Then for every $\tilde{x} = \begin{pmatrix} x \\ 1 \end{pmatrix}$ with $x \in \mathbb{R}^{N_0}$, denoting by W_k and W'_k the weight matrices without biases (i.e. \tilde{W}_k and \tilde{W}'_k with last column removed), then, for any 1-Lipschitz activation function σ , we have:

$$\|R_\theta(\tilde{x}) - R_{\theta'}(\tilde{x})\|_\infty \leq \sum_{\ell=1}^L \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} \right) \|W_\ell - W'_\ell\|_{\text{op},\infty} \|R_{\theta'_{\ell-1}}(\tilde{x})\|_\infty, \quad (18)$$

where we set by convention $R_{\theta'_{\ell-1}}(\tilde{x}) = x$ for $\ell = 1$, and $\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} = 1$ for $\ell = L$.

Proof. The proof of Inequality equation 18 follows by induction on $L \in \mathbb{N}$. For $L = 1$, with the Lipschitz condition and the definition of $\|\cdot\|_{\text{op},\infty}$, using the fact that the last columns of \tilde{W}_1 and \tilde{W}'_1 are equal, we have $\|\tilde{W}_1 \tilde{x} - \tilde{W}'_1 \tilde{x}\|_\infty = \|W_1 x - W'_1 x\|_\infty$ and:

$$\begin{aligned} \|R_{\theta_L}(\tilde{x}) - R_{\theta'_L}(\tilde{x})\|_\infty &= \|\sigma(\tilde{W}_1 \tilde{x}) - \sigma(\tilde{W}'_1 \tilde{x})\|_\infty \leq \|\tilde{W}_1 \tilde{x} - \tilde{W}'_1 \tilde{x}\|_\infty \\ &= \|W_1 \tilde{x} - W'_1 \tilde{x}\|_\infty \\ &\leq \|W_1 - W'_1\|_{\text{op},\infty} \|x\|_\infty. \end{aligned} \quad (19)$$

Now assume that property equation 18 holds for $L \geq 1$. At rank $L + 1$, using the fact that the activation function σ is 1-Lipschitz, we have :

$$\begin{aligned}
& \left\| R_{\theta_{L+1}}(\tilde{x}) - R_{\theta'_{L+1}}(\tilde{x}) \right\|_{\infty} \\
&= \left\| \sigma \left(\tilde{W}_{L+1} \begin{pmatrix} R_{\theta_L}(\tilde{x}) \\ 1 \end{pmatrix} \right) - \sigma \left(\tilde{W}'_{L+1} \begin{pmatrix} R_{\theta'_L}(\tilde{x}) \\ 1 \end{pmatrix} \right) \right\|_{\infty} \\
&\leq \left\| \tilde{W}_{L+1} \begin{pmatrix} R_{\theta_L}(\tilde{x}) \\ 1 \end{pmatrix} - \tilde{W}'_{L+1} \begin{pmatrix} R_{\theta'_L}(\tilde{x}) \\ 1 \end{pmatrix} \right\|_{\infty} \\
&= \left\| \tilde{W}_{L+1} \left(\begin{pmatrix} R_{\theta_L}(\tilde{x}) \\ 1 \end{pmatrix} - \begin{pmatrix} R_{\theta'_L}(\tilde{x}) \\ 1 \end{pmatrix} \right) + (\tilde{W}_{L+1} - \tilde{W}'_{L+1}) \begin{pmatrix} R_{\theta'_L}(\tilde{x}) \\ 1 \end{pmatrix} \right\|_{\infty} \\
&\leq \left\| \tilde{W}_{L+1} \begin{pmatrix} R_{\theta_L}(\tilde{x}) - R_{\theta'_L}(\tilde{x}) \\ 0 \end{pmatrix} \right\|_{\infty} + \left\| (\tilde{W}_{L+1} - \tilde{W}'_{L+1}) \begin{pmatrix} R_{\theta'_L}(\tilde{x}) \\ 1 \end{pmatrix} \right\|_{\infty} \\
&= \left\| W_{L+1} \begin{pmatrix} R_{\theta_L}(\tilde{x}) - R_{\theta'_L}(\tilde{x}) \\ 0 \end{pmatrix} \right\|_{\infty} + \left\| (W_{L+1} - W'_{L+1}) R_{\theta'_L}(\tilde{x}) \right\|_{\infty} \\
&\leq \|W_{L+1}\|_{\text{op},\infty} \left\| R_{\theta_L}(\tilde{x}) - R_{\theta'_L}(\tilde{x}) \right\|_{\infty} + \|W_{L+1} - W'_{L+1}\|_{\text{op},\infty} \left\| R_{\theta'_L}(\tilde{x}) \right\|_{\infty}.
\end{aligned}$$

$\left. \begin{array}{l} \sigma \text{ is 1-Lipschitz} \\ \text{triangle inequality} \\ \text{same last column} \\ \text{for } \tilde{W}_{L+1} \text{ and } \tilde{W}'_{L+1} \\ \text{Sub-multiplicativity} \end{array} \right\}$

Applying the induction hypothesis (Inequality equation 18) to the term $\left\| R_{\theta_L}(\tilde{x}) - R_{\theta'_L}(\tilde{x}) \right\|_{\infty}$, we have:

$$\left\| R_{\theta_L}(\tilde{x}) - R_{\theta'_L}(\tilde{x}) \right\|_{\infty} \leq \sum_{\ell=1}^L \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} \right) \|W_{\ell} - W'_{\ell}\|_{\text{op},\infty} \left\| R_{\theta'_{\ell-1}}(\tilde{x}) \right\|_{\infty}. \quad (20)$$

Substituting this bound back into the previous inequality and using that we have $\prod_{k=L+2}^{L-1} \|W_k\|_{\text{op},\infty} = 1$ by convention, we get:

$$\begin{aligned}
\left\| R_{\theta_{L+1}}(\tilde{x}) - R_{\theta'_{L+1}}(\tilde{x}) \right\|_{\infty} &\leq \|W_{L+1}\|_{\text{op},\infty} \sum_{\ell=1}^L \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} \right) \|W_{\ell} - W'_{\ell}\|_{\text{op},\infty} \left\| R_{\theta'_{\ell-1}}(\tilde{x}) \right\|_{\infty} \\
&\quad + \|W_{L+1} - W'_{L+1}\|_{\text{op},\infty} \left\| R_{\theta'_L}(\tilde{x}) \right\|_{\infty} \\
&= \sum_{\ell=1}^L \left(\prod_{k=\ell+1}^{L+1} \|W_k\|_{\text{op},\infty} \right) \|W_{\ell} - W'_{\ell}\|_{\text{op},\infty} \left\| R_{\theta'_{\ell-1}}(\tilde{x}) \right\|_{\infty} \\
&\quad + \left(\prod_{k=L+1+1}^{L+1} \|W_k\|_{\text{op},\infty} \right) \|W_{L+1} - W'_{L+1}\|_{\text{op},\infty} \left\| R_{\theta'_L}(\tilde{x}) \right\|_{\infty}.
\end{aligned} \quad (21)$$

We deduce

$$\left\| R_{\theta_{L+1}}(\tilde{x}) - R_{\theta'_{L+1}}(\tilde{x}) \right\|_{\infty} \leq \sum_{\ell=1}^{L+1} \left(\prod_{k=\ell+1}^{L+1} \|W_k\|_{\text{op},\infty} \right) \|W_{\ell} - W'_{\ell}\|_{\text{op},\infty} \left\| R_{\theta'_{\ell-1}}(\tilde{x}) \right\|_{\infty}. \quad (22)$$

This concludes the induction and proves the lemma. □

Note that, if we suppose that our networks have no bias, then, for all layers, we do not have to take account of the last column of weight matrices with included bias. We can do the same proof replacing \tilde{W}_l by W_l and \tilde{x} by x .

The result without bias also follows by the same induction, and it comes:

$$\|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq \sum_{\ell=1}^L \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} \right) \|W_\ell - W'_\ell\|_{\text{op},\infty} \|R_{\theta'_{\ell-1}}(x)\|_\infty. \quad (23)$$

The following lemma allows us to upper bound the output of a network based on the operator norms of each layer, without any specific conditions on their values.

Lemma A.3. *Let (L, \mathbf{N}) be an architecture with $L \geq 1$, denoting $\theta = (\tilde{W}_1, \dots, \tilde{W}_L) \in \Theta_{L, \mathbf{N}}$ a set of parameters associated with this architecture, adding the assumption $\sigma(0) = 0$. Then for every $\tilde{x} = \begin{pmatrix} x \\ 1 \end{pmatrix}$ where $x \in \mathbb{R}^{N_0}$, we have:*

$$\|R_{\theta_L}(\tilde{x})\|_\infty \leq \max \left(\max_{l=2, \dots, L} \prod_{s=\ell}^L \|\tilde{W}_s\|_{\text{op},\infty}; \prod_{s=1}^L \|\tilde{W}_s\|_{\text{op},\infty} \|\tilde{x}\|_\infty \right). \quad (24)$$

Proof. We prove equation 24 by induction on $L \in \mathbb{N}$. For $L = 1$, since σ is 1-Lipschitz and satisfies $\sigma(0) = 0$, we have:

$$\|R_{\theta_1}(\tilde{x})\|_\infty = \|\sigma(\tilde{W}_1 \tilde{x}) - \sigma(0)\|_\infty \leq \|\tilde{W}_1 \tilde{x} - 0\|_\infty = \|\tilde{W}_1 \tilde{x}\|_\infty \leq \|\tilde{W}_1\|_{\text{op},\infty} \|\tilde{x}\|_\infty. \quad (25)$$

Then by convention for $L = 1$ (as in Lemma A.2),

$$\max_{l=2, \dots, L} \prod_{s=\ell}^L \|\tilde{W}_s\|_{\text{op},\infty} = 1. \quad (26)$$

We deduce:

$$\|R_{\theta_L}(\tilde{x})\|_\infty \leq \|\tilde{W}_1\|_{\text{op},\infty} \|\tilde{x}\|_\infty \leq \max(1; \|\tilde{W}_1\|_{\text{op},\infty} \|\tilde{x}\|_\infty). \quad (27)$$

Now assume that the property holds for $L \geq 1$. At rank $L + 1$, using that operator norm is sub-multiplicative and the fact that σ is 1-Lipschitz and satisfies $\sigma(0) = 0$, we have:

$$\begin{aligned} \|R_{\theta_{L+1}}(\tilde{x})\|_\infty &= \left\| \sigma \left(\tilde{W}_{L+1} \begin{pmatrix} R_{\theta_L}(\tilde{x}) \\ 1 \end{pmatrix} \right) \right\|_\infty \\ &\leq \|\tilde{W}_{L+1}\|_{\text{op},\infty} \left\| \begin{pmatrix} R_{\theta_L}(\tilde{x}) \\ 1 \end{pmatrix} \right\|_\infty \\ &= \left\| \begin{pmatrix} \|\tilde{W}_{L+1}\|_{\text{op},\infty} \|R_{\theta_L}(\tilde{x})\|_\infty \\ \|\tilde{W}_{L+1}\|_{\text{op},\infty} \end{pmatrix} \right\|_\infty \end{aligned} \quad (28)$$

Then, applying the induction hypothesis to the term $\|R_{\theta_L}(\tilde{x})\|_\infty$, it comes:

$$\begin{aligned}
\left\| \left(\frac{\|\tilde{W}_{L+1}\|_{\text{op},\infty} \|R_{\theta_L}(\tilde{x})\|_\infty}{\|\tilde{W}_{L+1}\|_{\text{op},\infty}} \right) \right\|_\infty &= \max \left(\|\tilde{W}_{L+1}\|_{\text{op},\infty} \|R_{\theta_L}(\tilde{x})\|_\infty; \|\tilde{W}_{L+1}\|_{\text{op},\infty} \right) \\
&\leq \max \left(\|\tilde{W}_{L+1}\|_{\text{op},\infty} \cdot \max \left[\max_{l=2,\dots,L} \prod_{s=l}^L \|\tilde{W}_s\|_{\text{op},\infty}; \right. \right. \\
&\quad \left. \left. \prod_{s=1}^L \|\tilde{W}_s\|_{\text{op},\infty} \|\tilde{x}\|_\infty \right]; \|\tilde{W}_{L+1}\|_{\text{op},\infty} \right) \\
&= \max \left(\max \left[\max_{l=2,\dots,L} \prod_{s=l}^{L+1} \|\tilde{W}_s\|_{\text{op},\infty}; \right. \right. \\
&\quad \left. \left. \prod_{s=1}^{L+1} \|\tilde{W}_s\|_{\text{op},\infty} \|\tilde{x}\|_\infty \right]; \|\tilde{W}_{L+1}\|_{\text{op},\infty} \right) \\
&= \max \left[\max_{l=2,\dots,L} \prod_{s=l}^{L+1} \|\tilde{W}_s\|_{\text{op},\infty}; \right. \\
&\quad \left. \prod_{s=1}^{L+1} \|\tilde{W}_s\|_{\text{op},\infty} \|\tilde{x}\|_\infty; \|\tilde{W}_{L+1}\|_{\text{op},\infty} \right]
\end{aligned} \tag{29}$$

Then, noticing that the term $\|\tilde{W}_{L+1}\|_{\text{op},\infty}$ can be included in $\max_{l=2,\dots,L} \left(\prod_{s=l}^{L+1} \|\tilde{W}_s\|_{\text{op},\infty} \right)$ by adding index $\ell = L + 1$, we have

$$\|R_{\theta_{L+1}}(\tilde{x})\|_\infty \leq \max \left[\max_{l=2,\dots,L+1} \left(\prod_{s=l}^{L+1} \|\tilde{W}_s\|_{\text{op},\infty} \right); \prod_{s=1}^{L+1} \|\tilde{W}_s\|_{\text{op},\infty} \|\tilde{x}\|_\infty \right]. \tag{30}$$

This concludes the induction and proves the lemma. □

We can now prove our main theorem.

Proof of Theorem 4.1. We recall that θ'_ℓ is defined as the parameter deduced from θ' , associated with the architecture $(\ell, (N_0, \dots, N_\ell))$.

With the convention that

$$\begin{cases} R_{\theta'_{\ell-1}}(\tilde{x}) = \tilde{x} & \text{if } \ell = 1, \\ \prod_{k=\ell+1}^L \|W_k\|_{\text{op},q} = 1 & \text{if } \ell = L, \end{cases} \tag{31}$$

we have, with Lemma A.2:

$$\|R_\theta(\tilde{x}) - R_{\theta'}(\tilde{x})\|_\infty \leq \sum_{\ell=1}^L \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} \right) \|W_\ell - W'_\ell\|_{\text{op},\infty} \|R_{\theta'_{\ell-1}}(\tilde{x})\|_\infty. \tag{32}$$

Thus, using Lemma A.3 we can bound $\|R_{\theta'_{\ell-1}}(\tilde{x})\|_\infty$, with the convention that an empty product is equal to 1, it follows:

$$\|R_{\theta'_{\ell-1}}(\tilde{x})\|_{\infty} \leq \max \left[\max_{i=2,\dots,\ell-1} \left(\prod_{s=i}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty} \right); \prod_{s=1}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty} \|\tilde{x}\|_{\infty} \right] \quad (33)$$

Then, noticing that $\|\tilde{x}\|_{\infty} \geq 1$, and re-indexing the second max to include $\prod_{s=1}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty}$, we have:

$$\begin{aligned} \|R_{\theta'_{\ell-1}}(\tilde{x})\|_{\infty} &\leq \max \left[\max_{i=2,\dots,\ell-1} \left(\prod_{s=i}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty} \right) \|\tilde{x}\|_{\infty}; \prod_{s=1}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty} \|\tilde{x}\|_{\infty} \right] \\ &= \|\tilde{x}\|_{\infty} \max \left[\max_{i=2,\dots,\ell-1} \left(\prod_{s=i}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty} \right); \prod_{s=1}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty} \right] \\ &= \|\tilde{x}\|_{\infty} \max_{i=1,\dots,\ell-1} \left(\prod_{s=i}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty} \right). \end{aligned} \quad (34)$$

Using this bound in Equation equation 32, we get:

$$\|R_{\theta}(\tilde{x}) - R_{\theta'}(\tilde{x})\|_{\infty} \leq \|\tilde{x}\|_{\infty} \sum_{\ell=1}^L \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} \right) \|W_{\ell} - W'_{\ell}\|_{\text{op},\infty} \max_{i=1,\dots,\ell-1} \left(\prod_{s=i}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty} \right). \quad (35)$$

Then, recalling that $\tilde{x} = \begin{pmatrix} x \\ 1 \end{pmatrix}$ with $x \in [-D, D]^d$, we have

$$\|\tilde{x}\|_{\infty} \leq \max(D; 1) \quad (36)$$

Then, we know by Lemma A.1 that for every matrix in $\mathbb{R}^{m \times n}$:

$$\|W\|_{\text{op},\infty} = \max_{1 \leq i \leq m} \sum_{j=1}^n |w_{ij}| \quad (37)$$

Thus, recalling that $\theta = (\tilde{W}_1, \dots, \tilde{W}_L)$ and because for all l , $\dim(W_l) = N_l \times N_{l-1}$, we can write:

$$\|W_l\|_{\text{op},\infty} \leq N_{l-1} \max_{i,j} |(W_l)_{ij}| \leq N_{l-1} \|\theta\|_{\infty} \quad (38)$$

Using the previous inequality on $\|W_l - W'_l\|_{\text{op},\infty}$ and replacing it in Inequality equation 35, we deduce that:

$$\|R_{\theta}(\tilde{x}) - R_{\theta'}(\tilde{x})\|_{\infty} \leq \max(D; 1) \sum_{\ell=1}^L N_{\ell-1} \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} \right) \max_{i=1,\dots,\ell-1} \left(\prod_{s=i}^{\ell-1} \|\tilde{W}'_s\|_{\text{op},\infty} \right) \|\theta - \theta'\|_{\infty} \quad (39)$$

Thus, recalling that for all l ,

$$\|\tilde{W}_l\|_{\text{op},\infty} \leq r_l \quad \text{and} \quad \|\tilde{W}'_l\|_{\text{op},\infty} \leq r_l \quad (40)$$

Equation equation 39 becomes:

$$\begin{aligned}
\|R_\theta(\tilde{x}) - R_{\theta'}(\tilde{x})\|_\infty &\leq \max(D; 1) \sum_{\ell=1}^L N_{l-1} \left(\prod_{k=\ell+1}^L r_k \right) \max_{i=1, \dots, l-1} \left(\prod_{s=i}^{l-1} r_s \right) \|\theta - \theta'\|_\infty \\
&\leq \max(D; 1) \sum_{\ell=1}^L \left(N_{l-1} \max_{i=1, \dots, l-1} \prod_{\substack{j=i \\ j \neq l}}^L r_j \right) \|\theta - \theta'\|_\infty
\end{aligned} \tag{41}$$

Then taking the maximum over all layers, we finally have:

$$\|R_\theta(\tilde{x}) - R_{\theta'}(\tilde{x})\|_\infty \leq \max(D; 1) \left(\max_{l=1, \dots, L} \max_{i=1, \dots, l-1} \prod_{\substack{j=i \\ j \neq l}}^L r_j \right) \sum_{\ell=1}^L N_{l-1} \|\theta - \theta'\|_\infty \tag{42}$$

Then, we can rewrite this to show the geometric mean for partial products:

$$\|R_\theta(\tilde{x}) - R_{\theta'}(\tilde{x})\|_\infty \leq \max(D; 1) \left(\sqrt[L-1]{\max_{l=1, \dots, L} \left(\max_{i=1, \dots, l-1} \prod_{\substack{j=i \\ j \neq l}}^L r_j \right)} \right)^{L-1} \sum_{\ell=1}^L N_{l-1} \|\theta - \theta'\|_\infty \tag{43}$$

Finally, taking the supremum of both sides we obtain:

$$\sup_{x \in \Omega} \|R_\theta(\tilde{x}) - R_{\theta'}(\tilde{x})\|_\infty \leq \max(D; 1) \left(\sqrt[L-1]{\max_{l=1, \dots, L} \left(\max_{i=1, \dots, l-1} \prod_{\substack{j=i \\ j \neq l}}^L r_j \right)} \right)^{L-1} \sum_{\ell=1}^L N_{l-1} \|\theta - \theta'\|_\infty \tag{44}$$

□

We now provide an intermediate Theorem (dealing with MLP without biases) to complete Theorem 4.1 (MLP with biases not quantified) and Theorem 4.4 (CNN with no biases).

Theorem A.4 (Bound for neural networks without bias). *With the same settings as in Theorem 4.1, with $(b_1, \dots, b_L) = (b'_1, \dots, b'_L) = (0, \dots, 0)$ (i.e. $\forall l$, we can take $\tilde{W}_l = W_l$ and $\tilde{W}'_l = W'_l$, the standard weight matrices without included bias). For all $x \in \Omega$ the following bound holds.*

$$\sup_{x \in \Omega} \|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq D \left(\sqrt[L-1]{\max_{l=1, \dots, L} \prod_{\substack{k=1 \\ k \neq l}}^L r_k} \right)^{L-1} \sum_{\ell=1}^L N_{l-1} \|\theta - \theta'\|_\infty. \tag{45}$$

Proof. By analogy of the previous proof, we use Equation equation 23 (which corresponds to Lemma A.2 but without bias), and it comes:

$$\|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq \sum_{\ell=1}^L \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op}, \infty} \right) \|W_\ell - W'_\ell\|_{\text{op}, \infty} \|R_{\theta'_{\ell-1}}(x)\|_\infty \tag{46}$$

Now we want to bound the term $\|R_{\theta'_{\ell-1}}(x)\|_\infty$ by using equation 23. Thus, for any θ and with $\theta' = (0, \dots, 0)$, noticing that:

$$\begin{aligned} \forall l \geq 2, \|R_{\theta'_{\ell-1}}(x)\|_\infty &= 0, \\ \text{if } l = 1, \|R_{\theta'_{\ell-1}}(x)\|_\infty &= \|x\|_\infty, \text{ by convention} \end{aligned} \quad (47)$$

We have:

$$\|R_{\theta_{l-1}}(x)\|_\infty \leq \prod_{k=2}^{l-1} \|W_k\|_{\text{op},\infty} \|W_1 - 0\|_{\text{op},\infty} \|x\|_\infty = \prod_{k=1}^{l-1} \|W_k\|_{\text{op},\infty} \|x\|_\infty \quad (48)$$

Now for any θ, θ' without bias, we can bound $\|R_{\theta'_{\ell-1}}(x)\|_\infty$ in equation 23, and it comes:

$$\|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq \|x\|_\infty \sum_{\ell=1}^L \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} \right) \prod_{k=1}^{\ell-1} \|W'_k\|_{\text{op},\infty} \|W_\ell - W'_\ell\|_{\text{op},\infty} \quad (49)$$

Then, we use Lemma A.1 to get:

$$\|W_l\|_{\text{op},\infty} \leq N_{l-1} \max_{i,j} |(W_l)_{ij}| \leq N_{l-1} \|\theta\|_\infty \quad (50)$$

Hence recalling that for all k we have $\|W_k\|_{\text{op},\infty} \leq r_k$ and $\|W'_k\|_{\text{op},\infty} \leq r_k$, it comes:

$$\begin{aligned} \|R_\theta(x) - R_{\theta'}(x)\|_\infty &\leq \|x\|_\infty \sum_{\ell=1}^L N_{l-1} \left(\prod_{k=\ell+1}^L \|W_k\|_{\text{op},\infty} \right) \prod_{k=1}^{\ell-1} \|W'_k\|_{\text{op},\infty} \|\theta - \theta'\|_\infty \\ &\leq D \sum_{\ell=1}^L N_{l-1} \prod_{\substack{k=1 \\ k \neq \ell}}^L r_k \|\theta - \theta'\|_\infty \end{aligned} \quad (51)$$

Thus taking the maximum over all layers, we can rewrite the bound in terms of the geometric mean:

$$\|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq D \left(\sqrt[L-1]{\max_{l=1,\dots,L} \prod_{\substack{k=1 \\ k \neq l}}^L r_k} \right)^{L-1} \sum_{\ell=1}^L N_{l-1} \|\theta - \theta'\|_\infty \quad (52)$$

Finally, taking the supremum gives the desired result:

$$\sup_{x \in \Omega} \|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq D \left(\sqrt[L-1]{\max_{l=1,\dots,L} \prod_{\substack{k=1 \\ k \neq l}}^L r_k} \right)^{L-1} \sum_{\ell=1}^L N_{l-1} \|\theta - \theta'\|_\infty. \quad (53)$$

□

We now give the proof of our last theorem.

Proof of Theorem 4.4. Let $\theta = (\mathcal{H}_1, \dots, \mathcal{H}_L)$ the vector of parameters where each \mathcal{H}_ℓ represent the convolution matrix of layer ℓ .

With the convention that

$$\begin{cases} R_{\theta'_l}(x) = x & \text{if } l = 1, \\ \prod_{k=\ell+1}^L \|\mathcal{H}_k\|_{\text{op},\infty} = 1 & \text{if } l = L \end{cases} \quad (54)$$

We can start the proof using the same line of reasoning. Thus we can use directly Equation equation 49 that becomes for the convolutional case:

$$\|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq \|x\|_\infty \sum_{l=1}^L \prod_{k=\ell+1}^L \|\mathcal{H}_k\|_{\text{op},\infty} \prod_{k=1}^{l-1} \|\mathcal{H}'_k\|_{\text{op},\infty} \times \|\mathcal{H}_l - \mathcal{H}'_l\|_{\text{op},\infty} \quad (55)$$

Then by analogy we can write:

$$\begin{aligned} \|R_\theta(x) - R_{\theta'}(x)\|_\infty &\leq \|x\|_\infty \sum_{l=1}^L \prod_{k=\ell+1}^L \|\mathcal{H}_k\|_{\text{op},\infty} \prod_{k=1}^{l-1} \|\mathcal{H}'_k\|_{\text{op},\infty} \times \|\mathcal{H}_l - \mathcal{H}'_l\|_{\text{op},\infty} \\ &\leq D \sum_{l=1}^L \prod_{k=\ell+1}^L r_k \prod_{k=1}^{l-1} r_k \times \|\mathcal{H}_l - \mathcal{H}'_l\|_{\text{op},\infty} \\ &= D \sum_{l=1}^L \prod_{\substack{k=1 \\ k \neq l}}^L r_k \times \|\mathcal{H}_l - \mathcal{H}'_l\|_{\text{op},\infty} \end{aligned} \quad (56)$$

Then, we know by lemma A.1 that for every matrix in $\mathbb{R}^{m \times n}$ it holds :

$$\|W\|_{\text{op},\infty} = \max_{1 \leq i \leq m} \sum_{j=1}^n |w_{ij}| \quad (57)$$

Thanks to the convolutional structure, we improve the bound on $\|\mathcal{H}_l\|_{\text{op},\infty}$ given by Lemma A.1. Let us note the output of the previous layer as y_{l-1} . This output is a set of feature maps with dimensions $(n_{l-1} \times m_{l-1}) \times c_{l-1}$, where c_{l-1} is the number of feature maps (i.e., the number of filters) in the previous layer.

Next, recalling that, we want to express the convolution at layer l as a matrix multiplication: $\mathcal{H}_l \text{vec}(y_{l-1})$.

Then we write \mathcal{H}_l as a block-Toeplitz matrix:

$$\mathcal{H}_l = \begin{pmatrix} H_{l,1} \\ \vdots \\ H_{l,c_l} \end{pmatrix} \quad (58)$$

where each block $H_{l,i}$ is a Toeplitz matrix of size $(n_l m_l) \times (n_{l-1} m_{l-1} c_{l-1})$. These matrices are highly sparse, with each row composed of coefficients of the filters arranged in a specific pattern, while the remaining entries are zeros.

Thus, each block $H_{l,i}$ in \mathcal{H}_l represents the convolution operation between the i -th filter and the feature maps of the previous layer.

Hence, the overall dimensions of \mathcal{H}_l are:

$$\dim(\mathcal{H}_l) = (n_l m_l c_l) \times (n_{l-1} m_{l-1} c_{l-1}). \quad (59)$$

Then to bound the norm of \mathcal{H}_l , we consider only the non-zero coefficients in its rows, which are $p_l^2 \times c_{l-1}$, where p_l^2 denotes the size of the filters at layer l .

Thus, recalling that $\theta = (\mathcal{H}_1, \dots, \mathcal{H}_L)$ we have:

$$\|\mathcal{H}_l\|_{\text{op},\infty} \leq c_{l-1} \times p_l^2 \max_{i,j} |(\mathcal{H}_l)_{ij}| \leq c_{l-1} \times p_l^2 \|\theta\|_\infty \quad (60)$$

Using the previous inequality on $\|\mathcal{H}_l - \mathcal{H}'_l\|_{\text{op},\infty}$ we can bound the quantity $\|R_\theta(x) - R_{\theta'}(x)\|_\infty$, by:

$$\begin{aligned} \|R_\theta(x) - R_{\theta'}(x)\|_\infty &\leq D \sum_{l=1}^L \prod_{\substack{k=1 \\ k \neq l}}^L r_k \times \|\mathcal{H}_l - \mathcal{H}'_l\|_{\text{op},\infty} \\ &\leq D \sum_{l=1}^L c_{l-1} \times p_l^2 \prod_{\substack{k=1 \\ k \neq l}}^L r_k \|\theta - \theta'\|_\infty \end{aligned} \quad (61)$$

Thus taking the maximum over all layers, we can rewrite the bound in terms of the geometric mean:

$$\|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq D \left(\sqrt[L-1]{\max_{l=1,\dots,L} \prod_{\substack{k=1 \\ k \neq l}}^L r_k} \right)^{L-1} \sum_{\ell=1}^L c_{\ell-1} \times p_\ell^2 \|\theta - \theta'\|_\infty \quad (62)$$

Hence, we can conclude that:

$$\sup_{x \in \Omega} \|R_\theta(x) - R_{\theta'}(x)\|_\infty \leq D \left(\sqrt[L-1]{\max_{l=1,\dots,L} \prod_{\substack{k=1 \\ k \neq l}}^L r_k} \right)^{L-1} \sum_{\ell=1}^L c_{\ell-1} \times p_\ell^2 \|\theta - \theta'\|_\infty. \quad (63)$$

□

Then, we present the 3 specific architectures of residual and bottleneck block, in Resnet18, Resnet50 He et al. (2016) and MobilnetV2 Sandler et al. (2018).

B Calculations for MobileNetV2 and Resnets

B.1 Specific structure of Resnet18

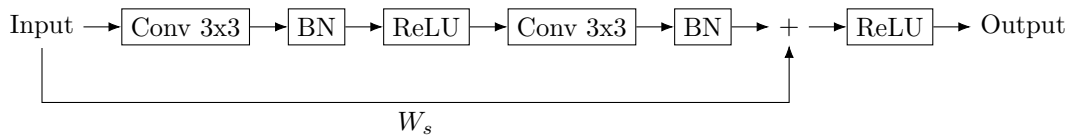


Figure 6: Structure of a Residual Block of ResNet18

Lemma B.1 (Matrix Representation of a Residual Block in ResNet-18). *The output $y \in \mathbb{R}^n$ of a residual block in ResNet-18, as illustrated in Figure 6, can be expressed as:*

$$y = \sigma(V_2 \cdot \tilde{\sigma}_1(V_1 \cdot f)), \quad (64)$$

where $f \in \mathbb{R}^n$ is the input, and V_1 and V_2 are defined as:

$$V_1 = \begin{pmatrix} W_1 \\ I \end{pmatrix} \in \mathbb{R}^{(d+n) \times n}, \quad V_2 = \begin{pmatrix} W_2 & W_s \end{pmatrix} \in \mathbb{R}^{m \times (d+n)}, \quad (65)$$

with $W_1 \in \mathbb{R}^{d \times n}$ and $W_2 \in \mathbb{R}^{m \times d}$ as the convolutional weight matrices, $I \in \mathbb{R}^{n \times n}$ as the identity matrix and $W_s \in \mathbb{R}^{m \times n}$ represents the shortcut weight matrix. Note that, if the input and output of the block have the same dimension $W_s = I$. The function $\tilde{\sigma}_1$ applies the non-linearity σ only to the term $W_1 \cdot f$, leaving the shortcut component $I \cdot f$ unchanged.

Proof. The residual block computes the output y as:

$$y = \sigma(\text{BN}_2(\text{Conv}_2(\text{ReLU}(\text{BN}_1(\text{Conv}_1(f)))) + W_s f), \quad (66)$$

where: Conv_1 and Conv_2 represent the convolutional layers with weight matrices W_1 and W_2 , respectively, BN_1 and BN_2 are batch normalization layers (omitted in the matrix formulation because we removed them in our experiments).

The first convolutional layer computes:

$$x_1 = \text{Conv}_1(f) = W_1 \cdot f, \quad (67)$$

where $W_1 \in \mathbb{R}^{d \times n}$. This result is passed through BN_1 and σ , yielding:

$$\tilde{x}_1 = \sigma(\text{BN}_1(x_1)). \quad (68)$$

In our matrix representation, we express this as:

$$\tilde{x}_1 = \tilde{\sigma}_1(V_1 \cdot f), \quad (69)$$

where $V_1 = \begin{pmatrix} W_1 \\ I \end{pmatrix} \in \mathbb{R}^{(d+n) \times n}$. The expanded computation is:

$$V_1 \cdot f = \begin{pmatrix} W_1 \cdot f \\ I \cdot f \end{pmatrix} = \begin{pmatrix} W_1 \cdot f \\ f_1 \\ \vdots \\ f_n \end{pmatrix}. \quad (70)$$

Thus:

$$\tilde{\sigma}_1(V_1 \cdot f) = \begin{pmatrix} \sigma(W_1 \cdot f) \\ f_1 \\ \vdots \\ f_n \end{pmatrix}. \quad (71)$$

Then the second convolutional layer computes:

$$x_2 = \text{Conv}_2(\sigma(\text{BN}_1(x_1))) = W_2 \cdot \tilde{x}_1, \quad (72)$$

where $W_2 \in \mathbb{R}^{m \times d}$. Combining the result with the shortcut connection $W_s f$, we obtain:

$$y = \sigma(x_2 + W_s f). \quad (73)$$

Using our matrix representation for V_2 , where $V_2 = \begin{pmatrix} W_2 & W_s \end{pmatrix} \in \mathbb{R}^{m \times (d+n)}$, the computation expands as:

$$V_2 \cdot \begin{pmatrix} \sigma(W_1 \cdot f) \\ f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} W_2 & W_s \end{pmatrix} \cdot \begin{pmatrix} \sigma(W_1 \cdot f) \\ f_1 \\ \vdots \\ f_n \end{pmatrix} = W_2 \cdot \sigma(W_1 \cdot f) + W_s \cdot f = x_2 + W_s f. \quad (74)$$

So the final output of the block is:

$$y = \sigma(V_2 \cdot \tilde{\sigma}_1(V_1 \cdot f)), \quad (75)$$

Thus, the residual block's output is equivalent to our matrix representation. \square

An immediate consequence of this Lemma is:

$$\begin{cases} \|V_1\|_{\text{op},\infty} = \max(1, \|W_1\|_{\text{op},\infty}), \\ \|V_2\|_{\text{op},\infty} = \|W_2\|_{\text{op},\infty} + 1, & \text{if } W_s = I, \\ \|V_2\|_{\text{op},\infty} \leq \|W_2\|_{\text{op},\infty} + \|W_s\|_{\text{op},\infty}, & \text{otherwise.} \end{cases} \quad (76)$$

B.2 Specific structure of Resnet50

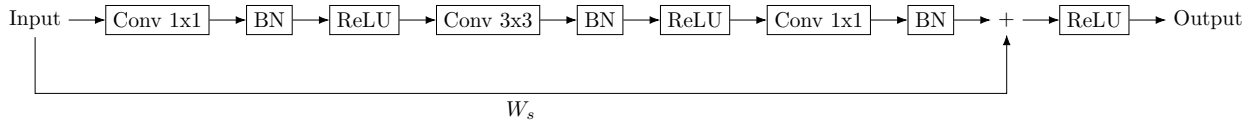


Figure 7: Structure of a Residual Block of ResNet50

Lemma B.2 (Matrix Representation of a Bottleneck Block in ResNet-50). *The output $y \in \mathbb{R}^n$ of a bottleneck block in ResNet-50, as illustrated in Figure 7, can be expressed as:*

$$y = \sigma(V_3 \cdot \tilde{\sigma}_2(V_2 \cdot \tilde{\sigma}_1(V_1 \cdot f))), \quad (77)$$

where $f \in \mathbb{R}^n$ is the input, and the matrices V_1 , V_2 , and V_3 are defined as follows:

$$V_1 = \begin{pmatrix} W_1 \\ I \end{pmatrix} \in \mathbb{R}^{(d_1+n) \times n}, \quad V_2 = \begin{pmatrix} W_2 & 0 \\ 0 & I \end{pmatrix} \in \mathbb{R}^{(d_2+n) \times (d_1+n)}, \quad V_3 = \begin{pmatrix} W_3 & W_s \end{pmatrix} \in \mathbb{R}^{m \times (d_2+n)}. \quad (78)$$

Here: $W_1 \in \mathbb{R}^{d_1 \times n}$, $W_2 \in \mathbb{R}^{d_2 \times d_1}$, and $W_3 \in \mathbb{R}^{m \times d_2}$ are the weight matrices of the three convolutional layers in the bottleneck block, W_s is the weight matrix associated with the shortcut, $I \in \mathbb{R}^{n \times n}$ is the identity, and 0 denotes zero matrices of appropriate dimensions.

The functions $\tilde{\sigma}_1$ and $\tilde{\sigma}_2$ apply the non-linearity σ only to specific components, leaving the shortcut components unchanged.

Proof. The bottleneck block computes the output y as:

$$y = \sigma(\text{BN}_3(\text{Conv}_3(\text{ReLU}(\text{BN}_2(\text{Conv}_2(\text{ReLU}(\text{BN}_1(\text{Conv}_1(f)))))))) + W_s f), \quad (79)$$

where Conv_1 , Conv_2 , and Conv_3 represent the three convolutional layers with weight matrices W_1 , W_2 , and W_3 , respectively, BN_1 , BN_2 , and BN_3 are batch normalization layers (omitted in the matrix formulation because we removed them in our experiments).

The first convolutional layer computes:

$$x_1 = \text{Conv}_1(f) = W_1 \cdot f, \quad (80)$$

where $W_1 \in \mathbb{R}^{d_1 \times n}$. This result is passed through BN_1 and σ , yielding:

$$\tilde{x}_1 = \sigma(\text{BN}_1(x_1)). \quad (81)$$

In our matrix representation, we express this as:

$$\tilde{x}_1 = \tilde{\sigma}_1(V_1 \cdot f), \quad (82)$$

where $V_1 = \begin{pmatrix} W_1 \\ I \end{pmatrix} \in \mathbb{R}^{(d_1+n) \times n}$. The expanded computation is:

$$V_1 \cdot f = \begin{pmatrix} W_1 \cdot f \\ I \cdot f \end{pmatrix} = \begin{pmatrix} W_1 \cdot f \\ f_1 \\ \vdots \\ f_n \end{pmatrix}. \quad (83)$$

Thus:

$$\tilde{\sigma}_1(V_1 \cdot f) = \begin{pmatrix} \sigma(W_1 \cdot f) \\ f_1 \\ \vdots \\ f_n \end{pmatrix}. \quad (84)$$

The second convolutional layer computes:

$$\tilde{x}_2 = \sigma(\text{Conv}_2(\tilde{x}_1)) = \sigma(W_2 \cdot \tilde{x}_1), \quad (85)$$

where $W_2 \in \mathbb{R}^{d_2 \times d_1}$. Using V_2 , the expanded computation is:

$$V_2 \cdot \begin{pmatrix} \sigma(W_1 \cdot f) \\ f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} W_2 & 0 \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} \sigma(W_1 \cdot f) \\ f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} W_2 \cdot \sigma(W_1 \cdot f) \\ f_1 \\ \vdots \\ f_n \end{pmatrix}. \quad (86)$$

So the output of this layer with our matrix representation is: $\begin{pmatrix} \sigma(W_2 \cdot \sigma(W_1 \cdot f)) \\ f_1 \\ \vdots \\ f_n \end{pmatrix}$.

The third convolutional layer combines the outputs of the second layer and the shortcut connection:

$$x_3 = \text{Conv}_3(\tilde{x}_2) = W_3 \cdot \tilde{x}_2 + W_s f, \quad (87)$$

and the final original output of the block is $y = \sigma(x_3)$

Using V_3 , this becomes:

$$V_3 \cdot \begin{pmatrix} \sigma(W_2 \cdot \sigma(W_1 \cdot f)) \\ f_1 \\ \vdots \\ f_n \end{pmatrix} = (W_3 \quad W_s) \cdot \begin{pmatrix} \sigma(W_2 \cdot \sigma(W_1 \cdot f)) \\ f_1 \\ \vdots \\ f_n \end{pmatrix} = W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot f)) + W_s f. \quad (88)$$

Then the final output is:

$$y = \sigma(V_3 \cdot \tilde{\sigma}_2(V_2 \cdot \tilde{\sigma}_1(V_1 \cdot f))), \quad (89)$$

ending the proof of the matrix representation. \square

An immediate consequence of this Lemma is that:

$$\begin{cases} \|V_1\|_{\text{op},\infty} = \max(1, \|W_1\|_{\text{op},\infty}), \\ \|V_2\|_{\text{op},\infty} = \max(1, \|W_2\|_{\text{op},\infty}), \\ \|V_3\|_{\text{op},\infty} = \|W_3\|_{\text{op},\infty} + 1, & \text{if } W_s = I, \\ \|V_3\|_{\text{op},\infty} \leq \|W_3\|_{\text{op},\infty} + \|W_s\|_{\text{op},\infty}, & \text{otherwise.} \end{cases} \quad (90)$$

B.3 Specific structure of MobileNetV2

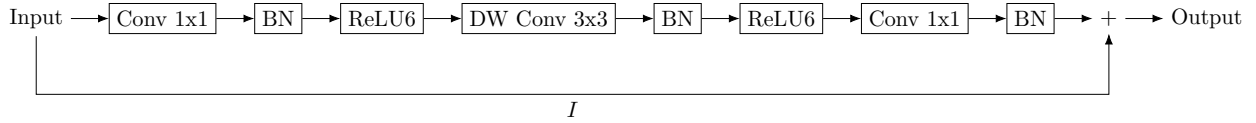


Figure 8: Structure of a Residual Block of MobileNetV2

Remark B.3. The only difference between Resnet-50 bottleneck block and MobileNetV2 bottleneck block is that MobileNetV2 uses inverted residuals bottleneck block (expansion before convolution) and Depthwise convolution (only one filter is applied for each input channel), without activation function at the end of the block. Thus the matrix representation can also be expressed as:

$$y = V_3 \cdot \tilde{\sigma}_2(V_2 \cdot \tilde{\sigma}_1(V_1 \cdot f)), \quad (91)$$

Where all matrices are defined in Lemma B.2.

C Experimental setup for the MLP case

For the Figure 5, we developed four MLPs with different depths. All of them were trained on MNIST dataset during 2 epochs, using the Adam optimizer with a learning rate of 0.001 and a batch size of 64. The MLP with depth 5 has four hidden layers with sizes [1024, 512, 256, 128]. The MLP with depth 7 has six hidden layers with sizes [1024, 512, 256, 128, 64, 32]. The MLP with depth 9 has eight hidden layers with sizes [1024, 512, 256, 128, 128, 64, 64, 32]. The MLP with depth 11 has ten hidden layers with sizes [1024, 512, 512, 256, 256, 128, 128, 64, 64, 32]. The models were quantized using uniform quantization with different bit widths: 4, 8, 16, and 24 bits only on the weights.

D Experimental Setup of Figure 4

Dataset: We used the Tiny ImageNet dataset as described in Section 5. Images were resized to 224×224 resolution. For training, standard normalization and data augmentation was applied.

Model and training: We replaced ResNet18, ResNet50 and MobileNetV2, final classification layer by a linear layer with 200 outputs and Batch normalization layers were removed. Models were initialized from pretrained ImageNet weights. Training was done for 80 epochs using SGD with learning rate 0.01, momentum 0.9, and weight decay $5 \cdot 10^{-4}$. Batch size was 128 for both training and validation.

AdaRound: applied layerwise using a calibration set of 256 samples. Each layer’s rounding parameters were optimized for 15 steps using Adam with learning rate 10^{-3} and regularization coefficient $\lambda = 0.01$.

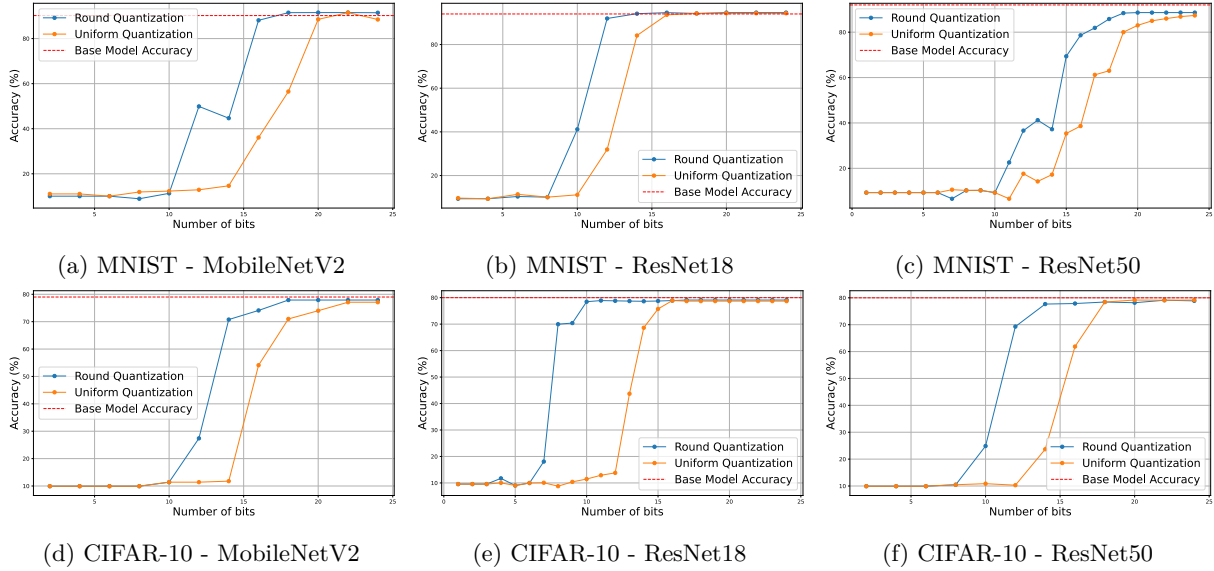


Figure 9: Graphs illustrating the effect of quantization on performance for two quantization functions (round and uniform). The results highlight how quantization reduces memory requirements while maintaining or approaching the base model’s accuracy. The amount of quantization needed to reach the base precision depends on the quantization function used.

E Experiences on MNIST and CIFAR-10 datasets

We use two widely-used image classification datasets. The MNIST dataset: 28x28 grayscale images of handwritten digits, with 60,000 training samples and 10,000 test samples across 10 classes. The CIFAR-10 dataset: 32x32 color images in 10 different classes, with 50,000 images for training and 10,000 for testing. In Figure 9, we analyze the impact of post-training quantization on the precision of various pretrained models, across two datasets: MNIST and CIFAR-10. We clearly observe the behavior reflected by the form of the bound: depth significantly influences quantization error. For instance, in the case of ResNet50 on MNIST, uniform quantization impacts precision as the model fails to reach the baseline precision even at 24 bits. Conversely, ResNet18 achieves the baseline precision with just 12 bits on MNIST (using uniform quantization). To a lesser extent, we can also suppose that other parameters play a role. For example, despite having very similar depths, MobileNetV2 and ResNet50 exhibit noticeably different quantized performances. This is probably due to the specific architecture of MobileNetV2 that uses residual bottlenecks and Relu6 activation function ($Relu6(x) := \min(\max(0, x), 6)$) which is known to better support quantization Sandler et al. (2018). Specifically, MobileNetV2 reaches baseline precision with 20-bit quantization on MNIST.

Furthermore, the dataset influences precision as well. On CIFAR-10, ResNet50 handles quantization much better, achieving the desired precision with only 18 bits of quantization (14 bits with rounding quantization). Finally, the quantization method itself significantly affects the results: rounding quantization offers an average gain of approximately 4 bits, regardless of the dataset or model, to achieve the desired precision. This behavior is accounted for in the bound through the factor involving $\|\theta - \theta'\|_\infty$.

F Experimental Setup of Figure 9

Dataset: We used MNIST and CIFAR-10 datasets as described in Appendix E. All images were resized to 224×224 to be compatible with our three pretrained architectures. Input normalization was applied for each channel. For MNIST, which contains grayscale images, the single channel was duplicated to produce 3-channel inputs. On MNIST, only 30% of the training and test sets were used, with a batch size of 32. On CIFAR-10, the full dataset was used with a batch size of 64.

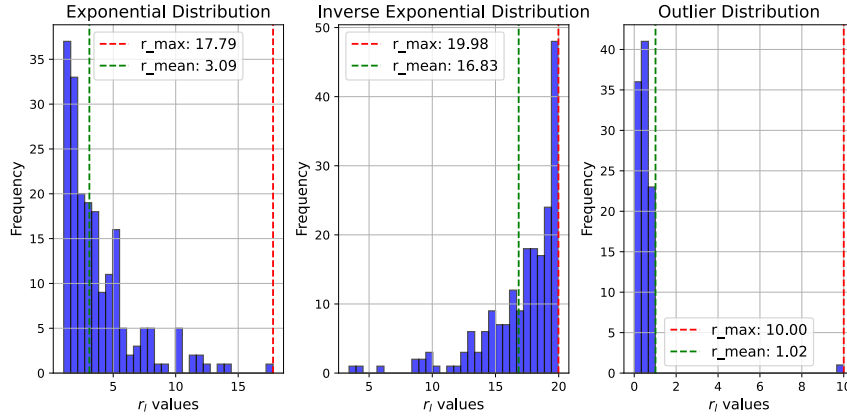


Figure 10: Comparison between the r_{mean} (green) used in Theorem 4.1 and r (red) used in Theorem 3.5, for three different simulated distributions, showing a smaller value compared to r for each distribution.

Models and training on MNIST: All models were initialized with ImageNet-pretrained weights. Batch normalization layers were removed from each architecture and replaced with identity layers. For all models, the final classification layer was replaced with a new linear layer with 10 outputs. ResNet18 and ResNet50 were fine-tuned by unfreezing only the last residual block (`layer4`) along with the final linear layer (`fc`), and trained for 1 epoch. In the case of MobileNetV2, only the last convolutional block (`features[-1]`) and the classifier were unfrozen. The rest of the network remained frozen. Training was performed for 15 epochs using the Adam optimizer with a learning rate of 0.001.

Models and training on CIFAR-10: For CIFAR-10, the full dataset was used. Batch normalization layers were also removed before training. The ResNet18 model was trained for 5 epochs, with `layer1`, `layer4`, and `fc` layers unfrozen. The ResNet50 model was trained for 20 epochs with all layers unfrozen. MobileNetV2 was trained for 35 epochs. In this case, the convolutional blocks `features[0]`, `features[1]`, `features[17]`, `features[18]` and the classifier were the only unfrozen layers. All training process used the Adam optimizer with a learning rate of 0.001.

G Comparison between r_{mean} and r_{max}

In Figure 10 we simulate several values of r_ℓ with different distributions. For the exponential distribution, there is variability across layers, most of r_ℓ values are small (less than 5) but the max is around 18. For this case we have the value of r_{mean} around 3. The opposite case is the second histogram where most of r_j values are large (more than 15), but even in this case, r_{mean} allows to better fit to the distribution, taking account of the few small values of r_j . Finally, the last scenario, is the best for our bound, because for all layers except the last one, $r_j \in [0, 1]$ and the last one is 10. With this distribution we have that, r_{mean} is equal to 1.02. We will see in Section 5 practical examples, and especially MobileNetV2, that is close to the exponential distribution, making us gain orders of magnitude as the approximation constant grows exponentially with respect to the depth (with parameter r_{mean}).

H Cross layer equalization as a preprocessing

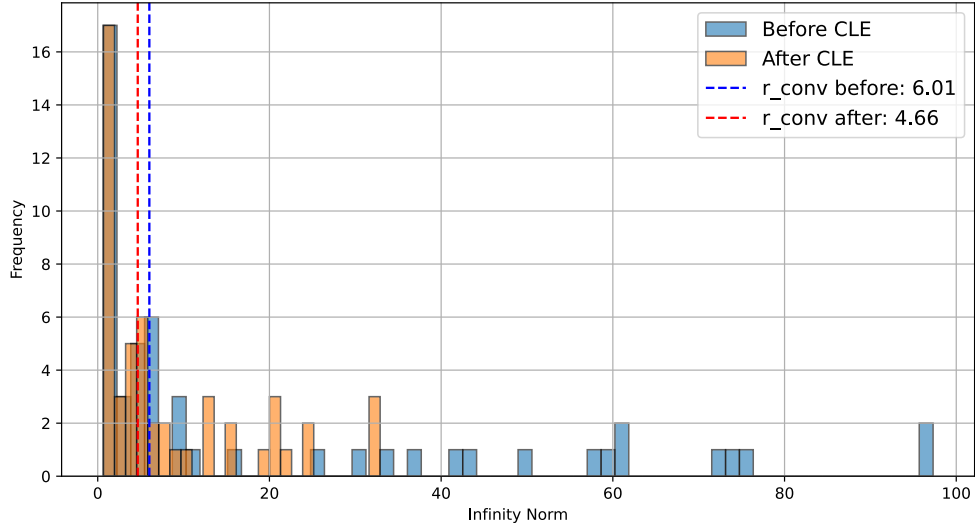
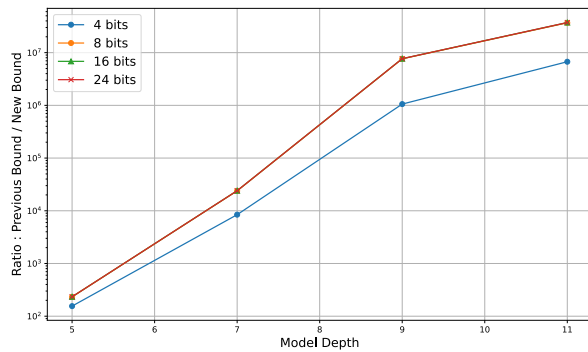


Figure 11: Comparison between weight norm distribution before and after cross-layer equalization on a 4-bits ResNet50, showing a smaller value of r_{mean} after CLE.

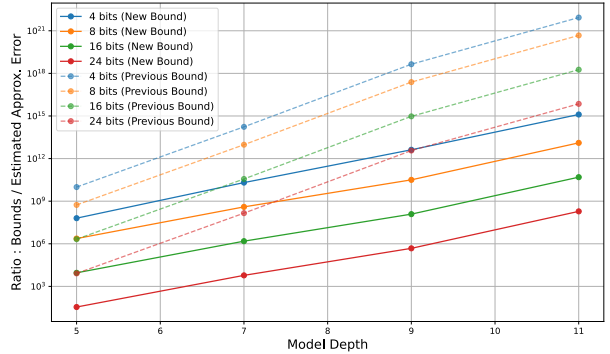
In Figure 11 we notice that the cross layer equalization have uniformized the distribution of weight norms and therefore the value of r_{conv} decreased as well. Thus our bound can be combined with preprocessing tends to reduce its pessimism, even if it is not sufficient to have a tight bound.

I MLPs with non-ReLu activation function

Settings of Figure 12 is exactly the same as Appendix C but we replace de ReLu activation functions, by sigmoid, implemented by Tanh, to show that our approach is robust to other activation functions.



(a) Ratio in log scale between the previous bound equation 4 and our bound equation 7 as a function of model depth for different quantization bit-widths.



(b) Ratio in log scale between bounds equation 4, equation 7 and the estimated error approximation, as a function of model depth for different quantization bit-widths.

Figure 12: Comparison for MLPs of depths 5, 7, 9 and 11 on MNIST. (a) shows how the ratio of our bound over the previous bound grows exponentially with depth, and (b) demonstrates that our bound reduces that exponential dependence across bit-widths.

J Hardware

Most experiments in this paper, including all MLP evaluations and quantization analyses on MNIST and CIFAR-10, were conducted on a MacBook Air with an Apple M2 chip (8-core CPU, 8-core GPU) and 16 GB of unified memory. The experiments shown in Figure 4 were performed on an NVIDIA DGX A100 system. The DGX hardware allows us to retrain full CNN architectures on the Tiny ImageNet dataset. We emphasize that execution time and efficiency metrics are not the focus of this work. As our contribution is primarily theoretical, experiments are here only to validate and illustrate the behavior of our approximation bounds in practice.