UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

Bachelor Project

Regularization by quantization error maximization

Exploring the effects of Quantization Aware Training on generalization

Author: Jonathan Wenshøj (JLV849)

Supervisors: Raghavendra Selvan, Bob Pepin

Date: 10/06/2024

Abstract

As deep neural networks (DNNs) grow increasingly large, efficient deployment becomes crucial, especially for small devices. One of the most popular solutions to this problem is quantizing the model. For low bit ($\leq 4-$bit) quantization, special training is required to maintain accuracy, usually in the form of Quantization Aware Training (QAT). Several authors have surprisingly reported improved accuracy after this model compression method, yet little literature explores this connection between quantization and generalization. In this work, we investigate the effects of QAT on generalization and theoretically show that QAT for a linear model is regularizing by maximizing the quantization error! We then empirically show that QAT causes the same behavior in DNNs, acting as an implicit regularizer scaling with step size, which results in sparse models and lower generalization gaps.

Contents

1 Introduction

Deep neural networks (DNNs) have achieved state-of-the-art results across various tasks, ranging from computer vision to natural language processing. However, the superior accuracy of DNNs seems correlated to the size of the model, increasing their cost and making them difficult to deploy on resource-constrained devices such as mobile phones. As a result, model compression techniques have gained significant attention as a means to reduce the memory footprint, latency, and computational requirements of DNNs while maintaining accuracy.

One widely used approach for model compression is quantization, which involves representing the weights and/or activations of a DNN using low-bit precision values instead of full-precision floating-point values. While quantization to low bits ($\leq 4-$ bit) leads to the most substantial compression rates, it often results in a significant drop in model accuracy. To mitigate this accuracy degradation, quantization-aware training (QAT) has emerged as an effective method. QAT exposes the DNN to the effects of quantization at training time, allowing the model to adapt to the low-bit representations. This often results in little to no performance degradation once quantized, and even in some cases accuracy improvements.

Despite the widespread usage of QAT methods, their impact on model generalization is relatively unexplored in the literature. In this work, we provide a theoretical and empirical analysis that sheds light on the implicit regularization effects of QAT and the implications for the generalization properties of quantized DNNs.

The main contributions are:

- We theoretically show that for a linear model, QAT is equivalent to optimizing a non-quantized model, with implicit regularization through maximizing quantization error. Consequently, the error is not well-modelled as independent noise.
- We empirically show that QAT implicitly regularizes, scaling with the step size, resulting in sparse models with a lower generalization gap compared to a non-quantized model.
- We empirically show that the features of the quantized model approximates those of a non-quantized model.
- We identify several sources of regularization coming from QAT: the STE, rounding method, step size, how the integer range is set, and the discreteness of the quantization operation itself.

2 Background

2.1 Supervised learning

In supervised learning, we are given a dataset S consisting of n pairs of i.i.d. samples from an unknown distribution of $p(X, Y)$. From a hypothesis set $\mathcal{H}$ we want to find a $h \in \mathcal{H}$

which minimizes $\mathcal{L}(h) = \mathbb{E}[l(h(X), Y)]$. For DNNs, we find a $f(\cdot) \in \mathcal{H}_{\mathbf{w}}$ parameterized by $\mathbf{w}$. Since we do not know $p(X, Y)$, we split S into two sets, S_{train} and S_{test}. S_{test} can then be used to approximate $\mathcal{L}(h)$. A model that performs well on S_{test} is said to generalize well, meaning it has learned features from S_{train}, which transfers well to S_{test}. We can define the generalization gap as $\mathcal{L}(h) - \hat{\mathcal{L}}(h)$; this value being low will mean that h is showing good generalization properties.

A hypothesis set might contain many h, which we a priori deem uninteresting. Excluding certain regions of the hypothesis space is termed regularization. Regularization comes in two forms: implicit and explicit. Implicit regularization occurs, for example, through the optimizer, the parameter initialization, or even the network architecture itself. Explicit regularization adds a term to the loss function, which is then optimized upon; examples include L_1 and L_2 regularization.

For DNNs, there is an intimate connection between the optimizer and the objective, which often interacts in ways that cause implicit regularization.

2.2 Quantization

Lets first establish the terminology used:

Quantization bin: All values in this range are quantized to a specific quantization level.

Quantization level: The value to which the bin is dequantized.

Quantization threshold: The threshold between two quantization bins.

Step size: The distance between two quantization thresholds.

Quantization maps a series of floating point values $\mathbf{x}$ in a range $[\beta, \alpha]$ into an integer grid (The quantization levels), which can then be de-quantized to $\hat{\mathbf{x}} \approx \mathbf{x}$. The integer grid is often chosen based on the floating point range. If the floats are all unsigned, a grid $\{0, \ldots, 2^b - 1\}$, where b is the number of bits in the quantizer, can be chosen, or if the floating point series is signed, we can set the grid to $\{-2^{b-1}, \ldots, 2^{b-1} - 1\}$. Depending on if one wants symmetry around 0, one can choose a narrow range $\{-(2^{b-1} - 1), \ldots, 2^{b-1} - 1\}$.

A uniform affine quantizer, widely used for quantizing DNNs due to its simplicity and hardware efficiency, comes in two forms: the asymmetric and the symmetric (Nagel, Fournarakis, Amjad, et al., 2021). The asymmetric is given by:

$$\bar{\mathbf{x}} = \text{clamp}\left(\left\lceil \frac{\mathbf{x}}{s} \right\rfloor + z, n, p\right) \tag{1}$$

$$\hat{\mathbf{x}} = q(\mathbf{x}) = s(\bar{\mathbf{x}} - z) \tag{2}$$

where the clamp$(\cdot, a, b)$ operation restricts the values to within the range $[a, b]$, $\lceil \cdot \rfloor$ is a rounding operation, s is the scale factor and z the zero-point, n and p the lower and upper quantization grid limits.

The symmetric quantizer is a special case of the asymmetric, given by $z = 0$:

$$\bar{\mathbf{x}} = \text{clamp}\left(\left\lceil \frac{\mathbf{x}}{s} \right\rfloor, n, p\right) \tag{3}$$

$$\hat{\mathbf{x}} = q(\mathbf{x}) = s\bar{\mathbf{x}} \tag{4}$$

When we quantize a vector $\mathbf{x}$, we incur quantization error which is defined as:

$$\boldsymbol{\Delta} = \mathbf{x} - \hat{\mathbf{x}} \tag{5}$$

The error has two sources: clipping and rounding. There is a trade-off between the two; if we increase the scale factor, we reduce the clipping error, but we then increase the rounding error since the rounding error is in the range $[-\frac{s}{2}, \frac{s}{2}]$ (Nagel, Fournarakis, Amjad, et al., 2021).

There exist multiple ways to set s; for example, it can be a learnable parameter such as with Learned Step Size Quantization (LSQ) (Esser et al., 2020). Another way to set s is to base it on the dynamic range of $\mathbf{x}$, thereby removing the clipping error. For unsigned integers, s is then given by:

$$s = \frac{\alpha - \beta}{2^b - 1} \tag{6}$$

Moreover, for signed integers with symmetry around 0:

$$s = \frac{\max(|\alpha|, |\beta|)}{2^{b-1} - 1} \tag{7}$$

The precision of quantization can be further controlled by adjusting the number of scale factors used to quantize a vector, $\mathbf{x}$. There are several approaches: using a single scale factor for the entire tensor, assigning a scale factor per channel, or having different scale factors for subblocks within $\mathbf{x}$. Although these methods enhance the precision of quantization, they introduce a trade-off. The increased precision comes at the cost of additional overhead in terms of computation required to calculate these scale factors and increased storage (Nagel, Fournarakis, Amjad, et al., 2021).

2.3 Quantization of DNNs

Generally, two parts of a DNN are quantized to lower bit precision: the parameters and/or the activations. This results in DNNs with reduced storage requirement, lowered memory usage, latency, and power consumption, and faster inference if both activations and parameters are quantized since the computations in the network is done using efficient integer arithmetic (Nagel, Fournarakis, Amjad, et al., 2021).

Overall, there are two approaches to quantization: Post-training quantization (PTQ) and quantization-aware training (QAT). PTQ aims to approximate the full precision model with as little loss in performance as possible. PTQ works by quantizing a model to a lower bit after training the model using full precision weights. Usually, this requires various correction measures to avoid loss in performance and is often restricted to 8-bit and above (Krishnamoorthi, 2018), though it is straightforward to execute. The second approach, QAT, covers several methods, what connects them is that there is some emulating of the effects of quantization during training. The model can then learn to adapt to this process, generally resulting in little to no degradation in performance after quantization, even at low bit (Krishnamoorthi, 2018). The downside is the coupling between quantization and training, which creates a less efficient training loop. The model is only resistant to quantization at the bit rate it was trained at. Often, the methods are combined, and the model is first trained with full precision weights and then fine-tuned using QAT.

The most common form of QAT works by having floating point weights $\mathbf{w}$ (The latent weights) work as gradient accumulators (Jacob et al., 2017) (Krishnamoorthi, 2018). The forward pass is performed using the de-quantized weights $\hat{\mathbf{w}}$. During the backward pass, the loss is taken with respect to $\mathbf{w}$, and then the latent weights are updated. This means that the gradient for the latent weights during QAT is given by:

$$\frac{\partial \mathcal{L}^q(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}(q(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}(q(\mathbf{w}))}{\partial q(\mathbf{w})} \cdot \frac{\partial q(\mathbf{w})}{\partial \mathbf{w}} \tag{8}$$

A problem with the above formulation is that the gradient of the rounding operation used in the quantizer is mostly zero, causing the last term to interrupt gradient-based learning. A popular solution to this problem is using the Straight-Through Estimator (STE) (Bengio et al., 2013) defined as:

$$\frac{\partial q(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{1}_{n \leq \mathbf{w}/s \leq p} \tag{9}$$

where $\mathbf{1}$ is the indicator function, n and p the clamping boundaries and s the scale factor. This means the gradient is estimated to be one if the scaled weights are within the integer grid limits and zero otherwise.

2.4 Quantization and generalization

Several works have reported accuracy gains as a side product of quantization (Courbariaux, Bengio, et al., 2016) (Xu et al., 2018) (Mishchenko et al., 2019) (Yin et al., 2020) (Chen et al., 2021), linking generalization and quantization. A common denominator among these works is some form of exposure to the effects of quantization during training. While works investigating the connection between generalization and quantization are limited, they have generally focused on viewing quantization as a form of implicit regularization induced by the quantization process. These previous works can broadly be separated into two approaches that revolve around the quantization error: Modelling the error as multiplicative or additive noise.

From the earliest work on quantization of DNNs such as BinaryConnect (Courbariaux, Bengio, et al., 2016) and Binarized Neural Networks (Courbariaux, Hubara, et al., 2016), there has been drawn a link from binary quantization to regularization by multiplicative noise, likening it to DropConnect (Wan et al., 2013) and Dropout (Srivastava et al., 2014), two popular methods for regularization. While both works empirically show improved performance for binary networks, they provide no theoretical analysis of the regularization effects of quantization. Additionally, the results are limited to binary quantization of weights and activations simultaneously.

The second interpretation of quantization error is to model the error as additive noise. In QReg (AskariHemmat, Hemmat, et al., 2022) and QGen (AskariHemmat, Jeddi, et al., 2024), this is the approach taken. They theoretically show that the additive noise model results in a gradient norm regularizer. Additionally, they empirically show an improved generalization gap, robustness, and flatness for several different bit widths, viewing quantization in QAT as an implicit regularization that inversely scales with the number of bits.

However, there are some concerns with the theoretical approach taken. It builds mainly on an approach from (Alizadeh et al., 2020), who argue that since the quantization error lies in the range $[-\frac{s}{2}, \frac{s}{2}]$, the error can be modeled as an L_∞-bounded perturbation within that range. By managing the impact of this perturbation on the loss, the model should theoretically maintain its robustness to PTQ for all bit widths with quantization error bounded by the L_∞ perturbation. However, the method did not outperform QAT in terms of accuracy. This suggests the method does not find better minima for quantization compared to QAT, perhaps since, at lower bits, the noise magnitude is large, making the loss landscape sub-optimal.

In QReg and QGen, they take the approach from (Alizadeh et al., 2020) and combine it with an insight from signal processing, which states that the quantization error can be modeled as independent uniform noise added to the signal (Widrow, 1956). They then model the quantization error during QAT as independent additive noise to the weights in their theoretical analysis. In (Alizadeh et al., 2020), the model is only assumed to be PTQed, and therefore, the quantization error only needs to be bounded by the error induced by PTQ. Hence, the independence of the error becomes irrelevant. While during QAT, it is unclear if the noise is i.i.d. after the first iteration. Additionally, the uniform independent noise model from signal processing is not valid when the step size gets large relative to the signal, a case which is common for low-bit quantization (Marco et al., 2005).

While most existing works fall into the categories of multiplicative or additive noise interpretations, (Zhang et al., 2022) showed theoretically that a binary neural network has a lower capacity compared to a floating point and empirically confirmed that this results in a lower generalization gap for the binary network.

Lastly there is work looking at the effect of the STE, and the phenomena of oscillations (Latent weights which consistently changes between two quantization levels), while often seen as a harmful phenomena (Défossez et al., 2022) (Gupta et al., 2023), the oscillations might also contribute a regularizing effect.

3 Method

3.1 Quantization of weights

To investigate the generalization properties of a quantized DNN, we focus on quantizing the weights during QAT with a per-tensor scale factor. QAT since it exposes the loss function to quantization error and quantization of weights to isolate the effects and ease the theoretical investigation. Since QAT is often applied after PTQ, outliers in the weights might exist, which makes it less desirable to use the dynamic range of the weights for the scale factor. However, to properly understand the regularizing effects of QAT, we train from scratch with QAT, meaning we won't have any issues with clipping outliers. So, a dynamic range is chosen to simplify things. Given that the weights are signed, we use a signed integer grid and narrow it to keep it symmetric around zero. It's important to note that this reduces the number of quantization levels by one, transforming, for instance, 2-bit into ternary. Additionally, letting $z = 0$, we can then define the uniform symmetric quantizer for the weights:

$$\bar{\mathbf{w}} = \left\lceil \frac{\mathbf{w}}{s} \right\rfloor = \left\lceil \frac{(2^{b-1}-1)\mathbf{w}}{\max(|\mathbf{w}|)} \right\rfloor \tag{10}$$

$$\hat{\mathbf{w}} = q(\mathbf{w}) = s\bar{\mathbf{w}} \tag{11}$$

$$\boldsymbol{\Delta} = \mathbf{w} - \hat{\mathbf{w}} \tag{12}$$

where $\mathbf{w}$ is the latent weights, s the scale factor and $\hat{\mathbf{w}}$ the dequantized weights.

3.2 Theoretical analysis of QAT

Given a model with quantized parameters and a gradient-based optimizer such as SGD (Kiefer et al., 1952) or Adam (Kingma et al., 2017), we are optimizing the gradient of the loss given in 8 as $\frac{\partial \mathcal{L}^q(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}(q(\mathbf{w}))}{\partial \mathbf{w}}$. We can observe that now the loss function depends on the output of $q(\mathbf{w})$. Hence, the loss will only change if the quantization level of a weight changes.

So for a weight $w_i \in \mathbf{w}$ the gradient can only become zero if w_i or w_j crosses a quantization threshold. Given a deterministic rounding in q, this would imply that $\mathbf{w}$ will cluster around quantization thresholds.

Let's investigate QAT during gradient-based optimization more closely, using a simple linear regression problem with MSE loss and a single quantized weight:

$$\min_w \mathcal{L}^q(w) = \mathbb{E}_{x,y\sim p(X,Y)} \left[\frac{1}{2}(q(w)x - y)^2 \right] \tag{13}$$

We can then take the gradient of $\mathcal{L}^q$ using the quantizer defined in 11 and the definition for QAT 8:

$$\begin{aligned} \frac{\partial \mathcal{L}^q(w)}{\partial w} &= x(q(w)x - y) \cdot \frac{\partial q(w)}{\partial w} \\ &= x(q(w)x + wx - wx - y) \cdot \frac{\partial q(w)}{\partial w} \\ &= (x(wx - y) + x^2(q(w) - w)) \cdot \frac{\partial q(w)}{\partial w} \\ &= \left(\frac{\partial \mathcal{L}(w)}{\partial w} + \lambda \frac{\partial \mathcal{R}(w)}{\partial w} \right) \cdot \frac{\partial q(w)}{\partial w} \end{aligned} \tag{14}$$

An iteration of gradient optimization on $\mathcal{L}^q(w)$ is then given by:

$$w_{t+1} = w_t - \eta \left(\frac{\partial \mathcal{L}(w_t)}{\partial w_t} + \lambda \frac{\partial \mathcal{R}(w_t)}{\partial w_t} \right) \cdot \frac{\partial q(w_t)}{\partial w_t} \tag{15}$$

The first term of the latent weight optimization, $\frac{\partial \mathcal{L}(w)}{\partial w}$ shows that we are optimizing the loss of a regular non-quantized weight, but within a restricted range, defined by the clipping boundaries.

If we look at the second term of the latent weight optimization, $\lambda \frac{\partial \mathcal{R}(w)}{\partial w} = x^2(q(w) - w))$ we can observe two interesting things; the squared input scale the regularization effect and we have that $q(w) - w = -(w - q(w)) = -\mathbf{\Delta}$. The latter means that during optimization, the model is *maximizing* the quantization error $\mathbf{\Delta}$. We maximize the error by moving towards the nearest quantization threshold. We can understand the effect as a pseudo-bin 1, which has some interesting interactions with the latent weight w.

3.2.1 Pseudo bin cases

The pseudo bin interacts with the latent weights in interesting ways. Let us assume we have $x = 1$ then $\lambda = 1$ and the negative gradient is $-(w - y)$, so we are always pointing at the global minima w^*.

If the optimal value w^*, lies outside the quantization bin w is in, then we have that $\frac{\partial \mathcal{L}(w)}{\partial w} > \frac{\partial \mathcal{R}(w)}{\partial w}$. So we see that the pseudo bin increases/decreases (depending on which side of the quantization threshold w is) the convergence speed of w. A special case of this would be that w^* lies outside the maximum quantization level. In this case, w would gravitate towards the edges of the current quantization bin, since it would consistently have a larger gradient than the regularization term, but there is no further bin to transition into.

When w is in the same bin as w^*, we then have $\frac{\partial \mathcal{L}(w)}{\partial w} \leq \frac{\partial \mathcal{R}(w)}{\partial w}$ (The case of equality happens when w^* is precisely on a quantization level and w is at the threshold).

For the case where w^* is not on a quantization level, we see the main effect of the pseudo-bin. This effect depends on the rounding method. Let us assume the quantizer uses a round-to-nearest; the pseudo bin will cause w to oscillate around the quantization threshold. In the most extreme case where w^* lies precisely at the threshold, w will oscillate with equal frequency into the upper and lower quantization bin. This means that the expected value $\mathbb{E}[q(w)] = w^*$ when w oscillates. The further towards a quantization level w^* is the more iterations the weight will spend in that quantization bin. This gives the network a way to, on average, represent the optimal weight value, even if it does not align precisely with a quantization level.

If the quantizer instead uses stochastic rounding (As defined in (Croci et al., 2022)), the behavior changes. Instead of w oscillating, w will now switch between being attracted to the pseudo bin above and below the current quantization level of w, with frequency relative to the closeness of w to its current quantization level. So we would expect w to converge to w^* during training, as opposed to round-to-nearest, where the expected value of $q(w)$ will converge.

3.2.2 L_0 regularization

Let us consider optimizing a vector of latent weights $\mathbf{w}$ and only the MSE gradient. If we assume that $\mathbf{w}^*$ follows a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, if we were to quantize the latent weights $\mathbf{w}$ after converging to $\mathbf{w}^*$, we would get a model with a low L_0 norm, since most of the density will be at the center of the distribution and will therefor be quantized to 0 (Though given we are trying to fit a Gaussian into a finite range, there might be larger spikes at the extremes).

The effect of the pseudo bin on the Gaussian distribution will be that weights within the range of the pseudo bin will get spread out within that range. The width of this pseudo bin is dictated by the step size, implying that fewer bins result in a broader pseudo bin. Consequently, with a lower bit depth in the quantizer, density increasingly shifts toward the thresholds, effectively spreading the latent weight distribution around the threshold and flattening out the Gaussian distribution.

The latent weights clustering at the threshold is composed of two types of weights: those oscillating around the threshold and those that cross over the threshold but have an optimal value close to the quantization level (Causing the gradient toward w^* to cancel out with the pull from the pseudo-bin, which flips as we cross the quantization threshold).

So, optimizing both terms does not necessarily mean that the density is different inside the quantization bins, as opposed to if we were only optimizing the only the MSE gradient— it is simply more spread out within the pseudo bin. However, we would still expect a low L_0 when quantizing a model trained on both terms.

3.2.3 Sources of regularization

So, we learn two main things from the analysis of the linear quantized model:

We are mainly optimizing the same loss as a non quantized model, except we prevent the regular loss from converging completely, and so we would expect a high feature similarity between a quantized and non-quantized model.

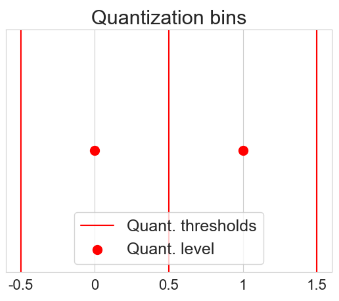
There are several sources of regularization which interact:

The STE combined with the rounding method fundamentally changes how the regularization term affects w.

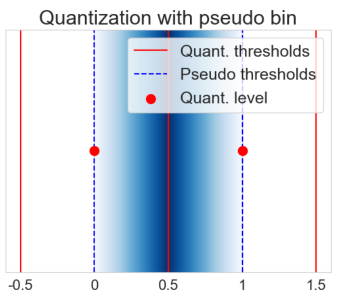
The step size and number of bins (which depends on the bit count of the quantizer).

The discreteness of the quantization operation when applied to the latent weights.

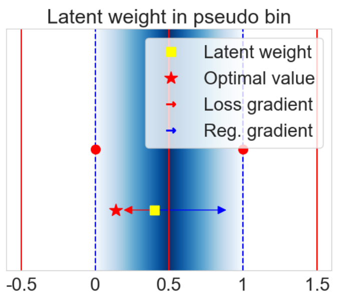
Additionally there is likely also some nuances to how the integer range is set.



(a)



(b)



(c)

Figure 1: The gradient of the regularization term in 15 creates an effect we can term pseudo bin. 1a shows two ordinary quantization bins. 1b, shows the same two bins, but now also one of the pseudo bins induced by the regularization term. The pseudo bin has its center on a quantization threshold and ranges from the quantization level of the bin below to the level of the bin above (The range of the pseudo bin is equal to the range of $\boldsymbol{\Delta}$ which is $[-\frac{s}{2}, \frac{s}{2}]$). All weights in a pseudo bin are pulled toward the center of the pseudo bin (a quantization threshold). The pull scales both by closeness to the quantization threshold (the closer, the stronger the pull. More blue means a stronger pull) and by the squared input. 1c shows the pull on the latent weight inside a pseudo bin.

4 Experiments

4.1 QAT of toy model

4.1.1 Motivation

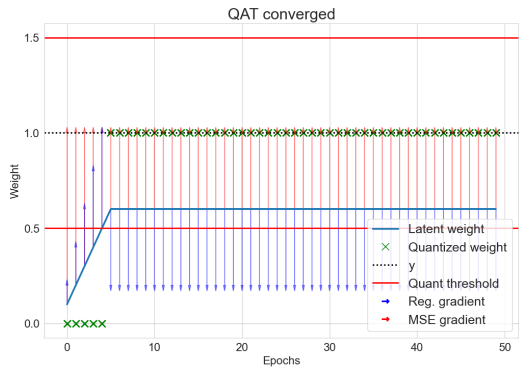
The theoretical regression model 15 implies an interesting set of behaviors affecting the optimization process during QAT. The gradient of the regularization term pulls towards quantization thresholds while the gradient for the MSE term pulls towards w^*. So, while the gradient of the MSE term is greater than the gradient of the regularization term, the model will converge typically (speed up/down based on which side of the threshold the latent weight is). However, as the gradient of the MSE term shrinks (the latent weight approaches w^*), the gradient of the regularization term will dominate, causing oscillations. To confirm these cases, we implement 15 and train with various target values.

4.1.2 Experimental setup

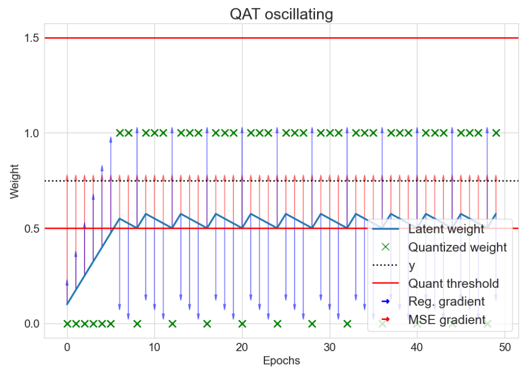
An implementation of the two terms in 15 is made, and then the model is optimized; this allows us to observe the individual effect of each term upon convergence. We let $x = 1$ and initialize $w = 0.1$ with a learning rate of 0.1 and train 50 iterations. Then, for the different experiments, we vary w^* to observe the effect of the regularization term. The quantizer in 11 is used with $s = 1$. Given that the model only has a single weight, 11 simplifies to a *round-to-nearest* operation.

4.1.3 Results

The results in 2 confirm our expectations. We can observe that the regularization term (blue arrow) is pulling the weight towards the quantization threshold. This effect becomes more pronounced the closer the weight is to a threshold. Additionally, we see that the weight spends a proportional amount of iteration in the bin where it is the furthest. We might have expected that the weight could have moved closer to the center when $w^* = 1$, thereby reducing the intensity of the regularization term, but as soon as we enter the bin for 1, the regularization term flips its sign and cancels out the movement of the MSE term.



(a) $w^* = 1$



(b) $w^* = 0.75$

Figure 2: We observe the effect of changing the value of w^*. In 2a, we observe how the latent weight w stays in the quantization bin for 1. In 2b, we see how w oscillates around the threshold.

4.2 QAT of DNN

4.2.1 Motivation

Now that we understand better how QAT imposes its implicit regularization on a linear model, we investigate a larger non-linear model to see if the same behavior is present here. We have three main motivations for this experiment:

Regularizing effects: Given that QAT imposes several forms of regularization, most importantly preventing the non-quantized loss from converging completely, we would expect a higher training loss and a lower generalization gap. Additionally, based on our hypothesis from 4.2, the implicit regularization effects should result in sparse models, scaling inversely with a number of bits in the quantizer.

Source of regularization: To confirm that the source of the regularization is because of the implicit regularization from QAT, we would expect a specific behavior of the latent weights. The regularization term should cause the distribution to flatten out by moving density towards the thresholds, causing spikes at these.

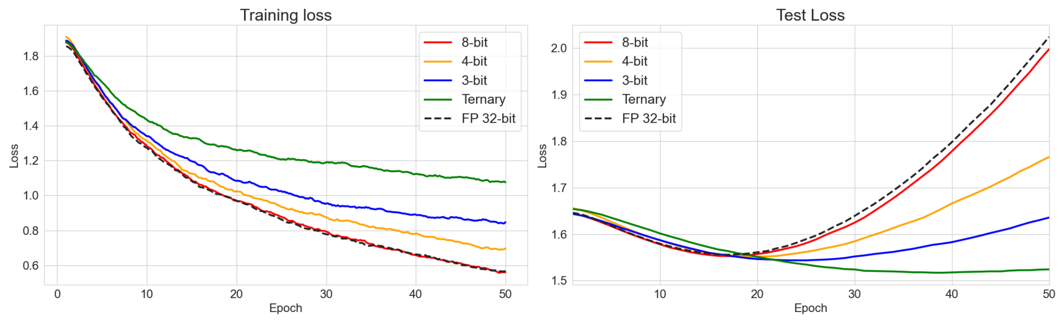
Approximation of non-quantized features: Given that the network is mainly optimizing the loss of a non-quantized model, we would expect that the features learned should be close to those of a non-quantized model.

4.2.2 Experimental setup

A DNN with 5 fully connected layers with 64 width and ReLu activation functions is implemented in PyTorch, trained from scratch on the complete CIFAR10 training dataset (60.000 samples), and evaluated on the test dataset (10.000 samples). An Adam optimizer without weight decay and a learning rate set to 0.001 is used. The quantizer used is defined in 11 and implemented in Brevitas (Pappalardo, 2023), with both round-to-nearest and stochastic rounding. The results from the stochastic rounding model are only used to compare the distribution of weights after training with the ones from the round-to-nearest quantizer. For the experiments, we examine quantization at the following bit widths: Ternary, 3-bit, and 4-bit. The results are compared to an 8-bit QAT model and a baseline DNN with the same architecture and optimizer as the quantized but trained without quantization. For comparing feature similarity between models, the CKA method proposed in (Kornblith et al., 2019) is used.

4.2.3 Results: Regularizing effects

The results support the hypothesis of the regularizing effects of QAT. In 3, we notice that the lower the bit count, the higher the training loss. However, the test loss reaches its lowest point with the ternary quantization. As we also see in 4, the generalization gap is lower, scaling inversely with the bit, showing how the low-bit quantized models exhibit better generalization properties. Finally, we observe the norms of the models during training in 5. The L_0 norm is clearly being regularized, while we also see a less intense effect on the L_1 norm.



(a) Training loss

(b) Test loss

Figure 3: Training and test loss during training of floating point model and the quantized models. At each epoch the models is evaluated on the test set.

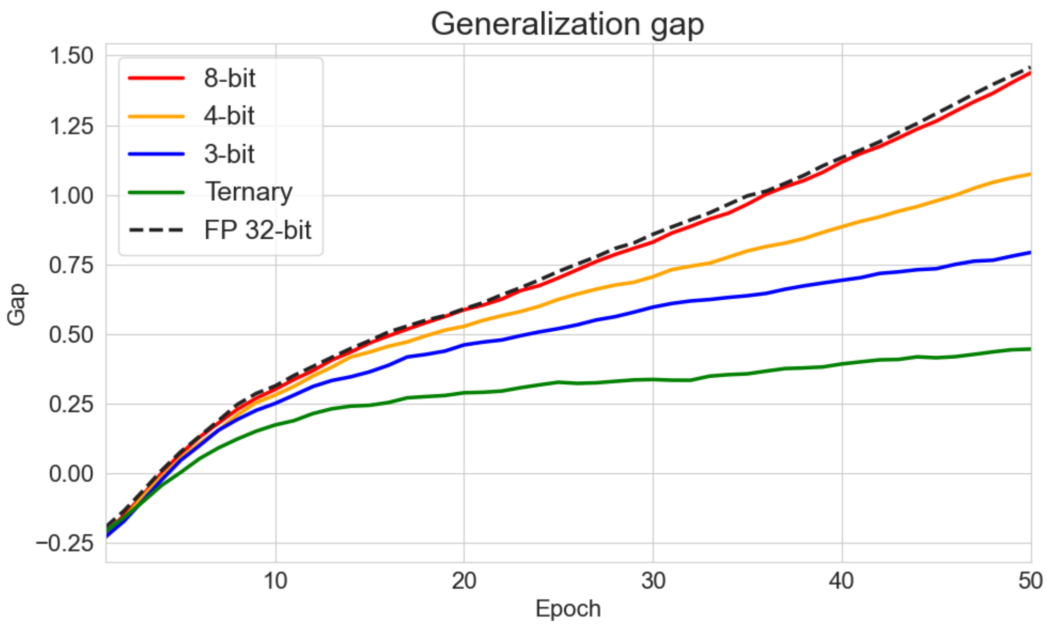
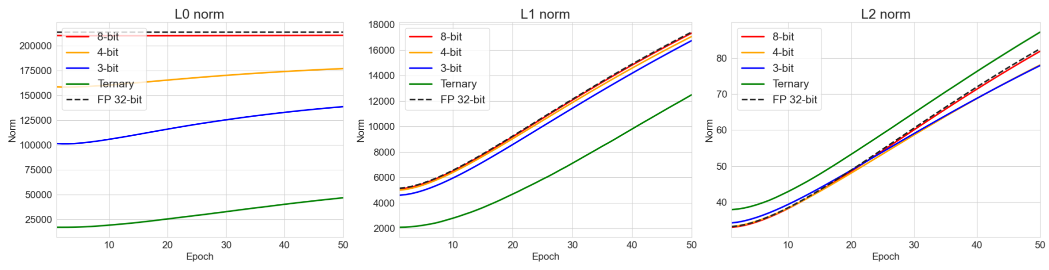


Figure 4: Generalization gap during training. Y-axis shows loss.



(a) L_0 norm (b) L_1 norm (c) L_2 norm

Figure 5: Different norms of the model weights during training. We look at the norm for the effective weights, i.e., for the QAT model, we measure the norm of $q(\mathbf{w})$, and for the FP model, we measure the norm of the floating point weights. The L_0 norm for the FP baseline is defined as number of weights $\geq 1e-5$.

4.2.4 Results: Weight distribution

When examining the latent weights after training, we observe that the effects of the pseudo bin present in the linear model also seem to be present in the DNN. In 6, we observe this expected behavior: The weights are clustering around thresholds and the extremes as we hypothesized. Compared to 8, the optimal floating point weights take a somewhat Gaussian distribution. The quantization then increasingly flattens out this distribution as the density is moved to fewer regularization bins. At 8-bit 8a we seem very close to the distribution of the non-quantized model 8b, though it is interesting to see that another effect of QAT (likely related to the symmetry of the quantizer) is that the latent weight distributions are more centered around 0. In contrast, the non-quantized model in 8b has more outliers lying towards the negative side. Observing the distribution of the latent weights for the low bit quantization, we also see a more significant spike at the negative extreme.

Lastly looking at 7 we can observe that using stochastic rounding instead of round-to-nearest removes the spikes at the thresholds as we expected, and it seems that the weights converge to the optimal values induced by the quantized loss function, rather than oscillating between thresholds.

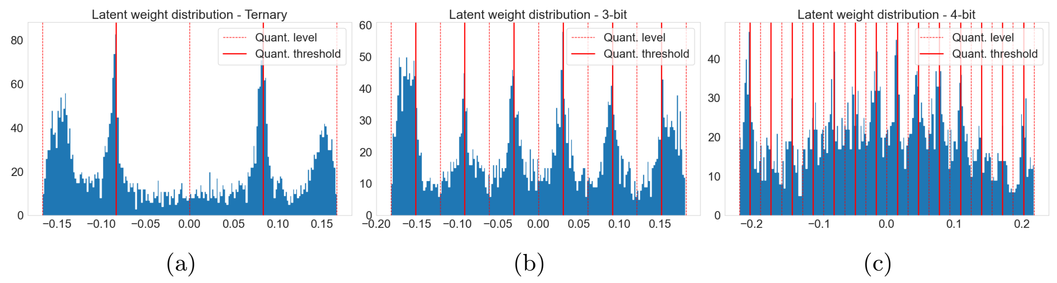


Figure 6: Distribution of latent weights after training for the second layer. Quantizer uses round-to-nearest. The quantization level is the value of the latent weight after $q(w)$, and the threshold is the value where the quantization level changes.

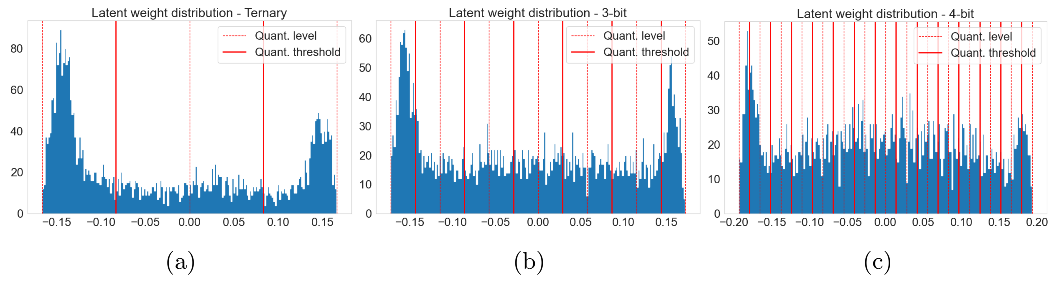
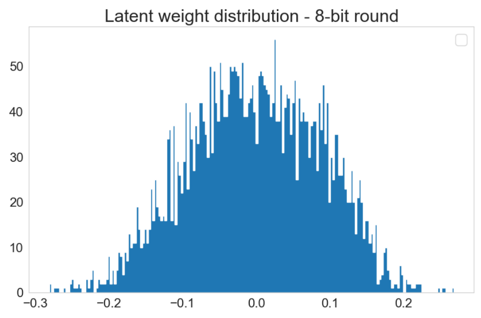
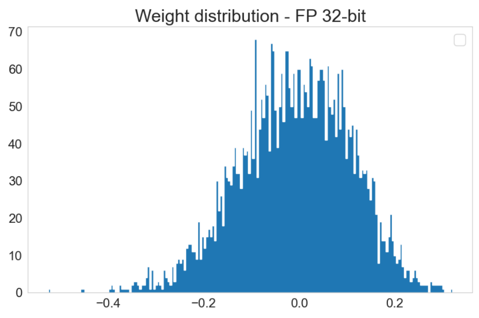


Figure 7: Distribution of latent weights after training for the second layer. Quantizer uses stochastic rounding



(a)



(b)

Figure 8: Distribution of the second layer's weights after training for an 8-bit model with stochastic rounding and an FP model. For clarity, the quantization thresholds and levels are omitted for the 8-bit plot

4.2.5 Results: Feature similarity

Comparing the layer-wise activations between different models in 9, we see that they learn somewhat the same representations. Especially for the first layer, we see a high similarity between the models. Most importantly, there does not seem to be any more difference between a quantized model and an FP model, as opposed to two different FP models with different seeds. Interestingly, it is not only the quantized weights that approximate the same features - if we look at the latent weights, we see that they approximate the FP model - except for the most extreme case of ternary quantization, where the latent weights diverge significantly compared to the FP model. Overall, it seems to indicate that the quantized models and the FP model are mostly implicitly optimizing the same objective.

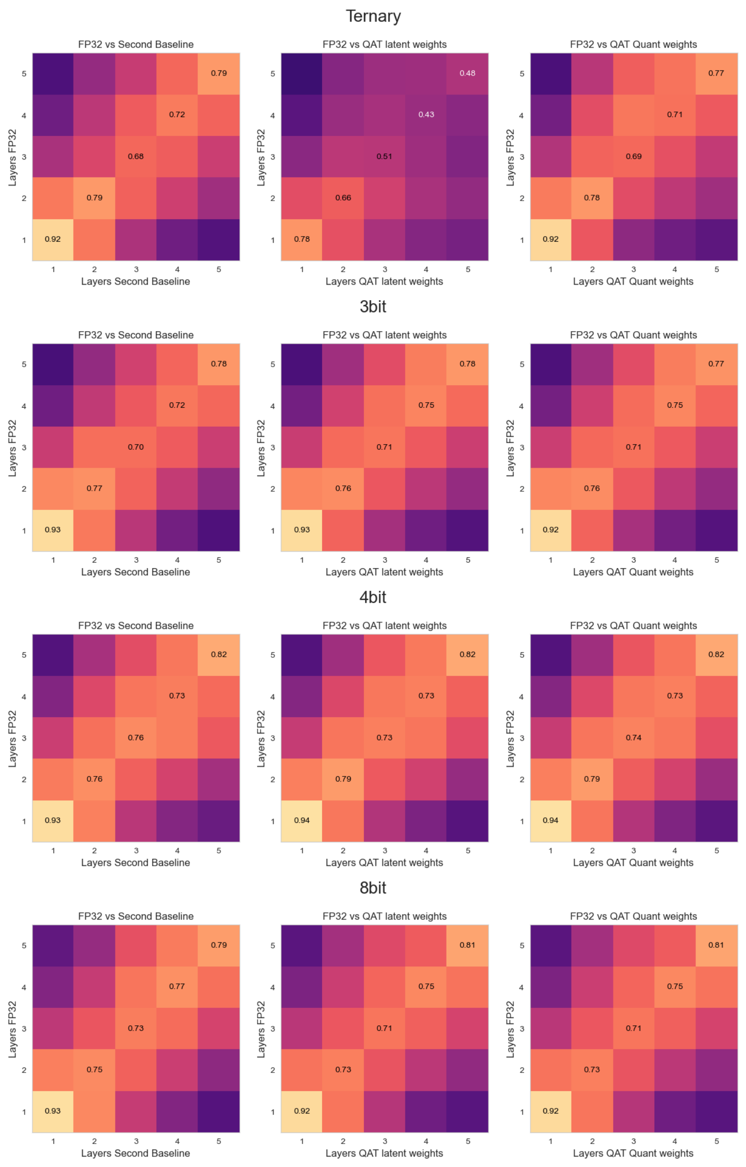


Figure 9: Diagonal shows the layerwise CKA similarity $\in [0, 1]$ (where 1/yellow is most similar) for two distinct networks. The rows correspond to bit level, while the first column is the non-quantized baseline compared with another baseline, the second column is the baseline compared with the activations of the latent weights and the third column is the baseline compared to the quantized weights.

5 Discussions

While we would expect a certain amount of spikes at the extremes (weights that have a pull stronger than the pseudo bin but no other quantization bin to transition to), it is not clear why the spikes are so larger at the extremes.

For the stochastic rounding we see larger spikes at the extremes, which probably indicates that those weights which were before oscillating or stuck at the threshold, has now converged properly as latent weights.

There could also be some relation to the fact that we put important weights on equal basis, and as such, when we do low bit quantization, many weights with medium impact on loss might be put into the highest value bin, causing the spikes at the ends, which disappear the higher the bit (At 8-bit we seemed to have recovered some-what the FP weights).

While we observe radical changes in the regularization effect based on bits (for ternary, we get an effect more akin to dropconnect), we still approximate the same features, which might imply that only fine tuning with QAT is enough.

5.1 Limitations

There are several limitations in rights to the DNN in 4.2; the network is simple and not an optimal architecture for image data. A better solution would be an architecture that included convolution layers. Additionally, it should be verified using more complex data.

The theoretical analysis 15 is for a simple linear model and should be expanded to a 2-layer network with some activation function, perhaps providing further insight. Additionally, it does not consider the initialization of the weight in relation to its interaction with the pseudo bin.

The quantizer 11 uses a range-based scale factor, while the theoretical analysis assumes clipping. This could change the dynamics since, having a range-based scale factor, we effectively let the network control the clipping boundaries and step size. It should be repeated with a more relevant baseline quantizer, such as LSQ (Esser et al., 2020).

Additionally, it is not clear what the effect of choosing a symmetric number of quantization levels around 0 (With $z = 0$, this is what is known as a mid-thread quantizer), as opposed to having one extra on one of the sides, is on the overall effects imposed by the pseudo-bin.

5.2 Future research

Other than the addressing the limitations:

Dithering, oscillations, and generalization To better understand the regularizing effects of QAT, it would be interesting to test with different oscillation dampening methods. Many of these methods are formulated as additive, which draws interesting parallels to dithering (As also noted in (Krishnamoorthi, 2018) sec.2.3, stochastic rounding is a special case of uniform additive noise before rounding. This corresponds to the idea of dithering in signal processing, which is used to de-correlate the quantization error and the signal) and Sharpness Aware Minimization.

Sparsity given the sparse models, there could be opportunities to train models which are both quantized to a given bit width and prunning ready to a fixed pruning rate by combining our insights with the proposed method in (Sahbi, 2024).

Topology of ternary networks The ternary networks show many interesting properties. Primarily, it delivers extremely sparse representations, which perhaps could be akin to binary lottery tickets. It would be interesting to experiment on these ternary network topologies and see what occurs if they are trained at different bit widths and initializations.

Latent weights and biased estimator While the latent weights is a biased estimator, the quantized weights are what is used for inference, not making it quite clear what the relation is between latent weights being a biased estimator and the quantized weights applied on the forward pass as also noted in (Nagel, Fournarakis, Bondarenko, et al., 2022).

6 Conclusions

In this work, we investigated the effects of QAT on the generalization abilities of DNNs. We theoretically analyzed QAT for a linear model, showing that the loss during a gradient-based optimization can be decomposed into optimizing a non-quantized model and a regularization term that maximizes the quantization error. We show on DNNs that QAT indeed provides an implicit form of regularization - which arises from a complex interaction between several parts of QAT, such as the rounding method and the STE. So, it does seem that while quantization reduces the bit precision of the parameters if implemented through QAT, it additionally provides a regularizing effect, which likely explains why we paradoxically sometimes observe accuracy improvements after model compression. Though it must be noted that it is not a free lunch, QAT significantly increases the cost and complexity of the training process.

7 Appendix

A Weight distribution for the second layer for FP32 model and FP32 with additive noise after training

A.1 Motivation

To confirm that i.i.d. additive noise does not result in the same behavior of the latent weights as induced by QAT.

A.2 Experimental setup

We train two FP baselines with the same setup as in 4.2, only one with additive uniform noise with 4-bit stepsize (The uniform noise lies in the range $[-\frac{s}{2}, \frac{s}{2}]$ where s is calculated by the formula 7 with $b = 4$) during training and one without noise. For 8-bit the noise becomes negligible and for lower bits the weights diverges if the noise is not scaled down considerably. The noise is scaled by 0.1.

A.3 Results

We see that the effect of additive uniform noise differs from what we would expect during QAT. THe main difference between the non-noised FP is that the weights has a smaller magnitude.

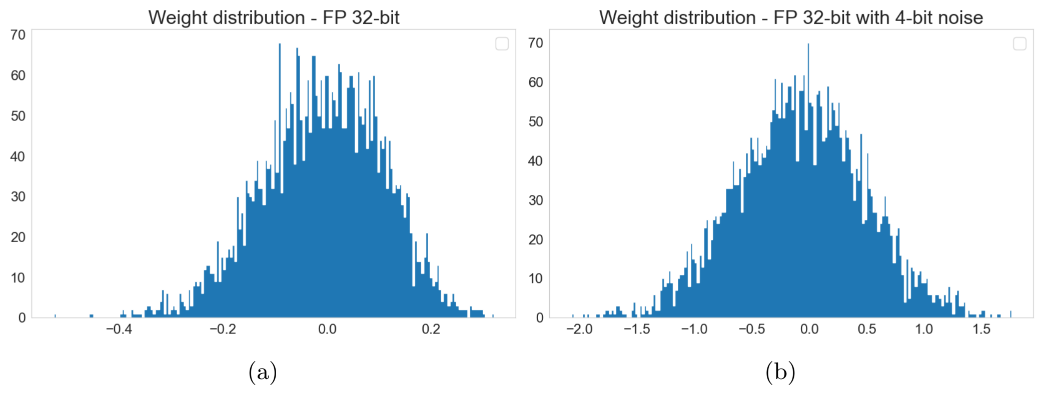


Figure 10: Distribution of the second layer's weights after training for a FP 32-bit model and a FP 32-bit with noise added corresponding to 4-bit uniform noise scaled by 0.1.

B Weight distribution for the first layer after training

B.1 Motivation

The second layer has fewer parameters than the first layer, so to see the effect on a larger layer, we also investigated the latent weights of the first layer.

B.2 Experimental setup

Same setup as in 4.2.

B.3 Results

The effects of the pseudo-bin is still present, we see spikes around the thresholds and the extremes, albeit at a much lower intensity than second layer. Though we still observe that the overall distribution gets flattened out (not too easy to see since the y-scale is different), compared to the FP model which has a very sharp spike around 0.

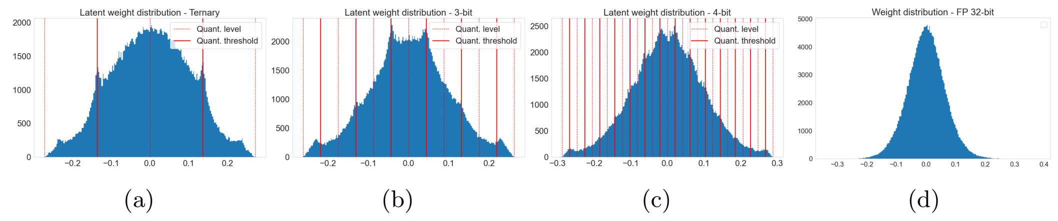


Figure 11: Distribution of the latent weights for the first layer after training and the weights for a FP 32-bit model. For the quantized models a round-to-nearest is used.

C Weight initialization

C.1 Motivation

For a ternary quantization with the 0 bin centered at 0, and a uniform initialization of model parameters, most of the weights will be quantized to 0, which could be the cause of the low L_0 norm for ternary weights.

C.2 Experimental setup

Using the same setup as in 4.2, though only with three fully connected layers. Additionally the weights is initialized with an "inverse gaussian", which is basically a gaussian distribution where the mass has been moved from the center to the extremes. This causes the quantized weights to mostly fall in the -1/1 bins at the start of training.

C.3 Results

We see that even if we move the density towards the outer bins at initialization, the model still converges to about the same L_0 norm as with the uniform initialization.

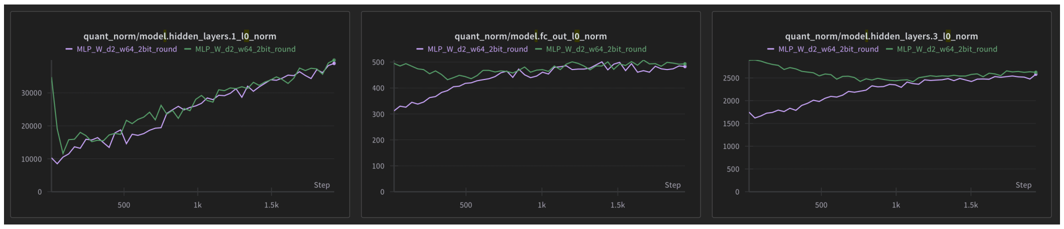


Figure 12: Here we have the L_0 norm for the different layers. The first cell shows the input layer, second cell the output layer and third cell the second layer. The green line is with inverse gaussian initialization and the purple line is for uniform initialization.

References

[1] Milad Alizadeh et al. *Gradient ℓ_1 Regularization for Quantization Robustness.* 2020. arXiv: `2002.07520 [cs.LG]` (cit. on p. 8).

[2] MohammadHossein AskariHemmat, Reyhane Askari Hemmat, et al. *QReg: On Regularization Effects of Quantization.* 2022. arXiv: `2206.12372 [cs.CV]` (cit. on p. 8).

[3] MohammadHossein AskariHemmat, Ahmadreza Jeddi, et al. *QGen: On the Ability to Generalize in Quantization Aware Training.* 2024. arXiv: `2404.11769 [cs.LG]` (cit. on p. 8).

[4] Yoshua Bengio et al. *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation.* 2013. arXiv: `1308.3432 [cs.LG]` (cit. on p. 7).

[5] Wentao Chen et al. *Quantization of Deep Neural Networks for Accurate Edge Computing.* 2021. arXiv: `2104.12046 [cs.CV]` (cit. on p. 7).

[6] Matthieu Courbariaux, Yoshua Bengio, et al. *BinaryConnect: Training Deep Neural Networks with binary weights during propagations.* 2016. arXiv: `1511.00363 [cs.LG]` (cit. on p. 7).

[7] Matthieu Courbariaux, Itay Hubara, et al. *Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1.* 2016. arXiv: `1602.02830 [cs.LG]` (cit. on p. 7).

[8] Matteo Croci et al. "Stochastic rounding: implementation, error analysis and applications". In: *R. Soc. Open Sci.* 9 (2022), p. 211631. DOI: `10.1098/rsos.211631` (cit. on p. 10).

[9] Alexandre Défossez et al. *Differentiable Model Compression via Pseudo Quantization Noise.* 2022. arXiv: `2104.09987 [stat.ML]` (cit. on p. 8).

[10] Steven K. Esser et al. *Learned Step Size Quantization.* 2020. arXiv: `1902.08153 [cs.LG]` (cit. on pp. 6, 19).

[11] Kartik Gupta et al. *Reducing the Side-Effects of Oscillations in Training of Quantized YOLO Networks.* 2023. arXiv: `2311.05109 [cs.CV]` (cit. on p. 8).

[12] Benoit Jacob et al. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference.* 2017. arXiv: `1712.05877 [cs.LG]` (cit. on p. 7).

[13] Jack Kiefer et al. "Stochastic Estimation of the Maximum of a Regression Function". In: *Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466. DOI: `10.1214/aoms/1177729392` (cit. on p. 9).

[14] Diederik P. Kingma et al. *Adam: A Method for Stochastic Optimization.* 2017. arXiv: `1412.6980 [cs.LG]` (cit. on p. 9).

[15] Simon Kornblith et al. *Similarity of Neural Network Representations Revisited.* 2019. arXiv: `1905.00414 [cs.LG]` (cit. on p. 14).

[16] Raghuraman Krishnamoorthi. *Quantizing deep convolutional networks for efficient inference: A whitepaper.* 2018. arXiv: `1806.08342 [cs.LG]` (cit. on pp. 6, 7, 19).

[17] D. Marco et al. "The validity of the additive noise model for uniform scalar quantizers". In: *IEEE Transactions on Information Theory* 51.5 (2005), pp. 1739–1755. DOI: `10.1109/TIT.2005.846397` (cit. on p. 8).

[18] Yuriy Mishchenko et al. "Low-Bit Quantization and Quantization-Aware Training for Small-Footprint Keyword Spotting". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 2019, pp. 706–711. DOI: `10.1109/ICMLA.2019.00127` (cit. on p. 7).

[19] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, et al. *A White Paper on Neural Network Quantization*. 2021. arXiv: `2106.08295 [cs.LG]` (cit. on pp. 5, 6).

[20] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, et al. *Overcoming Oscillations in Quantization-Aware Training*. 2022. arXiv: `2203.11086 [cs.LG]` (cit. on p. 20).

[21] Alessandro Pappalardo. *Xilinx/brevitas*. 2023. DOI: `10.5281/zenodo.3333552`. URL: `https://doi.org/10.5281/zenodo.3333552` (cit. on p. 14).

[22] Hichem Sahbi. "DAMP: Distribution-Aware Magnitude Pruning for Budget-Sensitive Graph Convolutional Networks". In: Apr. 2024, pp. 3070–3074. DOI: `10.1109/ICASSP48485.2024.10448148` (cit. on p. 20).

[23] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html` (cit. on p. 7).

[24] Li Wan et al. "Regularization of Neural Networks using DropConnect". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta et al. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1058–1066. URL: `https://proceedings.mlr.press/v28/wan13.html` (cit. on p. 7).

[25] B. Widrow. "A Study of Rough Amplitude Quantization by Means of Nyquist Sampling Theory". In: *IRE Transactions on Circuit Theory* 3.4 (1956), pp. 266–276. DOI: `10.1109/TCT.1956.1086334` (cit. on p. 8).

[26] Chen Xu et al. *Alternating Multi-bit Quantization for Recurrent Neural Networks*. 2018. arXiv: `1802.00150 [cs.LG]` (cit. on p. 7).

[27] Penghang Yin et al. *Quantization and Training of Low Bit-Width Convolutional Neural Networks for Object Detection*. 2020. arXiv: `1612.06052 [cs.CV]` (cit. on p. 7).

[28] Kaiqi Zhang et al. *Why Quantization Improves Generalization: NTK of Binary Weight Neural Networks*. 2022. arXiv: `2206.05916 [cs.LG]` (cit. on p. 8).