# Understanding How Orthogonality of Parameters Improves Quantization of Neural Networks

Sukru Burc Eryilmaz and Aysegul Dundar

***Abstract*— We analyze why the orthogonality penalty improves quantization in deep neural networks. Using results from perturbation theory as well as through extensive experiments with Resnet50, Resnet101, and VGG19 models, we mathematically and experimentally show that improved quantization accuracy resulting from orthogonality constraint stems primarily from reduced condition numbers, which is the ratio of largest to smallest singular values of weight matrices, more so than reduced spectral norms, in contrast to the explanations in previous literature. We also show that the orthogonality penalty improves quantization even in the presence of a state-of-the-art quantized retraining method. Our results show that, when the orthogonality penalty is used with quantized retraining, ImageNet Top5 accuracy loss from 4- to 8-bit quantization is reduced by up to 7% for Resnet50, and up to 10% for Resnet101, compared to quantized retraining with no orthogonality penalty.**

***Index Terms*— Deep neural networks, orthogonality regularization, perturbation theory, quantization.**

## I. Introduction

DEEP convolutional neural networks (CNNs) have achieved impressive results on a wide range of computer vision tasks from object classification, detection, segmentation, to image and video editing and interpolation. To achieve state-of-the-art results on these tasks, CNNs have become larger and deeper with increased complexity. On the one hand, the excellent accuracies coming from computationally heavy CNNs make them demanding to be used in various applications. On the other hand, it calls for computationally efficient implementations. There are several research directions to make CNNs run in a computationally efficient manner such as efficient implementations of CNNs on different hardware [1]–[6], novel convolutional neural network architectures that exploit memory and computation efficiency [7]–[11], knowledge distillation to decrease the number of parameters [12]–[15], pruning techniques to decrease the network size [16]–[18], and weight or activation quantization of CNNs from 32-bit floating point into lower bit-width representations [19]–[22].

Manuscript received September 7, 2021; revised February 7, 2022; accepted April 27, 2022. *(Corresponding author: Aysegul Dundar.)*

Sukru Burc Eryilmaz is with NVIDIA Corporation, Santa Clara, CA 95051 USA (e-mail: seryilmaz@nvidia.com).

Aysegul Dundar is with the Department of Computer Science, Bilkent University, 06800 Ankara, Turkey (e-mail: adundar@cs.bilkent.edu.tr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3171297.

Digital Object Identifier 10.1109/TNNLS.2022.3171297

In this work, we are interested in network quantization to reduce the model size. Besides reducing the model size, low-precision quantization of weights and activations takes advantage of the low bit-width fixed-point arithmetic units on hardware [23], [24], resulting in power, area, and latency savings. These costs roughly go down linearly for addition operations with the number of bits and quadratically for multiplication operations. Model size, on the other hand, goes down linearly with the bit-width of quantized weights. Consequently, low-precision quantization is a valuable technique for enabling the highly efficient execution of CNNs. However, reduced precision commensurately reduces the number of representable values. This causes perturbations for each value of the input, weight, and output of every quantized layer at inference time, and these perturbations are carried on to subsequent layers, resulting in loss of accuracy, which can be significant for sufficiently low bit-widths (especially for less than 8 bits). Quantizing model weights and activations of a model trained in high precision (32-bit or 16-bit floating point) requires additional processing either during the initial training stage, a retraining stage, or post-training processing stage to mitigate the accuracy loss [24]–[26]. Recently, a retraining method for quantization was introduced [25], where a pretrained model is retrained with quantization thresholds included in the model parameters together with weights, where the quantization thresholds are trained simultaneously with gradient descent for optimizing the range of representable values after quantization. This method achieves state-of-the-art results for several networks and is called trained quantization thresholds (TQTs).

In a different research line, orthogonality in weights has proved to be beneficial for different purposes, such as making optimization more efficient [27], [28] or improving the generalization of the resulting models [28], and improving the quality of images synthesized by generative adversarial models [29]. Orthogonality constraint has also been used for improving the robustness of neural networks to adversarial examples [30], [31], with the latter work also studying robustness in the context of quantization, albeit with small datasets, such as CIFAR-10 or CIFAR-100. Alizadeh *et al.* [32] implements a second-order optimization method for imposing orthogonality constraints on weights, but this method increases training time by an order of magnitude, and it is only studied for small models such as Resnet18. Also, their work does not study whether their method is still beneficial when combined with trained quantization methods, or whether there are cases where trained quantization can

replace the orthogonality regularization. Most importantly, both works [31], [32] attribute improved robustness or quantization from orthogonality to the ability of the orthogonality constraint to reduce the spectral norm (largest singular value) of weight matrices, without doing sufficient analysis. However, orthogonality imposes a stronger restriction, which is minimizing the condition number, or ratio of largest to smallest singular values, which determines the upper bound on the relative sensitivity of matrix multiplication. Previous studies have not studied in depth which of these effects contributes more to improved quantization and mostly cited the former to be the main contributor, in contrast to our observations.

In this article, through extensive analysis of condition numbers and spectral norms of weight matrices, we experimentally show that the ability of orthogonality constraint to minimize the condition number is what improves the accuracy of the quantized model, more so than its ability to lower spectral norm, in contrast to the claims in previous literature. Using existing results from perturbation theory, we support our experimental observations theoretically, namely how orthogonality constraint on weight parameters lowers the sharp upper bound on the relative sensitivity of a layer with respect to perturbations in inputs and weights (in this case, perturbations from quantization noise), by minimizing the condition number of weight parameters. We also perform our experiments on relatively large models such as Resnet50, Resnet101, and VGG19, using the ImageNet dataset. We further combine orthogonality with state-of-the-art TQTs quantized retraining method for quantization to study if the orthogonality is still beneficial when combined with existing quantization methods. Our method significantly improves the accuracy of the quantized model compared to the case where no orthogonality penalty is used in the model training stage. Contributions are summarized as follows.

1) We show through ablation experiments that, in contrast to existing literature, improved quantization due to orthogonality in neural networks is due to the reduction in condition number of weight parameters, and not necessarily due to the reduction in the spectral norm.
2) We use the existing literature in perturbation theory on the error bounds of perturbed matrices to theoretically explain the experimental results, that is, the observation that the reduced condition number leads to improved quantization.
3) By experimenting with a state-of-the-art quantized training method, we show that quantized training methods do not necessarily replace the orthogonality penalty in terms of improved quantization and that the orthogonality penalty still benefits quantization even when a quantized training method is employed.
4) While our aim is not to achieve the state-of-the-art quantization results, but rather to explain how orthogonality of parameters improves quantization of neural networks, we provide consistent improvements on three popular network architectures namely, ResNet50, ResNet101, and VGG19.

## II. Related Work

### A. Integer Quantization of Deep Neural Networks

Prior work on quantization includes employing ternary quantization [20], [33] or even as low as 1-bit quantization [21], [34]. However, these methods introduce higher accuracy loss than often desired, and more focus has been on intermediate bit-width quantization, with a bit-width of two or more [35]–[41]. For more practical deployment, recent work has emphasized uniform quantization [26], per-tensor quantization [25], and symmetric quantization [24]. Calibration of quantization thresholds using KL divergence or max-values of pretrained weights achieves good accuracy for CNNs for 8-bit quantization without a change in the training process [42]. Other works have explored quantization-aware training with different approaches [24], [43]–[45], where a quantization node is included in the training graph and it is made differentiable by using a straight-through estimator (STE) [46]. TQTs [25] report that a common problem with these approaches is that they do not balance well range and precision when training for quantization and implement a method where a pretrained model is retrained with quantization nodes that still use STE, but on top of that trains activation and weight thresholds in the log-domain. TQT achieves state-of-the-art results for some networks, such as MobileNetv2. As we show later, these quantized training methods do not fully replace the method of imposing the orthogonality penalty on parameters. Orthogonality penalty can be combined with quantized training methods to further improve the accuracy of the quantized model. In our experiments, we combine our method with TQT, and on top of that, we apply the orthogonality penalty during the initial training phase, as will be explained in Section V-A.

### B. Orthogonality Regularization of Parameters in Deep Neural Networks

As will be explained in more depth in Section III-A, orthogonal matrices, where the columns are orthogonal to each other, minimize a sharp upper bound on the relative sensitivity of matrix multiplication operation with respect to perturbations on the matrix or the input, according to perturbation theory literature. Quantization is a form of perturbation on the inputs or weights to a convolution or fully-connected layer, hence it is expected that orthogonality of weight matrices reduces the relative sensitivity of convolutional or fully connected layers to perturbations in the inputs or weights, hence improving the accuracy of the quantized model, and we theoretically explain and experimentally show this phenomenon. Prior works have used orthogonality of parameters in different contexts. Various works [47]–[52] have studied orthogonality of parameters in the context of recurrent neural networks for improving training process and mitigating vanishing gradients problem. Others [53], [54] studied the benefits of orthogonal initialization for CNNs for more efficient optimization.

These methods, however, do not have a mechanism to maintain orthogonality throughout the training, and orthogonality breaks down without a proper regularizer during training [28], [53]. To achieve orthogonality of weights during training,

several works [55]–[57] use hard orthogonality constraints based on the Stiefel manifold. These works using hard constraints require repeatedly computing singular value decomposition during training, which could be expensive to compute. Furthermore, when the weight matrix is rank-deficient (for instance, when the number of rows is lower than the number of columns), hard orthogonality constraints require extra precautions [28]. Xie *et al.* [27] use a soft orthogonality constraint, where a penalty term proportional to the Frobenius norm of $W^TW - I$, given by $\lambda\|W^TW - I\|_F$, encouraging weights to be orthonormal during training. This study reports significant accuracy gains in certain CNN architectures, although most gains are reported by custom architectures and not necessarily for state-of-the-art convolutional residual models. It replaces a weight decay regularizer with an orthonormality penalty, but we found in our experiments that using a combination of two (albeit with a reduced factor of weight decay when combined with an orthonormal regularizer) results in better accuracy for larger models. Bansal *et al.* [28], in addition to soft orthogonality regularizer in [27], also experiment with some additional orthogonality penalty formulations. To handle the case of short and fat (rank-deficient) $W$, they propose "double soft orthogonality regularization," which is given by $\lambda(\|W^TW - I\|_F + \|WW^T - I\|_F)$. However, the results of this same study suggest that a regular orthogonal regularizer consistently outperforms a double orthogonality regularizer. Another orthogonality regularizer formulation in [28] is $\lambda\|W^TW - I\|_2$, where Frobenius norm is replaced by 2-norm, which is equal to the largest singular value of $W^TW - I$. To estimate the largest singular value, this method relies on an approximation based on power iteration. However, one problem with this is that power iteration approximation is only good when the largest singular value is sufficiently larger than the rest of the singular values, which may not hold as the training progresses and the largest singular value becomes progressively smaller.

There is limited work on studying orthogonality in the context of quantization. As mentioned in Section I, while [31] mainly studies improved robustness with orthogonality against adversarial examples, they also observed improved quantization with small datasets. As mentioned earlier, both [31] and [32] attribute improved robustness or quantization to the reduced spectral norm of weight matrices as a result of orthogonality constraint. However, orthogonality further minimizes condition number, and [31] and [32] do not study the impact of reduced condition number on quantization. As shown in this article, we found that reduced condition number results in a more pronounced effect on quantization than the spectral norm.

Even though not exactly a regularizer for orthogonality, one approach to reduce perturbation effects (not in the context of quantization, but rather in the context of adversarial attacks or improving accuracy, in general) was to minimize $\|W\|_2$, or the largest singular value of $W$ [29], [58]. This forms an upper bound on the sensitivity of the matrix multiplication result with respect to perturbations on input. However, one problem with this approach is that it provides no bounds with respect to perturbations on $W$. Another problem is that the bound only consider the absolute values of the perturbations on the input and output and does not account for relative values. On the other hand, as explained in Section , orthonormality minimizes bounds on relative sensitivity to perturbations, both for input and $W$. Intuitively, relative values are more crucial in the context of quantization. As an example, a perturbation of 0.05 might be less important when the true value is 10.0, but it is more likely to cause issues if the true value is 0.10.

## III. Method

We use the quantized retraining (fine-tuning) method [25] on the pretrained weights of various convolutional neural network architectures. In this study, we explore the effect of orthogonality constraints by quantizing pretrained weights that are trained with and without orthogonality constraints. We first describe the intuition behind the use of orthogonality for improved quantized models by visiting perturbation theory (see Section III-A), then we move to the quantizer design used in the retraining phase which we keep the same in all the experiments (see Section III-B), and lastly explain how we set the orthogonality regularization while training networks (see Section III-C).

### A. Background on How Orthogonality Lowers the Error Bounds

In this section, we first explain how orthogonality can improve error bounds from perturbation and quantization. First, we note that both fully-connected and convolution layers can be formulated as matrix multiplication [28]. For fully-connected layers, this is trivial. For a convolution layer with weight tensor $C \in R^{C,H,W,K}$, where $C, H, W, K$ denote the number of input channels, filter height, filter width, and the number of output channels, respectively, the computation can be reformulated as a matrix multiplication where the weight matrix is of size $m \times n$, with $m = C \times H \times W$ and $n = K$. In the expressions below, we can associate $A$ with the weight matrix, $x$ as input, and $b$ as the output of a fully-connected layer or convolution layer. For simplicity, we assume the mini-batch size is 1, so $x$ and $b$ are vectors.

Let $Ax = b$ and $(A + \delta A)(x + \delta x) = b + \delta b$ hold, where $A$ is an $n \times n$ non-singular matrix, $x \in R^n$, $b \in R^n$, and $\delta A$, $\delta x$ denote perturbations in $A$ and $x$, respectively. $\delta b$ denotes the resulting perturbations in $b$. $\kappa(A)$ is the condition number of $A$ and is defined as $\kappa(A) = \|A\|\|A^+\|$. $\|A^+\|$ refers to the pseudo-inverse of $A$ and when $A$ is non-singular, it is equal to $\|A^{-1}\|$. This implies

$$\delta b = (\delta A)x + (\delta A)(\delta x) + A(\delta x). \tag{1}$$

Assume $\delta A = 0$ for simplicity. Thus,

$$\begin{aligned} \delta b = A(\delta x) \implies \|\delta b\| &\le \|A\|\|\delta x\| \\ \implies \frac{\|\delta b\|}{\|b\|} \le \frac{\|A\|\|\delta x\|}{\|b\|} &= \frac{\|A\|\|x\|\|\delta x\|}{\|b\|\|x\|} \\ &= \frac{\|A\|\|A^{-1}b\|\|\delta x\|}{\|b\|\|x\|} \le \frac{\|A\|\|A^{-1}\|\|\delta x\|\|b\|}{\|b\|\|x\|} \\ = \kappa(A)\,\frac{\|\delta x\|}{\|x\|} \implies \frac{\frac{\|\delta b\|}{\|b\|}}{\frac{\|\delta x\|}{\|x\|}} &\le \kappa(A) \end{aligned} \tag{2}$$

where norms represent any $p$-norm on matrices or vectors. Above, we used the fact that $\|A\|\|B\| \geq \|AB\|$ and $\|A\|\|x\| \geq \|Ax\|$ for any two matrices $A, B$ and vector $x$ for $p$-norms; these inequalities follow from the definition of $p$-norm. For 2-norm, $\kappa(A) = \sigma_1/\sigma_n$, the ratio of the largest and smallest singular values of $A$. The last inequality implies that relative sensitivity of the result with respect to perturbations on $x$ is bounded by $\kappa(A)$. These bounds are reasonably sharp. For instance, one can show that equality holds when $\delta b = c_1\sigma_n$, $\delta x = c_i\sigma_n u_n$, $b = c_2 v_1$, where $c_1, c_2 \neq 0$, $c_1, c_2 \in R$, $u_n$, $v_n$, and $\sigma_n$ are the $n$'th left singular vector, $n$'th right singular vector, and $n$'th singular value of $A^{-1}$, respectively, and $v_i$ is the first right singular vector of $A^{-1}$.

Now assume $A$ is a full rank non-square matrix, with size $m \times n$, where $m \geq n$. $Ax = b$ and (1) above still hold, but $b = A^{-1}x$ does not hold since $A$ is not invertible when it is not square. Instead, we have $A^TAx = A^Tb$, where $A^TA$ is a symmetric positive definite matrix, and hence invertible because $A$ is full rank. Note that full-rankness is a reasonable assumption since weights are initialized randomly in neural networks. Thus, $x = (A^TA)^{-1}A^Tb$, and following the inequality given in (2) from the second line:

$$\begin{aligned}\frac{\|\delta b\|}{\|b\|} &\leq \frac{\|A\|\|\delta x\|\|(A^TA)^{-1}A^Tb\|}{\|b\|\|x\|} \\ &\leq \frac{\|A\|\|\delta x\|\|(A^TA)^{-1}\|\|A^T\|\|b\|}{\|b\|\|x\|} \\ &= (\kappa(A))^2\frac{\|\delta x\|}{\|x\|} \implies \frac{\frac{\|\delta b\|}{\|b\|}}{\frac{\|\delta x\|}{\|x\|}} \leq (\kappa(A))^2 \end{aligned} \tag{3}$$

where we used the fact that $\|A\|\|A^T\|\|(A^TA)^{-1}\| = (\kappa(A))^2$ [59]. Thus, similar error bounds exist for non-square $A$ that directly depend on $\kappa(A)$. Now assume $\delta x = 0$, $\delta A, \delta b \neq 0$.

For square, nonsingular $A$

$$\begin{aligned}\delta b = A(\delta x) &\implies \frac{\|\delta b\|}{\|b\|} = \frac{\|A(\delta x)\|}{\|b\|} \leq \frac{\|\delta A\|\|x\|}{\|b\|} \\ \frac{\|\delta A\|\|x\|}{\|b\|} &= \frac{\|\delta A\|\|A^{-1}b\|}{\|b\|} \leq \frac{\|\delta A\|\|A^{-1}\|\|b\|}{\|b\|} \\ &= \frac{\|\delta A\|}{\|A\|}\|A^{-1}\|\|A\| \\ \implies \frac{\frac{\|\delta b\|}{\|b\|}}{\frac{\|\delta A\|}{\|A\|}} &\leq \kappa(A). \end{aligned} \tag{4}$$

For non-square, full rank $A$

$$\begin{aligned}\frac{\|\delta A\|\|x\|}{\|b\|} &= \frac{\|\delta A\|\|(A^TA)^{-1}A^Tb\|}{\|b\|} \\ &\leq \frac{\|\delta A\|\|(A^TA)^{-1}\|\|A^T\|\|A\|}{\|A\|} \\ \implies \frac{\frac{\|\delta b\|}{\|b\|}}{\frac{\|\delta A\|}{\|A\|}} &\leq (\kappa(A))^2. \end{aligned} \tag{5}$$

This shows that for the cases above, the relative sensitivity of matrix multiplication with respect to both $A$ and $x$ (where $A$ can be associated with weights and $x$ with inputs) has sharp bounds that are directly a function of $\kappa(A)$. $\kappa(A)$ cannot be smaller than 1 [60], and orthogonality results in $\kappa(A)$ achieving its smallest value 1, thus minimizing the sharp upper bound of the relative sensitivities for the cases above.

For $m < n$, where $A$ is a short and fat matrix and cannot be full rank, no bounds exist for relative sensitivity, and $A$ also cannot be exactly orthogonal. One can still show that the following bound on sensitivity (but not relative sensitivity as used earlier) with respect to perturbations in $x$ exists:

$$\frac{\|A(x+\delta x) - Ax\|}{\|\delta x\|} = \frac{\|A\delta x\|}{\|\delta x\|} \leq \|A\| \tag{6}$$

where $\|A\| = \sigma_1$, for the 2-norm, largest singular value of $A$. As explained below, the orthogonality constraint we use on $A$ also imposes the columns to be normal, which would imply $\sigma_1 = 1$, which prohibits the error bound above from growing also, benefiting this case as well.

### B. Quantizer Design

Uniform quantization allows using integer arithmetic for the multiply-add operations and is the most commonly used design choice [26]. Symmetric quantization refers to the case where 0 in the quantized integer domain is mapped to 0 in the real domain. Furthermore, we use signed integers in quantized representations. As a result, the mapping between real and quantized domains can be described as $x = s * q$ where $x$ is in the real domain and $q$ is in the quantized domain, and $s$ is the scaling factor for mapping. Another design choice is related to the granularity of the scale picked for quantization. To utilize an integer math pipeline, the granularity needs to be per-tensor or per-row for activation tensors and per-tensor or per-column (i.e., per-filter) for weight tensors [26]. Per-tensor quantization results in a good performance and is much cheaper to implement [25], [26], so we use per-tensor scaling both for activations and tensors. We use simulated quantization similar to previous literature [24]–[26], where 16-bit floating-point values are quantized and dequantized with the following steps.

*1) Quantize:* This step consists of scaling, rounding, and clipping. For a per-tensor threshold $t$, scaling occurs such that the largest value that can be represented in the quantized domain ($2^{(b-1)}$ for signed integers for a bit-width of $b$) maps to the threshold value chosen. Scaling factor $s$ is determined by threshold $t$. Then, nearest integer rounding with round-half-to-even is performed. Finally, elements that exist with the largest representable value in the integer domain are clipped to the largest representable integer value.

*2) De-Quantize:* This step reverses the scaling step in quantizing phase above. This is used to emulate the effect of quantization while keeping the scale of the tensor in a high-precision data type. Finally, the process can be defined as

$$\text{quant}(x) = \text{clip}(\text{round}(x/s); -2^{(b-1)}, 2^{(b-1)} - 1) \times s \tag{7}$$

where $s = t/2^{(b-1)}$. We use a straight-through-estimator (STE) to backpropagate through quantization nodes for the retraining phase. We also train the threshold $t$ in log-domain with gradient descent for the input and weight tensors, as described in [25], for each convolution and fully-connected layer.

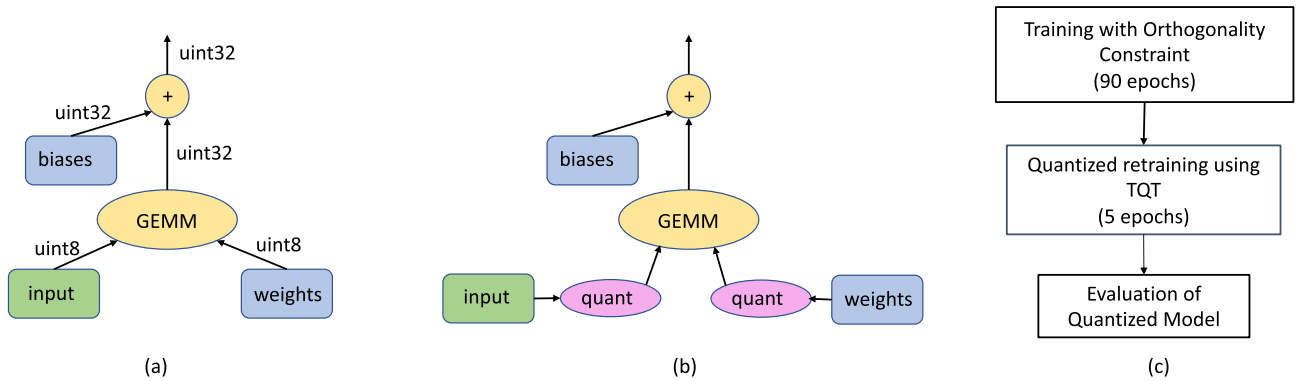



Fig. 1. (a) Integer-arithmetic inference, (b) simulated quantization for training diagrams, and (c) flowchart of the pipeline. GEMM refers to matrix-multiply operation, which includes both fully-connected layers and convolution layers. uint-b refers to unsigned $b$-bit integer quantization. For simulated quantization, all data is kept in the 32-bit floating points format, but quantization is simulated using quantize/de-quantize operations explained in Section III-B. Quantization is only performed for input and weight tensors of fully-connected and convolution layers, which implies that the batch-normalization layer is not quantized. This helps the study focus on the effect of weights trained with orthogonality penalty on quantization.

Fig. 1 describes the emulation of integer quantization during training. It is worth noting that we only use quantization nodes for the input tensors and weight tensors for convolution and fully-connected layers, which means batch-normalization layers are not quantized to keep the focus on the convolution and fully-connected layers for the purpose of understanding the effect of orthogonality of their weight parameters on quantization.

As mentioned, we use TQT [25] as our baseline retraining method. This means that we first train the model with and without orthogonality constraints from scratch and use the TQT retraining method on each of these trained models. The flowchart of the pipeline is given in Fig. 1(c).

### C. Orthogonality Regularization

As mentioned in the related work section, different methods exist for orthogonality regularization. Results from hard constraints from the literature are not impressive, so we only consider soft orthogonality constraints. We use soft orthogonality regularization based on the Frobenius norm, which is given by

$$\lambda \| W^T W - I \|_F. \tag{8}$$

The reason why we choose this regularization is that it gives good performance in terms of accuracy [28], does not rely on iterative approximation as spectral-norm-based regularizations do, and has low-performance overhead during training (results in only a few percent reductions in training throughput). The gradient of this loss term can be expressed by $4\lambda W(W^T W - I)$, which can be seen as a different form of weight decay. One issue about this penalty term [28] is that for $W \in R^{m \times n}$, where $m < n$, $W^T W$ cannot be identity, since $W$ is rank-deficient in that case. However, the double soft orthogonality penalty suggested in [28], which is an alternative that handles rank-deficient cases as explained earlier, performs worse than regular soft orthogonal regularization. Bansal *et al.* [28] suggests that orthogonality restriction for rank-deficient matrices can result in additional bias during training, but results from the same study suggests this bias can still be useful for accuracy. For this reason, we stick with soft orthogonal regularization based on the Frobenius norm. However, we believe there is value in exploring different orthogonality regularization formulations in the context of quantization, as explained later in the article with guidance from experimental results obtained in this study.

## IV. Experiments

We run extensive experiments on three popular convolutional neural networks, namely ResNet50, ResNet101 [61], and VGG19 [62] with ImageNet dataset. For ease of direct reproducibility, we use the hyperparameters suggested in the PyTorch vision library, unless otherwise noted.[1] In the initial stage, we train both models with different weight decay ($w_d$) and orthogonality penalty parameter ($\lambda$) values. We performed a hyperparameter search for ($w_d$), where we start with the default value and decrease or increase by powers of 10. For each of these cases, we also search for ($\lambda$) within values of 0 as well as powers of 10, where ($\lambda$) being 0 means no orthogonality penalty is used. For fairness, each $w_d/\lambda$ configuration is trained exactly once on Resnet50, Resnet101, and VGG19 networks for 90-epochs for the initial training phase. This makes sure that there is no cherry-picking involved in the experiments. Each training run takes roughly 20 h on a DGXV100-128 GB system.

After the initial 90-epoch training sessions with or without orthogonality penalty, we use the TQTs method on all trained models (for each $w_d/\lambda$ configuration). We initialize the trainable threshold parameters for the quantization node using the threshold values obtained by running max calibration [42] both for activation and weights on the trained weights. We run the TQT retraining phase separately for three precision configurations: 8/8, 6/6, and 8/4, where $A/W$ refers to bitwidths of input activation and weight matrix, respectively. For each case, TQT retraining is done for five epochs, as suggested in [25]. Each TQT run (five epochs) takes around 15 min on a DGXV100 system. There are some implementation differences in how

[1] https://github.com/pytorch/vision/tree/master/references

TABLE I

TOP1 AND TOP5 VALIDATION ACCURACIES OF DIFFERENT NETWORK ARCHITECTURES. QUANTIZATION BITWIDTH CONFIGURATION IS REPRESENTED AS *A*/*W*, WHERE *A* DENOTES ACTIVATION BIT-WIDTH AND *W* DENOTES THE BIT-WIDTH OF WEIGHT PARAMETERS. THE FIRST ROW FOR EACH *A*/*W* CASE SHOWS $w_d$ THAT RESULTS IN THE HIGHEST FP32 ACCURACY WITHOUT ORTHOGONALITY REGULARIZATION. THE SECOND ROW SHOWS THE CASE WITH ORTHOGONALITY REGULARIZATION, WITH $w_d/\lambda$ THAT RESULTS IN THE CLOSEST FP32 ACCURACY TO THE FIRST ROW. NOTE THAT DESPITE FP32 ACCURACIES BEING ALMOST IDENTICAL FOR EACH *A*/*W* CASE, QUANTIZED ACCURACY IS IMPROVED WITH ORTHOGONALITY PENALTY

| ResNet50 | $w_d$ | $\lambda$ | FP32 accuracy before TQT | Quantized accuracy after TQT |
|---|---|---|---|---|
| 8/8 no Orthogonal regularizer | $10^{-4}$ | - | 75.37 / 92.58 | 75.09 / 92.39 |
| 8/8 with Orthogonal regularizer | $10^{-5}$ | $10^{-5}$ | 75.88 / 92.89 | **75.68 / 92.79** |
| 6/6 no Orthogonal regularizer | $10^{-4}$ | - | 75.37 / 92.58 | 73.25 / 91.24 |
| 6/6 with Orthogonal regularizer | $10^{-5}$ | $10^{-6}$ | 75.38 / 92.50 | **74.27 / 91.77** |
| 8/4 no Orthogonal regularizer | $10^{-4}$ | - | 75.37 / 92.58 | 69.53 / 89.11 |
| 8/4 with Orthogonal regularizer | $10^{-5}$ | $10^{-7}$ | 74.64 / 91.74 | **71.88 / 90.20** |
| ResNet101 | $w_d$ | $\lambda$ | FP32 accuracy before TQT | Quantized accuracy after TQT |
| 8/8 no Orthogonal regularizer | $10^{-4}$ | - | 77.09 / 93.57 | 76.77 / 93.35 |
| 8/8 with Orthogonal regularizer | $10^{-4}$ | $10^{-5}$ | 77.57 / 93.85 | **77.30 / 93.72** |
| 6/6 no Orthogonal regularizer | $10^{-4}$ | - | 77.09 / 93.57 | 75.43 / 92.59 |
| 6/6 with Orthogonal regularizer | $10^{-5}$ | $10^{-6}$ | 76.90 / 93.26 | **76.12 / 92.77** |
| 8/4 no Orthogonal regularizer | $10^{-4}$ | - | 77.09 / 93.57 | 71.78 / 90.50 |
| 8/4 with Orthogonal regularizer | $10^{-5}$ | $10^{-6}$ | 76.90 / 93.26 | **73.79 / 91.51** |
| VGG19 | $w_d$ | $\lambda$ | FP32 accuracy before TQT | Quantized accuracy after TQT |
| 8/8 no Orthogonal regularizer | $10^{-4}$ | - | 72.31 / 90.89 | 72.40 / 90.86 |
| 8/8 with Orthogonal regularizer | $10^{-4}$ | $10^{-5}$ | 72.68 / 91.28 | **72.80 / 91.26** |
| 6/6 no Orthogonal regularizer | $10^{-4}$ | - | 72.31 / 90.89 | 71.62 / 90.46 |
| 6/6 with Orthogonal regularizer | $10^{-4}$ | $10^{-5}$ | 72.80 / 91.26 | **72.17 / 90.85** |
| 8/4 no Orthogonal regularizer | $10^{-4}$ | - | 72.31 / 90.89 | 71.24 / 90.29 |
| 8/4 with Orthogonal regularizer | $10^{-4}$ | $10^{-5}$ | 72.68 / 91.28 | **71.79 / 90.74** |

we apply the TQT method compared to [25]. Namely, they switch to Adam optimizer for the TQT phase and implement a sophisticated learning rate schedule during retraining. For simplicity, we use a stochastic gradient descent optimizer during quantized retraining similar to the initial model training phase and use the same hyperparameters as if the training was the continuation of the initial 90-epoch training phase. For quantization threshold, we still use a separate learning rate similar to [25], a learning rate of $10^{-4}$ for quantization thresholds during retraining. Another difference in our implementation is that we initialize quantization thresholds for input activations for retraining using the max-calibration (MAX) method explained in [25] as opposed to the KL-J method, where MAX uses 50 unlabeled images randomly selected from the validation set and computes maximum absolute value within the resulting input activation for initializing quantization thresholds for activation. This is done for simplicity and ease of reproducibility. Quantization thresholds for weights also use the MAX method, which is similar to [25]. We use per-tensor quantization both for activations and weights, instead of per-channel.

## V. RESULTS

This section presents the results of various networks with (see Section V-A) and without TQT (see Section V-C). We investigate the training stability by reporting the training curves in Section V-B and provide an in-depth analysis in Section V-D.

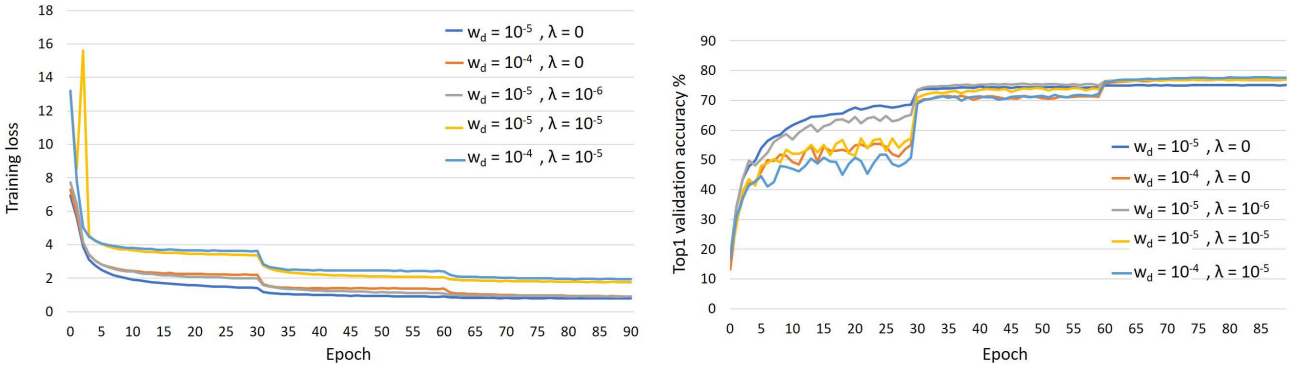
### A. Experiments With TQT

Results are summarized in Table I for ResNet50, ResNet101, and VGG19 models. In Table I, for each model and *A*/*W* configuration, the first two rows show the cases without and with orthogonality regularizer, respectively, where their pre-TQT FP32 accuracies are almost identical. Sometimes, one of them achieves a slightly better score than the other before quantization. With an orthogonality regularizer, quantized accuracies improve noticeably, despite FP32 accuracies being very close. For ResNet50, for 8-bit activation width, and 4 bit for weight parameters, FP32 accuracy for the network with no orthogonal regularizer achieves 75.37% Top 1 accuracy versus 74.64% Top 1 accuracy achieved by the network with the orthogonal regularizer. On the other hand, when these networks are quantized, the network trained with the orthogonal regularizer achieves 71.88% Top 1 accuracy, a significantly better result than the 69.53% achieved by the network which is trained without the orthogonal regularizer. The results are consistent for ResNet50, ResNet101, and VGG19 models for 8/8, 6/6, 8/4 bitwidth configurations as can be seen from Table I. Later, we analyze where this difference is coming from.

We also see that to achieve optimal results, the orthogonality penalty often requires reducing the weight decay penalty. In general, there is a balance between weight decay, which reduces spectral norm, and orthogonality penalty, which reduces the condition number, of weight matrices. This balance is further explored in Section . It is also important to note that more aggressive quantization gives better results with reduced regularization, both for weight decay and orthogonality penalties. This is expected as quantized training (TQT) also acts as a form of a regularizer by itself, and it may not favor additional regularizers being very strict.

### B. Training Curves

Fig. 2 shows the evolution of Resnet101 training for different configurations of weight decay ($w_d$) and orthonormality




(a) (b)

Fig. 2. Evolution of training for Resnet101 for varying combinations of weight decay ($w_d$) and orthonormality penalties ($\lambda$). Training loss for higher $\lambda$ is higher due to additional terms in the loss function. In the validation accuracy curve, the regularization effect of the orthogonality penalty is more clearly observed, and its effect on the evolution of validation accuracy seems similar to that of increasing weight decay. (a) Evolution of training loss. (b) Evolution of Top1 validation accuracy.

TABLE II

IN THIS EXPERIMENT, QUANTIZATION IS PERFORMED RIGHT AFTER THE INITIAL 90-EPOCH TRAINING PHASE, E.G., WITH NO ADDITIONAL QUANTIZED RETRAINING. QUANTIZATION BITWIDTH CONFIGURATION IS REPRESENTED AS $A/W$, WHERE A DENOTES ACTIVATION BIT-WIDTH AND W DENOTES THE BIT-WIDTH OF WEIGHT PARAMETERS. THE CORRESPONDING $w_d$ IS SET TO $10^{-4}$ AND $\lambda$ VALUES ARE SHOWN. EXPERIMENTS ARE REPEATED THREE TIMES AND MEDIAN VALUES AND MAXIMUM DEVIATION FROM THE MEDIAN VALUES ARE PRESENTED. THE IMPROVEMENTS COMING FROM THE ORTHOGONAL REGULARIZER ARE EVEN MORE SIGNIFICANT IN THIS SETTING. WE ALSO FIND THAT RESULTS FROM NETWORKS TRAINED WITH ORTHOGONAL REGULARIZER ARE MORE ROBOUST SHOWING SMALLER DEVIATION FROM THE MEDIAN VALUES

| Resnet50v1.5 | $\lambda$ | FP32 accuracy before quantization | Quantized accuracy |
|---|---|---|---|
| 8/8 no Orthogonal regularizer | 0 | 76.29±0.28 / 93.06±0.13 | 76.12±0.24 / 92.98±0.10 |
| 8/8 with Orthogonal regularizer | $10^{-5}$ | 76.52±0.05 / 93.01±0.04 | 76.31±0.10 / 92.89±0.05 |
| 6/6 no Orthogonal regularizer | 0 | 76.29±0.28 / 93.06±0.13 | 74.19±0.37 / 91.95±0.23 |
| 6/6 with Orthogonal regularizer | $10^{-5}$ | 76.52±0.05 / 93.01±0.04 | 74.93±0.14 / 92.21±0.15 |
| 5/5 no Orthogonal regularizer | 0 | 76.29±0.28 / 93.06±0.13 | 65.36±0.51 / 86.78±0.33 |
| 5/5 with Orthogonal regularizer | $10^{-5}$ | 76.52±0.05 / 93.01±0.04 | 70.58±0.33 / 89.62±0.17 |

penalties ($\lambda$). Training loss for higher $\lambda$ is higher due to additional term in the loss function. There is a spike observed for the setting of $w_d = 10^{-5}$ and $\lambda = 10^{-5}$ which suggests that one can potentially come up with a more carefully tailored orthogonality penalty schedule during training, as opposed to constant $\lambda$. However, we did not find this occasional spike that happens at the early stage of the training affecting the validation scores. As can be seen, training is very stable with the training loss decreasing at every epoch. The learning rate is halved at epochs 30 and 60. In the validation accuracy curve, the regularization effect of the orthogonality penalty is more clearly observed, and its effect on the evolution of validation accuracy seems similar to that of increasing weight decay. Even though the effect on the curves seems similar for these regularizers, their effect on quantized accuracy is different as will also be analyzed in Section V-D.

### C. Experiments Without TQT

Table II shows results for a setup that is different from previously presented. We look at the impact of the orthogonalization penalty on quantized accuracy when no additional quantization-aware retraining is used. In other words, the model is directly quantized by post-training quantization (using MAX calibration as explained in the main text) after the 90-epoch training session. These results are obtained with Resnet50v1.5. For each configuration, three experiments are run and the median is shown, together with the maximum deviation from the median. In this scenario, introducing an orthogonality penalty improves quantization by a lot more than in the case where TQT is used since without TQT there is a lot more to gain from orthogonality. Namely, running TQT helps improve quantized accuracy by adjusting the ranges, and without TQT, the orthogonality penalty is the only mechanism that can adjust the weights so that the impact of perturbation due to quantization is reduced. Notice that FP32 accuracies are very similar with and without the orthogonality penalty, but the gap between quantized accuracies widens especially for more aggressive quantization. Specifically, for 5-bit precision, the quantized Top 1 accuracy is 65.36% for the model without orthogonality penalty, whereas it is 70.58% for the model with orthogonality penalty, showing a significant accuracy gain of the orthogonality.

In this setup, we also provide the maximum deviation from the median among the three runs of experiments. We find that models trained with an orthogonal regularizer output more robust results and deviate less compared to the networks trained without an orthogonal regularizer. This holds for both FP32 accuracy before quantization and quantized accuracy. We do not aggressively experiment with this setup and only

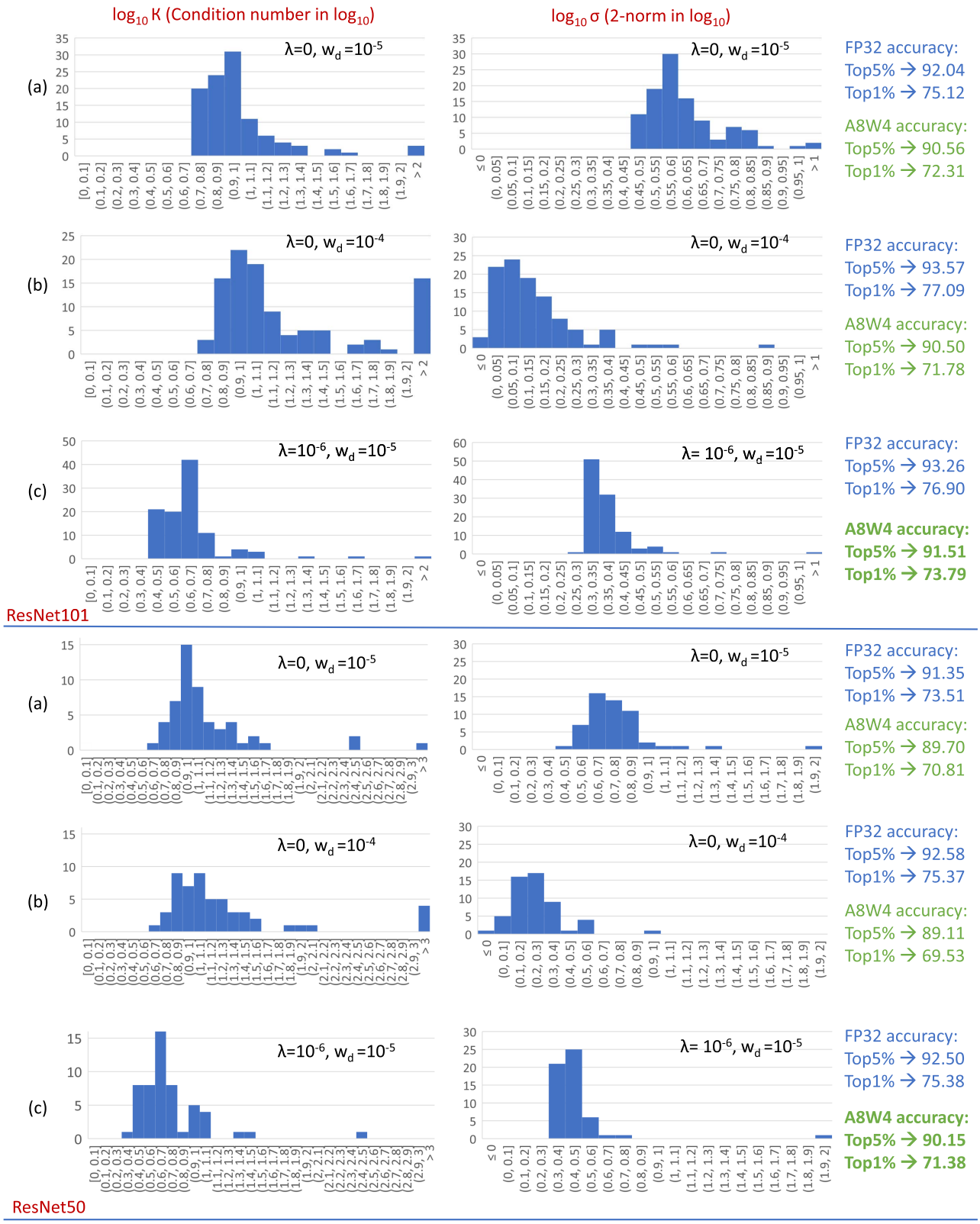



Fig. 3. Histogram of condition numbers and histogram of 2-norms of weight matrices for all convolution or fully connected layers for ResNet101 and ResNet50 after training and before TQT. From (a) to (b), the weight decay coefficient increases, while the orthonormality penalty stays 0, which reduces 2-norms (spectral norms) significantly but slightly increases the condition numbers (the distribution shifts to the right). This increased regularization increases FP32 validation accuracy (before quantization), but reduces quantized accuracy (obtained with TQT). On the other hand, adding the orthonormality penalty on top of (a) which is presented in (c) does not reduce 2-norms as much as (b), but substantially reduces condition numbers of weight matrices, also giving rise to higher quantized accuracies.

run with ResNet50, since this setup is less challenging to showcase improvements.

### D. Analysis

Table I shows that despite FP32 accuracies in the non-quantized case can be very close, the orthogonality penalty improves quantized accuracy compared to the case with no orthogonality penalty. This shows that not all regularizers are created equally when it comes to quantization, and two regularizers that result in similar FP32 accuracies may not perform the same after quantization. We show further analysis below to explain where this difference stems from.

We analyze how the distribution of condition number and spectral norm of weight matrices vary after the initial 90-epoch training stage with different orthogonality regularizer ($\lambda$) and weight decay ($w_d$) configurations, for Resnet101 and Resnet50 in Fig. 3. We find that behavior is consistent between the two models under these two different regularizers. As we increase the weight decay coefficient from Fig. 3(a) and (b) ($w_d$ increases to $10^{-4}$ from $10^{-5}$) without the orthogonal regularizer, spectral norms reduce substantially, while condition numbers stay similar or slightly increase. With the original weight decay parameter but with orthogonality penalty introduced, from Fig. 3(a)–(c), condition numbers reduce substantially, resulting in optimal quantized accuracy among these three cases. This shows that the reduction in condition numbers is the cause for the improved quantization and not the spectral norm. This finding aligns with the theoretical background given in the Methods section on how a lower condition number reduces the bounds on the perturbation error, wherein our case quantization represents the perturbation.

This finding is important and an important contribution to this work as it shows for the first time, in contrast to existing literature, improved quantization due to orthogonality in neural networks is because of the reduction in condition number of weight parameters, and not due to the reduction in the spectral norm. This behavior is in line with the results presented in Table I. As shown in Table I, ImageNet Top5 accuracy loss for 4- to 8-bit quantization is reduced by up to 7% for Resnet50, and up to 10% for Resnet101, compared to trained quantization with no orthogonality penalty. This shows that quantized training does not quite have the same effect as the orthogonality penalty, and the orthogonality penalty still improves quantization when combined with a quantized training method.

## VI. Conclusion

In this work, we theoretically explain and experimentally show that the ability of orthogonality constraint to reduce the condition number is the key feature that improves the accuracy of the quantized model, more so than the reduction in spectral norms of weights. To the best of our knowledge, this is shown for the first time and in contrast to existing literature. We experiment with Resnet50, Resnet101, and VGG19 and combine the orthogonality penalty with a state-of-the-art TQT quantized retraining method for quantization. Improvements are consistent for all networks for every bit of setting. This shows that quantized training does not capture all the benefits of the orthogonality penalty.

## References


[1] S. Chetlur *et al.*, "CuDNN: Efficient primitives for deep learning," 2014, *arXiv:1410.0759*.

[2] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 G-ops/s mobile coprocessor for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 682–687.

[3] A. Dundar, J. Jin, V. Gokhale, B. Martini, and E. Culurciello, "Memory access optimized routing scheme for deep networks on a mobile coprocessor," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2014, pp. 1–6.

[4] J. Jin, V. Gokhale, A. Dundar, B. Krishnamurthy, B. Martini, and E. Culurciello, "An efficient implementation of deep convolutional neural networks on a mobile coprocessor," in *Proc. IEEE 57th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2014, pp. 133–136.

[5] A. Dundar, J. Jin, B. Martini, and E. Culurciello, "Embedded streaming deep neural networks accelerator with applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 7, pp. 1572–1583, Jul. 2017.

[6] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[7] J. Jin, A. Dundar, and E. Culurciello, "Flattened convolutional neural networks for feedforward acceleration," 2014, *arXiv:1412.5474*.

[8] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[9] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[10] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[11] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.

[12] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[13] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *Proc. Int. Conf. Learn. Represent.*, 2018.

[14] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," 2017, *arXiv:1711.05852*.

[15] H. Yin *et al.*, "Dreaming to distill: Data-free knowledge transfer via DeepInversion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8715–8724.

[16] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2270–2278.

[17] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1389–1397.

[18] G. Huang, S. Liu, L. V. D. Maaten, and K. Q. Weinberger, "CondenseNet: An efficient DenseNet using learned group convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2752–2761.

[19] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4107–4115.

[20] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," 2016, *arXiv:1605.04711*.

[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* New York, NY, USA: Springer, 2016, pp. 525–542.

[22] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.

[23] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learn. Unsupervised Feature Learn. Workshop, Adv. Neural Inf. Process. Syst. (NIPS)*, 2011.

[24] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[25] S. R. Jain, A. Gural, M. Wu, and C. H. Dick, "Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks," 2019, *arXiv:1903.08066*.
[26] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," 2020, *arXiv:2004.09602*.
[27] D. Xie, J. Xiong, and S. Pu, "All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6176–6185.
[28] N. Bansal, X. Chen, and Z. Wang, "Can we gain more from orthogonality regularizations in training deep networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4261–4271.
[29] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[30] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 854–863.
[31] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," in *Proc. Int. Conf. Learn. Represent.*, 2019. [Online]. Available: https://openreview.net/forum?id=ryetZ20ctX
[32] M. Alizadeh, A. Behboodi, M. van Baalen, C. Louizos, T. Blankevoort, and M. Welling, "Gradient $\ell_1$ regularization for quantization robustness," in *Proc. Int. Conf. Learn. Represent.*, 2020. [Online]. Available: https://openreview.net/forum?id=ryxK0JBtPr
[33] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *Proc. Int. Conf. Learn. Represent.*, 2016.
[34] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016, *arXiv:1602.02830*.
[35] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.
[36] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 365–382.
[37] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "WRPN: Wide reduced-precision networks," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[38] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave Gaussian quantization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5918–5926.
[39] S. Jung *et al.*, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4350–4359.
[40] M. Nagel, R. A. Amjad, M. Van Baalen, C. Louizos, and T. Blankevoort, "Up or down? Adaptive rounding for post-training quantization," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 7197–7206.
[41] Y. Li *et al.*, "BRECQ: Pushing the limit of post-training quantization by block reconstruction," in *Proc. Int. Conf. Learn. Represent.*, 2020.
[42] S. Migacz, "8-bit inference with tensorrt," in *Proc. GPU Technol. Conf.*, 2017, vol. 2, no. 4, p. 5.
[43] J. Choi, Z. Wang, S. Venkataramani, P. I.-Jen Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," 2018, *arXiv:1805.06085*.
[44] C. Baskin *et al.*, "NICE: Noise injection and clamping estimation for neural network quantization," 2018, *arXiv:1810.00162*.
[45] A. Goncharenko, A. Denisov, S. Alyamkin, and E. Terentev, "Fast adjustable threshold for uniform neural network quantization (winning solution of LPIRC-II)," *Int. J. Comput. Inf. Eng.*, vol. 13, no. 9, pp. 495–499, 2019.
[46] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.
[47] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1310–1318.
[48] V. Dorobantu, P. Andre Stromhaug, and J. Renteria, "DizzyRNN: Reparameterizing recurrent neural networks for norm-preserving backpropagation," 2016, *arXiv:1612.04035*.
[49] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1120–1128.
[50] Z. Mhammedi, A. Hellicar, A. Rahman, and J. Bailey, "Efficient orthogonal parametrisation of recurrent neural networks using householder reflections," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2401–2409.
[51] E. Vorontsov, C. Trabelsi, S. Kadoury, and C. Pal, "On orthogonality and learning recurrent networks with long term dependencies," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3570–3578.
[52] S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas, "Full-capacity unitary recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4880–4888.
[53] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," 2013, *arXiv:1312.6120*.
[54] D. Mishkin and J. Matas, "All you need is a good init," 2015, *arXiv:1511.06422*.
[55] K. Jia, D. Tao, S. Gao, and X. Xu, "Improving training of deep neural networks via singular value bounding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4344–4352.
[56] M. Harandi and B. Fernando, "Generalized BackPropagation, Étude de cas: Orthogonality," 2016, *arXiv:1611.05927*.
[57] M. Ozay and T. Okatani, "Optimization on submanifolds of convolution kernels in CNNs," 2016, *arXiv:1610.07008*.
[58] Y. Yoshida and T. Miyato, "Spectral norm regularization for improving the generalizability of deep learning," 2017, *arXiv:1705.10941*.
[59] E. Darve and M. Wootters, *Numerical Linear Algebra With Julia*, vol. 172. Philadelphia, PA, USA: SIAM, 2021.
[60] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD, USA: Johns Hopkins Univ. Press, 2013.
[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[62] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015.

**Sukru Burc Eryilmaz** received the B.Sc. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey, and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, USA, in 2011, 2013, and 2017 respectively.

He is currently a Senior Architect of deep learning architecture with NVIDIA Corporation, Santa Clara, CA, USA, where he works on optimizing deep learning hardware and software, both at single-chip scale and supercomputer scale. His research interests include performance analysis and optimization of machine learning workloads, distributed neural network training, AI systems HW/SW co-design, and numerical linear algebra.

**Aysegul Dundar** received the B.Sc. degree in electrical and electronics engineering from Boğaziçi University, İstanbul, Turkey, in 2011, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 2016, under the supervision of Prof. E. Culurciello.

She is currently an Assistant Professor of computer science with Bilkent University, Ankara, Turkey. In CVPR 2018, she won first place in the Domain Adaptation for Semantic Segmentation Competition in the Workshop on Autonomous Vehicle challenge.