Optimal Clipping and Magnitude-aware Differentiation for Improved Quantization-aware Training

Charbel Sakr [1] **Steve Dai** [1] **Rangharajan Venkatesan** [1] **Brian Zimmer** [1] **William J. Dally** [1] **Brucek Khailany** [1]

Abstract

Data clipping is crucial in reducing noise in quantization operations and improving the achievable accuracy of quantization-aware training (QAT). Current practices rely on heuristics to set clipping threshold scalars and cannot be shown to be optimal. We propose Optimally Clipped Tensors And Vectors (OCTAV), a recursive algorithm to determine MSE-optimal clipping scalars. Derived from the fast Newton-Raphson method, OCTAV finds optimal clipping scalars on the fly, for every tensor, at every iteration of the QAT routine. Thus, the QAT algorithm is formulated with provably minimum quantization noise at each step. In addition, we reveal limitations in common gradient estimation techniques in QAT and propose magnitude-aware differentiation as a remedy to further improve accuracy. Experimentally, OCTAV-enabled QAT achieves state-of-the-art accuracy on multiple tasks. These include training-from-scratch and retraining ResNets and MobileNets on ImageNet, and Squad fine-tuning using BERT models, where OCTAV-enabled QAT consistently preserves accuracy at low precision (4-to-6-bits). Our results require no modifications to the baseline training recipe, except for the insertion of quantization operations where appropriate.

1 Introduction

Deep neural networks (DNNs) are powerful models achieving state-of-the-art accuracy on various cognitive tasks such as image classification, object detection, and natural language processing (LeCun et al., 2015). However, DNN successes have been achieved at the expense of a high computational and parameter complexity. Indeed, networks commonly require around 4 billion multiply-accumulates (MACs) (He et al., 2016) or have over 100 million parameters (Taigman et al., 2014). With further progress in deep learning, the growth of DNN resource requirements shows no sign of slowing down (Bianco et al., 2018). Fortunately, reduced-precision implementation has been shown to largely lower the computational complexity of deep learning models (Gupta et al., 2015).

Reduced-precision deep learning is widely adopted and multi-faceted. One option is to perform post-training quantization (PTQ), which benefits inference only and consists of taking a pretrained network and implementing it in reduced-precision, with no retraining allowed. While conceptually simple, PTQ is challenging, as determining an accuracy-preserving quantization strategy is non-trivial. In recent works, proposed practical PTQ solutions include speculative hybrid high/low precision number formats (Jain et al., 2019; Lin et al., 2017) and Batch-Norm-guided data free quantization (Nagel et al., 2019).

The results presented in our work apply to any quantization setup, including PTQ. However, the main feature is the ability to optimize quantization metadata on the fly. This is most arguably effective for another facet: quantization-aware training (QAT), where weights and activations are quantized during training. A superset of this problem is fully quantized training (FQT) (Sakr & Shanbhag, 2019), where gradients and weight updates are also quantized. While we focus on QAT here, an interesting and important extension of our work is to apply our results to FQT.

1.1 Quantization-aware training and related works

There are three main use cases for QAT: **training-from-scratch** where the starting point is a randomly initialized network; **retraining** where a pretrained model is quantized and retrained for a short time on the same dataset; and **fine-tuning** where the starting point is a model pretrained on one dataset and trained on another.

Early works on QAT showed that binary-weighted (Courbariaux et al., 2015) and fully-binarized (Hubara et al., 2016) networks can be accurately trained on simple models and datasets. To improve accuracy in more difficult tasks,

[1]The authors are with NVIDIA Corporation, Santa Clara, CA 95051 USA. Correspondence to: Charbel Sakr <csakr@nvidia.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

DoReFa-Net (Zhou et al., 2016) increased the forward precision to 4-bit and used *max-scaling*, i.e., matching the largest quantized representation to the largest value in the set of elements (tensor or vector) to be quantized.

An advantage of max-scaling is that it is well-defined. Indeed, max-scaled QAT can be implemented using the same training recipe as a full precision baseline, simply by inserting the quantization operations where appropriate. Thus, there are no hyperparameters required, and results can be readily reproduced. Unfortunately, max-scaling incurs large amounts of quantization noise, harming accuracy. Nevertheless, quantization fidelity can be improved with clipping, which has been researched by several recent works.

Making the *clipping scalar* a learned parameter, as was done in PACT (Choi et al., 2018), has been shown to significantly improve accuracy. However, by virtue of the added learnable parameters, a QAT-dedicated training recipe is required. As such, PACT is highly sensitive to hyperparameter tuning and is therefore difficult to reproduce and cannot easily generalize. An advantage of PACT is that, much like max-scaling, it computes quantization metadata on the fly. We term such a scheme **dynamic quantization**, which is useful for setups with time-varying tensor statistics, such as training-from-scratch and fine-tuning.

Alternatively, **static quantization** was explored by (Wu et al., 2020) who used calibration on pretrained data in order to fix the clipping scalars at the start of QAT. Such an approach can only be applied when tensor statistics minimally change over time, e.g., in a short retraining or fine-tuning setup. Further, the choice of calibration strategy is closely tied to the performance of QAT, necessitating network-specific exploration. Percentile calibration was shown to be robust (Wu et al., 2020), and state-of-the-art retraining accuracy was obtained through extensive calibration exploration (Abdolrashidi et al., 2021). An advantage of static quantization is its conceptual simplicity; once a good calibration strategy is identified, a QAT routine can be readily and reproducibly implemented.

The above works all use uniform quantization, also called integer quantization (Wu et al., 2020) and fixed-point quantization (Sakr & Shanbhag, 2019). In this work, we also focus on uniform quantization, which is well-suited for efficient hardware implementations, particularly at low (less-than-8-bit) precision (Han et al., 2016).

Still, we do note that recent works have shown promising results for QAT under non-uniform quantization: structured quantization, such as that derived from low precision floating-point (Wang et al., 2018; Sun et al., 2019) and logarithmic (Lee et al., 2017; Zhao et al., 2021) number systems, as well as custom formats, such as Flexpoint (Köster et al., 2017), AdaptivFloat (Tambe et al., 2019), and LQNets (Zhang et al., 2018). We derive all our theoretical and experimental results for uniform quantization; however, our work can be extended to non-uniform quantization.

Finally, in our work, we use the exact same training recipe as the full-precision baseline for training-from-scratch and fine-tuning. For retraining, we use a shortened version of the training-from-scratch recipe. Recent works have attempted to improve accuracy using QAT-specific training techniques, such as distillation (Choi et al., 2020). Such works are orthogonal to ours; our methods can be inserted on top of any training routine.

1.2 Contributions

None of the prior arts provides guarantees on the optimality of the chosen clipping scalars. Most works use a strategy believed to be adequate, e.g., relying on the training algorithm (Choi et al., 2018), or yielding small quantization noise on calibrated data (Wu et al., 2020). In contrast, our work makes the following contributions:

- We derive OCTAV: a fast recursive algorithm based on the Newton-Raphson method to determine MSE-minimizing clipping scalars. With OCTAV, optimal quantization metadata can be computed for every tensor, at every iteration of the QAT routine. Thus, the QAT algorithm is formulated with minimum quantization noise at each iteration.
- We analyze common candidates for quantized gradient estimation, for which we reveal risks of gradient explosion and partial premature stoppage of convergence. We avoid these risks by proposing magnitude-aware differentiation, which leads to a noticeable improvement in QAT accuracy.
- We show that OCTAV-enabled training-from-scratch QAT achieves state-of-art accuracy on several ImageNet benchmarks. Indeed, 4-bit training of ResNet-50, ResNet-18, ResNet-101, and MobileNet-V2 models results in less-than-1% accuracy degradation compared to the full precision baseline. We also provide promising results on the much-harder-to-quantize MobileNet-V3-Small and MobileNet-V3-Large. For all results, no modification to the baseline training recipe is made.
- We find OCTAV-enabled QAT to always yield highly accurate solutions in 4-bit retraining. We find static quantization more accurate for large models, such as ResNets; and thus propose static-OCTAV for fast calibration yielding high accuracy. In contrast, small models such as MobileNets require dynamic quantization for retraining, and OCTAV is shown to be far superior to any other strategy.
- Finally, we find OCTAV-enabled QAT to be most appropriate for Squad fine-tuning of BERT models. Even when restricted to static quantization, we find that static-OCTAV consistently outperforms other calibration methods.

2 Clipped Quantization

Consider some data x derived from a distribution $f_X()$. We define B-bit quantization as the process of mapping x to one of 2^B predefined levels $\{r_i\}_{i=1}^{2^B}$. The quantized data is obtained as: $\mathbb{Q}(x) = \arg\min_{\{r_i\}_{i=1}^{2^B}} |x - r_i|$. The choice of $\{r_i\}_{i=1}^{2^B}$ is crucial in setting the fidelity of quantization, which we metricize via the mean squared error (MSE):

$$J = \mathbb{E}\left[(\mathbb{Q}(X) - X)^2\right] \tag{1}$$

For unconstrained quantization, this metric can be minimized using the Lloyd-Max algorithm (Lloyd, 1982).

Among uniformly constrained quantizers, we first introduce the maxed-scaled one. Assume there exists a scalar $s_{\max}$ such that $f_X(x) = 0$ for $|x| > s_{\max}$. In practice, $s_{\max}$ can be the largest available element in absolute value. The max-scaled quantizer assigns the levels $\{r_i\}_{i=1}^{2^B}$ as an arithmetic progression on $[-s_{\max}, s_{\max}]$[1]. Thus, the max-scaled quantization operation is given by:

$$\mathbb{Q}(x) = s_{\max} \cdot 2^{1-B} \cdot \text{round}\left(x \cdot 2^{B-1}/s_{\max}\right) \tag{2}$$

with the rounding operation being applied on integers[2]. This quantizer has been extensively studied in signal processing (Goel & Shanbhag, 1998) and machine learning (Sakr et al., 2017) and its MSE, derived using an additive model of quantization noise, is given by $J = s_{\max}^2 \frac{4^{-B}}{3}$.

Often, max-scaling is data inefficient due to its large quantization range and thus step size. A simple, but powerful method to improve uniform quantization is to allow for data clipping (Sakr & Shanbhag, 2021; Gonugondla et al., 2020). Specifically, a narrower quantization interval $[-s, s]$ is used, with the clipping scalar $s < s_{\max}$, and the quantization operation given by:

$$\begin{aligned}\mathbb{Q}(x) &= \text{clip}\left(s \cdot 2^{1-B} \cdot \text{round}\left(x \cdot 2^{B-1}/s\right), -s, s\right) \\ &= \begin{cases} -s & \text{if } x < -s \\ s \cdot 2^{1-B} \cdot \text{round}\left(x \cdot 2^{B-1}/s\right) & \text{if } x \in [-s, s] \\ s & \text{if } x > s \end{cases}\end{aligned} \tag{3}$$

With clipping, the MSE in (1) depends on s and is given by:

$$J(s) = \frac{4^{-B}}{3} s^2 \int_0^s f_{|X|}(x)dx + \int_s^\infty (s - x)^2 f_{|X|}(x)dx \tag{4}$$

[1]We assume signed data, without loss of generality. In Appendix A, we list all required modifications for our results to apply to the unsigned case, i.e., quantization over $[0, s_{\max}]$.

[2]For notational simplicity and mathematical tractability, boundary effects introduced by number systems (e.g., two's complement) are neglected. They are, however, implemented in our experiments.

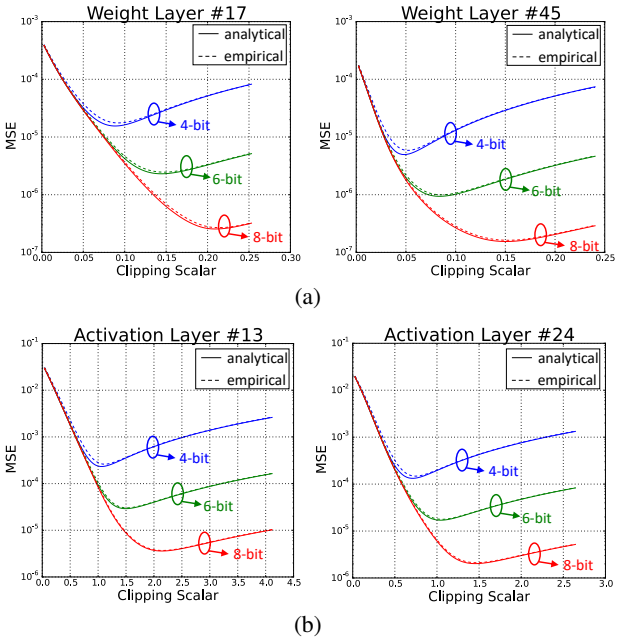


Figure 1: Sweep of the quantization MSE as a function of the clipping scalar s for two arbitrary weight (a) and activation (b) layers in a pretrained ResNet-50 model. Activation data is obtained by sampling a random input batch from the training set. Solid lines are obtained by evaluating the MSE formula in (4) using histograms and numerical integration. Dashed lines are obtained by empirically evaluating (1), i.e., quantizing each tensor element according to (3) and averaging the resulting squared errors.

where $f_{|X|}()$ is the distribution of the absolute value of the data. Equation (4) is obtained by evaluating (1) using the law of total expectation; the aforementioned additive noise model is assumed on the discretization interval $[-s, s]$ and the definition of MSE is used when clipping occurs. For discretization noise, the term $\frac{s^2 4^{-B}}{3}$ does not require *a priori* knowledge of data distribution. It is obtained through sampling theory where quantization noise arises via approximating the neighborhood of a quantization level of *any* distribution as a local rectangle (Widrow & Kollár, 2008).

Figure 1 shows variations in quantization MSE as a function of clipping scalar. We use data from a trained ResNet-50 model (He et al., 2016), with activations corresponding to a random input batch sampled from the ImageNet dataset (Deng et al., 2009). We depict both the evaluation of (4) using histograms and numerical integration, and the empirically measured MSE in (1) via element-wise quantization and squared error averaging. We observe the following:

- Our formula closely matches the empirical MSE. Thus, we use it as a building block for our upcoming analyses.
- There exists an optimal scalar s^* minimizing the MSE.

This optimum balances the trade-off between discretization and clipping noise. When $s < s^*$, excess clipping leads to an increase in J in spite of smaller discretization noise. Conversely, when $s > s^*$, clipping is minimal but the larger quantization step size causes an increase in discretization noise and J.

- The optimal scalar s^* is a function of both data distribution $f_X()$ and number of bits B. The dependence on $f_X()$ is identified by virtue of s^* being different for different layers (e.g., when $B = 4$, s^* is approximately 0.1 and 0.05 for weight layers #17 and #45, respectively). The dependence on B is identified by virtue of s^* varying with precision when data is unchanged (e.g., for activation layer #13, s^* is approximately 1.0 and 2.0 for $B = 4$ and $B = 8$, respectively).

Finding s^* can be done **offline** through brute force search, i.e., sweeping the value of s. However, this task is highly time-consuming and hard to implement dynamically. The analytical evaluation of (4) requires histograms to estimate $f_X()$ and numerical integration. Similarly, an empirical evaluation requires successive rounding and reduction operations on large tensors. In the next section, we present a method to determine the optimal clipping scalar s^* **online**.

3 Optimally Clipped Tensors And Vectors

We present our main theoretical result: a recursive formula to analytically determine s^*.

Theorem 3.1. *Given a data distribution $f_X()$, the clipping scalar s^* minimizing the clipped quantization MSE in* (4) *can be found by assigning a random guess s_1 and recursively computing $\{s_n\}_{n>1}$ until convergence using:*

$$s_{n+1} = \frac{\mathbb{E}\left[|X| \cdot \mathbb{1}_{\{|X|>s_n\}}\right]}{\frac{4^{-B}}{3}\mathbb{E}\left[\mathbb{1}_{\{|X|\leq s_n\}}\right] + \mathbb{E}\left[\mathbb{1}_{\{|X|>s_n\}}\right]} \tag{5}$$

Proof. We provide the complete proof in Appendix B. For the benefit of the interested reader, we here mention the main idea behind the result. It consists of using the Newton-Raphson algorithm, which recursively computes $s_{n+1} = s_n - J'(s_n)/J''(s_n)$. In Appendix B, we show how first and second derivatives of $J(s)$ are derived to obtain (5). □

Theorem 3.1 applies to an arbitrary distribution, and the following corollary applies to tensor and vector quantization.

Corollary 3.2. *The clipping scalar s^* minimizing the clipped quantization MSE in a tensor or vector $\vec{t}$ can be found by assigning a random guess s_1 and recursively computing $\{s_n\}_{n>1}$ until convergence using:*

$$s_{n+1} = \frac{\sum_{x\in\vec{t}}\left[|x| \cdot \mathbb{1}_{\{|x|>s_n\}}\right]}{\frac{4^{-B}}{3}\sum_{x\in\vec{t}}\left[\mathbb{1}_{\{0<|x|\leq s_n\}}\right] + \sum_{x\in\vec{t}}\left[\mathbb{1}_{\{|x|>s_n\}}\right]} \tag{6}$$

Proof. The empirical distribution of the data inside $\vec{t}$ is used in lieu of the abstract $f_X()$ in Theorem 3.1. Thus, expectations in (5) are replaced by average summations. Numerator and denominator are both multiplied by the number of elements in $\vec{t}$, suppressing the need for division. As zeros can be represented using integer quantization, zero elements in $\vec{t}$ are excluded from the distribution (see first term in the denominator). This is done to prevent an over-estimation of the total quantization noise for very sparse tensors. □

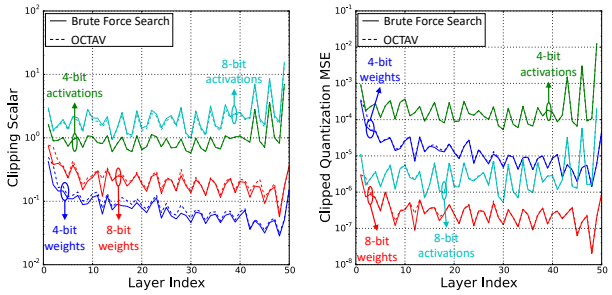
We call the algorithm in Corollary 3.2 Optimally Clipped Tensors And Vectors (OCTAV). Derived from the Newton-Raphson method, OCTAV converges very quickly algorithmically. We only use 10 iterations for any OCTAV implementation in this paper. Additionally, OCTAV is insensitive to the choice of initial guess. Indeed, for weight and activation tensors of a pretrained ResNet-50 network, OCTAV consistently converges to the same solution for various choices of s_1 including 0, $s_{\max}$, $3\sigma_{\vec{t}}$, $4\sigma_{\vec{t}}$, and $5\sigma_{\vec{t}}$, where $\sigma_{\vec{t}}$ denotes the tensor standard deviation. In our upcoming experiments, we use $s_1 = \sum_{x\in\vec{t}}|x| / \sum_{x\in\vec{t}}\mathbb{1}_{|x|>0}$, which is the value of s_3 if the initial guess is set to $s_{\max}$.

Computationally, each iteration of the OCTAV algorithm can be implemented using fast operations. Indeed, the only vector/tensor operations required are the indicator function, which is realizable via simple Boolean datatype casting, and element-wise absolute values, multiplications, and comparisons. Afterwards, sum reductions are performed and only residual scalar operations remain, including one division.

The algorithmic and computational efficiencies of OCTAV make it significantly faster than a conventional brute force search for s^*. On a CPU, and with no code optimizations, OCTAV is $\sim$10$\times$ faster than brute force when applied to weight and activation tensors of a BERT-Base model. Details of this comparison are included in Appendix F.

Importantly, all operations required in (6) are tensor operations. Thus, OCTAV can be implemented on GPUs using any deep learning package. For instance, our implementation only invokes native PyTorch (Paszke et al., 2017) operations. Consequently, we can embed OCTAV into any QAT routine to realize dynamic quantization using optimal clipping scalars for each tensor at each iteration. The added optimization does incur an overhead, but because OCTAV is fast, it is possible to perform the desired QAT in reasonable amounts of time. We also note that all OCTAV operations are broadcastable and can be used when sub-tensor scaling is required (Wu et al., 2020). Thanks to the broadcasts, optimization for finer-grained scaling incurs no slowdown.

The OCTAV algorithm is guaranteed to converge to the global optimum of the convex MSE $J(s)$ in (4). The trade-off between clipping and discretization noise discussed in Section 2 leads to this convexity, which is verified by virtue



(a) (b)

Figure 2: Comparison of (a) optimal clipping scalar as determined by a brute force search (solid lines) and OCTAV (dashed lines), and (b) corresponding empirically measured clipped quantization MSE at every weight and activation layer in a pretrained ResNet-50 model for $B = 4, 8$. The OCTAV-determined clipping scalar is the result of invoking Corollary 3.2 and (6) while the brute force search is realized by sweeping the value and s and empirically evaluating (1).

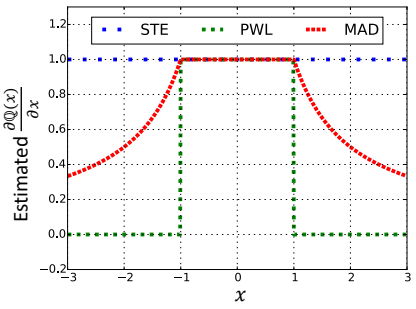


Figure 3: Gradient estimators for the clipped quantization operation including STE, PWL, and our proposed MAD. For this example, we assume the clipping scalar to be $s = 1$.

of the second derivative $J''(s)$ being positive (see Appendix B). In Figure 2(a), we plot the optimal clipping scalar for all weight and activation layers in a ResNet-50 pretrained model, as determined by OCTAV and a brute force search. Consistently, both solutions are either equal or close to one another. Even in the case of a slight mismatch, the resulting quantizers have identical MSE, as shown in Figure 2(b).

Recall that (4) is derived with additive noise assumed in the discretization region. This model is valid provided the quantization step is small (Widrow & Kollár, 2008). On rare occasions, and in the presence of very large outliers, this model can be inaccurate. Indeed, using a large clipping scalar to cater for outliers at the expense of quantizing all small values to zero leads to one local minimum of the empirical MSE in (1) unidentified by (4). In this case, OCTAV still converges to the local minimum balancing discretization and clipping, where the noise model in (4) is valid. This rare phenomenon was observed in some activation layers of BERT models and investigated in Appendix G. Interestingly, the local minimum to which OCTAV converges to is a favorable one for QAT, as shown in Section 5.3.

The formulation above minimizes the quantization MSE by choice. We aspire to train with minimum quantization noise variance, and have shown above how to do so using OCTAV. Minimizing noise is a desirable feature of QAT, and our promising experimental results in Section 5 support this contention. Using the above idea of online optimization using the Newton-Raphson algorithm, it is possible to derive similar methods for minimizing alternative quantization fidelity metrics such as L_p-norm, KL divergence, and others. Such optimizations are beyond the scope of this paper, but can form the basis of interesting extensions of our work.

4 Improving QAT Gradient Estimation

The OCTAV algorithm enables QAT with minimal noise at each iteration, thereby boosting accuracy. Nonetheless, QAT fundamentally relies on differentiating the discontinuous quantization operation, requiring a gradient estimator. The estimation choice impacts convergence, warranting an analysis of available options. We present mathematical limitations of the commonly employed straight-through estimator (STE) and piece-wise linear (PWL) gradients for clipped quantization. We then overcome these limitations by proposing the magnitude-aware derivative (MAD).

4.1 Limitations of Current Gradient Estimation

We start with an analysis of gradient back-propagation using the STE that overlooks (3) to set $\frac{\partial^{(\text{STE})} \mathbb{Q}(x)}{x} = 1$. With clipping, this approximation results in gradient explosion, which causes instability. This is shown via a second order (variance) study similar to that of (He et al., 2015) for initialization. Such analysis is useful for assessing the suitability of back-propagation with quantization (Sakr et al., 2019).

For an arbitrary activation x_l at layer l, we write $\Delta x_l = \frac{\partial \mathcal{L}}{\partial x_l}$ to be the true gradient with respect to the loss function $\mathcal{L}$. This gradient is fundamentally defined as the rate of marginal change in loss function for a marginal change in activation value. Further, let $\Delta^{(\text{STE})} x_l$ be the estimate of Δx_l under STE. The following result holds.

Proposition 4.1 (Gradient explosion with STE)**.** *In an L-layer network, there exists a positive δ such that the ratio of variances of STE gradient $Var\left(\Delta^{(STE)} X_l\right)$ to true gradient $Var\left(\Delta X_l\right)$ at layer l is lower bounded by:*

$$\frac{Var\left(\Delta^{(STE)} X_l\right)}{Var\left(\Delta X_l\right)} \geq (1+\delta)^{L-l} \tag{7}$$

Proof. The proof is provided in Appendix C. The main insight is that STE carries excess variance due to its assigning unity to gradients of clipped weight (see Figure 3). □

The result in Proposition 4.1 highlights an exponential explosion of back-propagated STE gradients. In contrast, the PWL estimator sets $\frac{\partial^{(\text{PWL})}\mathbb{Q}(x)}{x} = \mathbb{1}_{x\in[-s,s]}$ and does not suffer from such gradient explosion. However, weight tensors trained using PWL encounter a partial stoppage of convergence, as early as the first training iteration. Early stopping is equivalent to model size reduction, which can impede the achievable accuracy. The following result holds.

Proposition 4.2 (Convergence stoppage with PWL)**.** *Given a statically clipped $N_{\vec{w}}$-element weight tensor $\vec{w}$, whose gradient is estimated using PWL, only $\tilde{N}_{\vec{w}}^{(i)}$ of its parameters are leaned at iteration i, and the following inequalities hold:*

$$N_{\vec{w}} > \tilde{N}_{\vec{w}}^{(i)} \geq \tilde{N}_{\vec{w}}^{(i+1)} \tag{8}$$

Proof. The proof is provided in Appendix D. The main insight is that PWL repeatedly zeroes out gradients of clipped weights (see Figure 3), halting their updates. □

The monotonic decrease in Proposition 4.2 requires static quantization. Nevertheless, dynamic quantization exhibits a similar, albeit milder, convergence stoppage, where the first strict inequality in (8) also holds.

4.2 Magnitude-aware Differentiation

To formulate an improved gradient estimator, we first present a simple result: rather than treating clipping as a piece-wise selection, we write it as a *magnitude attenuation*.

Proposition 4.3. *The clipping operator is given by:*

$$clip(x, -s, s) = \alpha \cdot x \tag{9}$$

where $\alpha = \mathbb{1}_{\{|x|\leq s\}} + \frac{s}{|x|}\mathbb{1}_{\{|x|>s\}}$

Proof. The result can readily be obtained by replacing the indicator function by its definition, i.e.:

$$\mathbb{1}_{\{|x|\leq s\}} = (1 - \mathbb{1}_{\{|x|>s\}}) = \begin{cases} 1 & \text{if } |x| \leq s \\ 0 & \text{if } |x| > s \end{cases}$$ □

Using Proposition 4.3, we formulate the magnitude-aware derivative (MAD). Treating α as a constant in (9), we obtain:

$$\frac{\partial^{(\text{MAD})}\mathbb{Q}(x)}{x} = \mathbb{1}_{\{|x|\leq s\}} + \frac{s}{|x|}\mathbb{1}_{\{|x|>s\}} \tag{10}$$

In Figure 3, we plot the three gradient estimators: STE, PWL, and MAD, for $s = 1$. They are identical in the discretization region. However, while PWL zeroes out the clipping region, MAD uses a magnitude-aware attenuation factor and is continuous. Therefore, for a MAD-trained weight tensor $\vec{w}$, we do guarantee that $\tilde{N}_{\vec{w}}^{(i)} = N_{\vec{w}}$ at any iteration i, and there is no early stoppage of convergence.

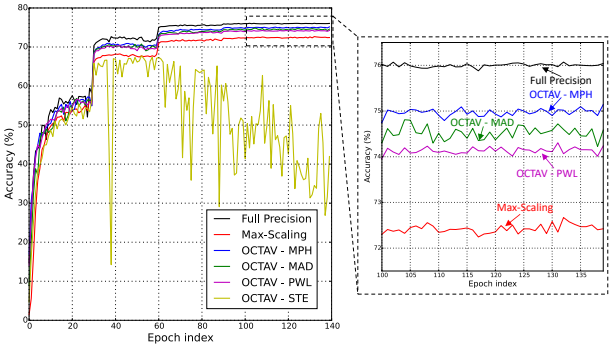


Figure 4: Convergence curves for 4-bit ResNet-50 training-from-scratch including full precision baseline, max-scaling, and OCTAV-enabled QAT. For the last, various gradient estimation strategies discussed in Section 4 are included: STE, PWL, MAD, and MPH.

Table 1: 4-bit ResNet-50 training-from-scratch accuracy

Full Precision	Max-Scaling	OCTAV			
		STE	PWL	MAD	MPH
76.07	72.67	67.75	74.31	74.81	**75.15**

In some measure, PWL and MAD are similar. The former approximates $\frac{\partial \mathbb{1}_{\{|x|\leq s\}}}{\partial s} = \frac{\partial \mathbb{1}_{\{|x|>s\}}}{\partial s} = 0$. This style of approximation is predominant and useful in deep learning, e.g., it is used to train networks with ReLU-like activation functions. Similarly, MAD approximates a combination of indicator functions as being a constant to obtain a useful gradient estimator that evades the limitations of PWL.

Above, we have shown that MAD improves differentiation for quantized weights. There is no clear advantage to using MAD over PWL for activations. The only difference between the two is the occasional zeroing out of activation gradients under PWL. We argue that this is in fact desirable for potential regularization. Indeed, it mimics Dropout (Srivastava et al., 2014) in the backward path. Thus, we generally recommend using MAD for weight gradients and PWL for activation gradients. In our experiments, we term such a combination MAD-PWL Hybrid (MPH).

Finally, we note that MAD is different from *magnitude-aware* gradients in Bi-Real Net (Liu et al., 2018), where a triangular pulse combined with an approximate sign operator estimates gradients for *binarization only*. In contrast, and orthogonally, we formulate a derivative to clipped quantization by analyzing its magnitude attenuation effect.

5 Quantization-aware Training Studies

We conduct numerical experiments to show the impact of our proposed methods. We evaluate training-from-scratch and retraining QAT using ResNet (He et al., 2016) and MobileNet (Sandler et al., 2018; Howard et al., 2019) models

Table 2: Accuracies for training-from-scratch QAT on ImageNet

Network	Full Precision Baseline	$B = 4$		$B = 6$		$B = 8$	
		OCTAV	Max-Scaling	**OCTAV**	Max-Scaling	**OCTAV**	Max-Scaling
ResNet-50	76.07	**75.15**	72.67	76.07	76.01	76.24	76.12
ResNet-18	70.12	**69.17**	65.65	69.78	69.52	70.07	70.19
ResNet-101	77.28	**76.48**	72.53	77.30	77.04	77.31	77.15
MobileNet-V2	71.71	**70.88**	69.17	71.64	71.79	71.71	71.77
MobileNet-V3-Small	65.99	54.68	0.39	**65.02**	60.17	65.98	65.14
MobileNet-V3-Large	72.97	65.86	1.25	**72.12**	69.38	72.89	72.78

deployed on the ImageNet (Deng et al., 2009) dataset for image classification. For fine-tuning QAT, we use BERT (Devlin et al., 2018) language models pretrained on the Wikipedia (Wikimedia Foundation, 2021) and BookCorpus (Zhu et al., 2015) datasets and fine-tuned on Squad v1.1 (Rajpurkar et al., 2016) for question-answering. Our implementations are derived from the NVIDIA 'Deep Learning Examples' repository [3]. All details are in Appendix E.

5.1 Training-from-scratch QAT on ImageNet

To get started, we single out 4-bit ResNet-50 training-from-scratch to point out various aspects of our results. In Figure 4, we compare convergence of test accuracy for the full precision baseline, max-scaled QAT, and OCTAV-enabled QAT, including gradient estimation options from Section 4.

Conforming to the gradient explosion prediction in Proposition 4.1, using the STE to differentiate the clipped quantization operation leads to a clear training instability. Furthermore, using PWL is clearly inferior to MAD, which confirms our analysis on early stoppage of convergence for the former and justifies our proposal for the latter. We obtained a marginal improvement in accuracy over MAD with the MPH scheme, due to its better regularization property described at the end of Section 4.2. All further clipped QAT results use MPH.

Table 1 summarizes ResNet-50 results and lists achieved accuracy for various schemes considered. We find that OCTAV-enabled QAT improves on max-scaling by $\sim$2.5% and achieves a less-than-1% accuracy drop compared to the full precision baseline. This by itself matches the current state-of-the-art in 4-bit QAT (Choi et al., 2018). We emphasize the significance of this result by recalling that no modification to the training recipe was required, such as adding learned parameters or hyperparameter tuning.

Additional training-from-scratch experiments are reported in Table 2 for various networks and precisions. We consider 4-bit, 6-bit, and 8-bit QAT for: ResNet-50, ResNet-18, ResNet-101, MobileNet-V2, MobileNet-V3-Small, and MobileNet-V3-Large. We highlight results yielding high accuracy at low-precision.

For ResNets, 6-bit is enough to achieve close-to-baseline accuracy, even with max-scaling. However, at 4-bit, OCTAV is required to remain within 1% of the baseline while max-scaling leads to a drop in accuracy of up to $\sim$5%.

Similar trends are observed for MobileNet-V2; we find this network easier to quantize compared to MobileNet-V3. We speculate that this is due to the former using ReLU6 activations as opposed to the latter, which uses the Hardswish function (Howard et al., 2019).

Max-scaled QAT of MobileNet-V3 leads to a large drop in accuracy of $\sim$ 3-to-5 % at 6-bit, and further quantization to 4-bit fails to converge and results in near zero accuracy. In contrast, OCTAV-enabled QAT of MobileNet-V3 is successful at 6-bit. A noticeable, but not total, accuracy drop at 4-bit leaves room for improvement. We speculate that accuracy can be recovered through a QAT-friendly training recipe, such as distillation (Park & Yoo, 2020), but this is beyond the scope of our work.

5.2 Retraining ImageNet networks at 4-bit

We also study 4-bit retraining of the aforementioned networks. We use a *shortened* version of the same training recipe, with details included in Appendix E. We include static quantization in our results and follow similar methods as (Wu et al., 2020). We report results for 99.9^{th}, 99.99^{th}, and 99.999^{th} percentile calibration, which were claimed to work well in the retraining setup. For these experiments, Resnet-50 and ResNet-101 are retrained for 15 epochs, while other networks are retrained for 30 epochs.

We also include static-OCTAV, which calibrates clipping scalars using (6). As calibration can be performed offline, we also include results when using a 100-point brute force MSE sweep to set the clipping scalars.

Retraining results are listed in Table 3 and promising OCTAV results are highlighted. A clear trend is observed. Large models, such as ResNets, are much easier to retrain compared to small models, such as MobileNets. This finding is consistent with recent literature (Dbouk et al., 2020) and leads to several novel conclusions.

[3] Code retrieved from: `https://github.com/NVIDIA/DeepLearningExamples`.

Table 3: Accuracies for short 4-bit retraining QAT on ImageNet

Network	Dynamic Quantization		Static Quantization				
	OCTAV	Max-Scaling	**OCTAV**	MSE Sweep	99.9^{th} Perc.	99.99^{th} Perc.	99.999^{th} Perc.
ResNet-50	**75.38**	71.44	**75.84**	75.85	75.66	75.51	75.29
ResNet-18	**69.16**	65.53	**69.18**	69.28	69.04	69.08	68.93
ResNet-101	**76.10**	70.88	**76.96**	77.01	76.79	76.99	76.34
MobileNet-V2	**69.32**	66.94	0.66	0.93	1.72	2.14	2.76
MobileNet-V3-Small	**53.52**	0.10	0.43	0.58	0.65	0.10	1.46
MobileNet-V3-Large	**64.97**	39.46	0.39	0.30	0.34	0.71	0.57

Table 4: Accuracies for long 4-bit retraining QAT on ImageNet using OCTAV

Network	Dynamic-OCTAV	Static-OCTAV
ResNet-50	**76.21**	**76.46**
ResNet-18	**69.90**	**70.13**
ResNet-101	**76.84**	**77.48**
MobileNet-V2	**71.23**	1.21
MobileNet-V3-Small	**58.93**	0.80
MobileNet-V3-Large	**69.21**	0.60

For large models, static quantization is most suitable, though OCTAV-enabled dynamic quantization also yields high accuracy, within $\sim$1% of the baseline. As large models are easy to quantize, we speculate that a low-precision solution exists near the pretrained starting point. When an aspect of the parameters (the clipping scalars) is forced to be static, the model rapidly settles around a close solution to this starting point. In the case of ResNets, this solution is highly accurate. Highest accuracy is consistently achieved using static-OCTAV and the MSE sweep, both similar as expected.

For small models, such as MobileNets, static quantization is unfit, yielding near-zero accuracy. For such models, a good quantized solution is unlikely to be close to the pretrained starting point. Thus, retraining requires dynamic quantization to track changes in tensor distributions occurring as the model adapts to low precision. Furthermore, OCTAV is found to be far superior to other strategies and can recover some accuracy across all MobileNets.

Interestingly, we find that retraining can be less accurate than training-from-scratch. For MobileNet-V2, OCTAV reaches 69.32% for the former and 70.88% for the latter. However, this is due to the shorter training time of 30 epochs used in retraining. When equalizing QAT time and retraining for 300 epochs using OCTAV, we obtain an accuracy of **71.23**%. Thus, using a pretrained starting-point has some merits in spite of the large amounts of quantization noise suffered by MobileNets at low precision.

To further explore the potential of OCTAV and to understand the merits of retraining, we also perform *long* 4-bit retraining of all ImageNet models considered. We retrain with OCTAV both dynamically and using static calibration, following the above setup. The full long retraining recipes are identical to those employed for training-from-scratch experiments in Section 5.1 but for two aspects. First, the starting point is the pretrained model in full precision; and, second, the starting learning rate value is attenuated by a factor of 10^{-2} compared to that of the training-from-scratch recipe. For these experiments, ResNets and MobileNets are retrained for 150 and 300 epochs, respectively.

Accuracies for long 4-bit retraining QAT are reported in Table 4 and promising results are highlighted. Compared to results in Table 3, long retraining always leads to a noticeable improvement, which establishes the merits of prolonged training. Consistent with earlier observations, static-OCTAV is found to yield highest accuracy for ResNets, but suffers catastrophic degradation on MobileNets. Similarly, dynamic-OCTAV once more yields high accuracy for all networks.

For ResNets, the achieved accuracy is either close to, or superior to that of the full precision baseline. This indicates that OCTAV's quantization effects are so small and fundamentally fall below the training algorithm's inherent noise floor. Our results are comparable in absolute terms to the state-of-the-art reported accuracies for 4-bit QAT on ResNets established using learned quantization such as PACT (Choi et al., 2018) and LSQ (Esser et al., 2019). Comparing relative accuracy to the starting full precision baseline, our results improve the state-of-the-art. For instance, for ResNet-50, 4-bit OCTAV improves the baseline accuracy by $\sim$0.4% while 4-bit LSQ degrades it by $\sim$0.2%.

Trends of long MoblileNet retraining are similar to those previously observed for training-from-scratch and short retraining. Failure of static calibration, in spite of prolonged training time, illustrates the importance of tracking tensor statistics when retraining small models. The improvement in final accuracy under dynamic-OCTAV is promising but not enough in the case of Mobilenet-V3. As mentioned in Section 5.1, further QAT techniques may be required to achieve close-to-baseline accuracy.

5.3 Fine-tuning QAT of BERT Models on Squad

Finally, we study fine-tuning QAT of BERT-Base and BERT-Large on Squad v1.1. As research on low-precision transformer networks is still in the early stages, we study QAT using every bit-width from 4 to 8 bits to understand the

Table 5: Accuracies for fine-tuning BERT-Base & BERT-Large on Squad v1.1

Network		BERT-Large (Baseline Accuracy: 91.00)					BERT-Base (Baseline Accuracy: 88.24)				
# of bits B		4	5	6	7	8	4	5	6	7	8
Dynamic Quantization	**OCTAV**	**87.09**	**89.77**	**90.51**	**90.81**	**90.78**	**84.51**	**86.30**	**87.43**	**88.28**	**88.34**
	Max-Scaling	6.92	80.06	87.71	90.04	90.48	11.51	78.97	85.17	87.46	88.01
Static Quantization	**OCTAV**	**87.08**	**89.54**	**90.60**	**90.79**	**90.61**	**83.60**	**85.82**	**87.14**	**87.67**	**88.02**
	MSE Sweep	85.54	89.77	90.39	90.80	90.55	81.82	84.16	87.14	87.68	87.97
	99.9^{th} Perc.	86.98	89.79	89.99	90.07	90.11	81.06	85.78	86.73	86.84	87.34
	99.99^{th} Perc.	6.90	87.63	90.38	90.79	90.33	67.90	83.20	86.78	87.60	87.94
	99.999^{th} Perc.	4.56	5.66	89.76	90.44	90.83	26.85	82.15	86.27	87.51	88.08

difficulties in quantizing these networks.

We employ the same strategies of dynamic and static quantization as were used for retraining in Section 5.2. For BERT-Base, we compare calibration times on a CPU when using OCTAV and the MSE sweep. Details are included in Appendix F and the former is found to be $\sim 10\times$ faster.

Fine-tuning results are included in Table 5 where various F1 scores are reported, and OCTAV results are highlighted.

For both networks, 7-bit or more is enough to match the baseline accuracy, regardless of the quantization strategy. At 6-bit, only dynamic-OCTAV yields an accuracy within 1% of the baseline for the two networks. At lower precision, dynamic-OCTAV exhibits the most graceful degradation in accuracy, reaching a drop of $\sim 4\%$ for both networks at 4-bit. As fine-tuning consists of re-adapting the model to a new task, it is reasonable to expect changes in tensor statistics, warranting the use of dynamic quantization for tracking. Thus, dynamic-OCTAV being consistently superior to all other strategies is expected.

Static quantization performs generally well, and static-OCTAV is clearly its best candidate. Its accuracy gracefully degrades to a 4-to-5 % drop compared to the baseline. A surprising result is the definite superiority of static-OCTAV compared to the MSE sweep. It turns out that, for a few activation layers, the clipped quantization MSE is not convex, leading to divergent solutions for the two strategies. This issue occurs due to the presence of large outliers. The local minimum closest to zero is selected by OCTAV while the MSE sweep chooses to zero out all small values. This phenomenon is described in Appendix G. The better choice made by OCTAV leads to $\sim 1.5\%$ better accuracy compared to the sweep at 4-bit.

6 Discussion

6.1 Current limitations and directions for future work

We have shown, analytically and empirically, that OCTAV-enabled QAT improves accuracy of low-precision training without requiring modifications to the learning algorithm. As efforts to scale down DNN precision continue, an interesting avenue of future research is to combine OCTAV with quantization-dedicated training recipes, such as distillation, to increase accuracy even further.

In addition, while the OCTAV overhead is at least an order of magnitude lower than an equivalent sweep, it is still pronounced. Research to reduce this overhead is needed to accelerate FQT in an accuracy-optimal manner. Similarly, for DNN inference acceleration, often hindered by dynamic quantization, techniques to match OCTAV accuracy in the static setup are desired. Our proposed static-OCTAV calibration strategy is one step in that direction.

Finally, the theoretical technique employed in this work may be applied to complexity reduction beyond quantization. We have formulated quantization noise as an objective function to be minimized on the fly using the Newton-Raphson algorithm. Similarly, other hardware-aware models, such as those for sparsification, can be rapidly optimized for reduced complexity. Such work can be impactful in the context of neural architecture search for hardware-efficient DNNs.

6.2 Conclusion

We have proposed OCTAV for enabling QAT with minimal quantization noise for each tensor at every iteration. We have also analyzed current clipped gradient estimators and proposed magnitude-aware differentiation as a tool to further improve QAT. Empirically, we have demonstrated that our methods lead to state-of-the-art accuracy in low-precision training of DNNs, without modifying the learning algorithm. Our contributions are an important step in the advancement of low-complexity deep learning.

Acknowledgement

The authors wish to thank Ben Keller and Hao Wu from NVIDIA for useful discussions.

References

Abdolrashidi, A., Wang, L., Agrawal, S., Malmaud, J., Rybakov, O., Leichner, C., and Lew, L. Pareto-optimal quantized resnet is mostly 4-bit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern*

Recognition, pp. 3091–3099, 2021.

Bianco, S., Cadene, R., Celona, L., and Napoletano, P. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.

Choi, J., Wang, Z., Venkataramani, S., Chuang, P. I.-J., Srinivasan, V., and Gopalakrishnan, K. PACT: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

Choi, Y., Choi, J., El-Khamy, M., and Lee, J. Data-free network quantization with adversarial knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 710–711, 2020.

Courbariaux, M. et al. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pp. 3123–3131, 2015.

Dai, S., Venkatesan, R., Ren, M., Zimmer, B., Dally, W., and Khailany, B. Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference. *Proceedings of Machine Learning and Systems*, 3, 2021.

Dbouk, H., Sanghvi, H., Mehendale, M., and Shanbhag, N. Dbq: A differentiable branch quantizer for lightweight deep neural networks. In *European Conference on Computer Vision*, pp. 90–106. Springer, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. Learned step size quantization. In *International Conference on Learning Representations*, 2019.

Goel, M. and Shanbhag, N. Finite-precision analysis of the pipelined strength-reduced adaptive filter. *Signal Processing, IEEE Transactions on*, 46(6):1763–1769, 1998.

Gonugondla, S. K., Sakr, C., Dbouk, H., and Shanbhag, N. R. Fundamental limits on the precision of in-memory architectures. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pp. 1737–1746, 2015.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., and Dally, W. J. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

He, K. et al. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

Hubara, I. et al. Binarized neural networks. In *Advances in Neural Information Processing Systems*, pp. 4107–4115, 2016.

Jain, S., Venkataramani, S., Srinivasan, V., Choi, J., Gopalakrishnan, K., and Chang, L. BiScaled-DNN: Quantizing long-tailed datastructures with two scale factors for deep neural networks. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2019.

Köster, U., Webb, T., Wang, X., Nassar, M., Bansal, A. K., Constable, W., Elibol, O., Hall, S., Hornof, L., Khosrowshahi, A., et al. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 1740–1750, 2017.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Lee, E. H., Miyashita, D., Chai, E., Murmann, B., and Wong, S. S. Lognet: Energy-efficient neural networks using logarithmic computation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5900–5904. IEEE, 2017.

Lin, Y. et al. PredictiveNet: an energy-efficient convolutional neural network via zero prediction. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017.

Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., and Cheng, K.-T. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced

training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 722–737, 2018.

Lloyd, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Nagel, M., Baalen, M. v., Blankevoort, T., and Welling, M. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1325–1334, 2019.

Park, E. and Yoo, S. Profit: A novel training method for sub-4-bit mobilenet models. In *European Conference on Computer Vision*, pp. 430–446. Springer, 2020.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *NeurIPS Workshop on Automatic Differentiation*, 2017.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Sakr, C. and Shanbhag, N. R. Per-tensor fixed-point quantization of the back-propagation algorithm. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Sakr, C. and Shanbhag, N. R. Signal processing methods to enhance the energy efficiency of in-memory computing architectures. *IEEE Transactions on Signal Processing*, 69:6462–6472, 2021.

Sakr, C. et al. Analytical guarantees on numerical precision of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3007–3016, 2017.

Sakr, C. et al. Accumulation bit-width scaling for ultra-low precision training of deep networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Srivastava, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Sun, X., Choi, J., Chen, C.-Y., Wang, N., Venkataramani, S., Srinivasan, V., Cui, X., Zhang, W., and Gopalakrishnan, K. Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks. In *NeurIPS*, 2019.

Taigman, Y. et al. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

Tambe, T., Yang, E.-Y., Wan, Z., Deng, Y., Reddi, V. J., Rush, A., Brooks, D., and Wei, G.-Y. Adaptivfloat: A floating-point based data type for resilient deep learning inference. *arXiv preprint arXiv:1909.13271*, 2019.

Wang, N., Choi, J., Brand, D., Chen, C.-Y., and Gopalakrishnan, K. Training deep neural networks with 8-bit floating point numbers. In *Advances in Neural Information Processing Systems*, 2018.

Widrow, B. and Kollár, I. Quantization noise. *Cambridge University Press*, 2:5, 2008.

Wikimedia Foundation. Wikimedia downloads, 2021. URL `https://dumps.wikimedia.org`.

Wu, H., Judd, P., Zhang, X., Isaev, M., and Micikevicius, P. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

Zhang, D., Yang, J., Ye, D., and Hua, G. LQ-Nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 365–382, 2018.

Zhao, J., Dai, S., Venkatesan, R., Liu, M.-Y., Khailany, B., Dally, B., and Anandkumar, A. Low-precision training in logarithmic number system using multiplicative weight update. *arXiv preprint arXiv:2106.13914*, 2021.

Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

Supplementary Material

The main section of our paper contains all necessary content to understand our work. This supplementary section is provided to the interested reader as an avenue to go further. Here, we include an extension of all mathematical results to the unsigned quantization case, proofs of various theorems and propositions, implementation details needed to reproduce our experimental results, timing comparisons between OCTAV and the MSE sweep, and an investigation on the occurrence of non-convex clipped quantization MSE.

A Results for Unsigned Quantization

All mathematical results presented in the main paper made the assumption that any data being quantized was signed, i.e., lying in $[-s_{\max}, s_{\max}]$. A study of the unsigned case is also relevant, as unsigned activations (such as ReLU) are commonly employed. Hereafter, we list all required modifications to our mathematical results for the unsigned case, i.e., quantization over $[0, s_{\max}]$.

We start with the max-scaled quantization operation, which, rather than (2), is given by:

$$\mathbb{Q}(x) = s_{\max} \cdot 2^{-B} \cdot \text{round}\left(x \cdot 2^B / s_{\max}\right)$$

and its MSE, derived from an additive model of quantization noise is given by $J = s_{\max}^2 \frac{4^{-B}}{12}$. Effectively, when data is unsigned, there is no sign bit, and instead an additional least-significant-bit. Thus, for an identical value of $s_{\max}$, an unsigned quantizer has a twice smaller quantization step and consequently a $4\times$ smaller MSE than its signed counterpart.

Similarly, the unsigned clipped quantization operation, using a clipping scalar $s < s_{\max}$ is given by:

$$\begin{aligned}\mathbb{Q}(x) &= \text{Rclip}\left(s \cdot 2^{-B} \cdot \text{round}\left(x \cdot 2^B / s\right), s\right) \\ &= \begin{cases} s \cdot 2^{-B} \cdot \text{round}\left(x \cdot 2^B / s\right) & \text{if } x \in [0, s] \\ s \quad \text{if } x > s \end{cases}\end{aligned}$$

where Rclip() is the right clipping operator used in lieu of the usual two-sided clipping operator. Further, the unsigned clipped quantization MSE can be re-derived, and rather than (4), it is given by:

$$J(s) = \frac{4^{-B}}{12} s^2 \int_0^s f_X(x)dx + \int_s^\infty (s-x)^2 f_X(x)dx$$

where we note the factor of 12 rather than 3 in the first term and the use of $f_X()$ rather than $f_{|X|}()$; since the data is unsigned, there is no need to consider absolute values.

With the above expression for the unsigned clipped quantization MSE, Theorem 3.1 can be re-derived in the same manner as Appendix B below, and it can be readily shown that the recursion in (5) is modified to:

$$s_{n+1} = \frac{\mathbb{E}\left[X \cdot \mathbb{1}_{\{X > s_n\}}\right]}{\frac{4^{-B}}{12}\mathbb{E}\left[\mathbb{1}_{\{X \le s_n\}}\right] + \mathbb{E}\left[\mathbb{1}_{\{X > s_n\}}\right]}$$

and similarly, the recursion (6) in Corollary 3.2 is modified to:

$$s_{n+1} = \frac{\sum_{x \in \vec{t}}\left[x \cdot \mathbb{1}_{\{x > s_n\}}\right]}{\frac{4^{-B}}{12}\sum_{x \in \vec{t}}\left[\mathbb{1}_{\{0 < x \le s_n\}}\right] + \sum_{x \in \vec{t}}\left[\mathbb{1}_{\{x > s_n\}}\right]}$$

where in both cases, we note the use of 12 rather than 3 in the first denominator term, and the lack of need for absolute values.

Finally, we append to Proposition 4.3 the result related to the right clipping operator, which is given by:

$$\text{Rclip}(x, s) = \left(\mathbb{1}_{\{x \le s\}} + \frac{s}{x}\mathbb{1}_{\{x > s\}}\right) \cdot x$$

and, consequently, for unsigned activations, rather than (10), the MAD estimator is given by:

$$\frac{\partial^{(\text{MAD})}\mathbb{Q}(x)}{x} = \mathbb{1}_{\{x \le s\}} + \frac{s}{x}\mathbb{1}_{\{x > s\}}$$

with the only modification made being the removal of absolute values.

B Proof of Theorem 3.1

Our strategy is to use the Newton-Raphson algorithm. The algorithm consists of selecting a random guess s_1 and iteratively computing $\{s_n\}_{n>1}$ until convergence using the following recursion:

$$s_{n+1} = s_n - \frac{J'(s_n)}{J''(s_n)} \tag{11}$$

This recursion requires expressions for the first and second derivatives of the clipped quantization MSE $J(s)$. Obtaining closed form expressions for these derivatives is possible, but complicated. Thus, we first re-write the clipped quantization MSE in (4) as follows:

$$J(s) = \frac{4^{-B}}{3} s^2 \mathbb{E}\left[\mathbb{1}_{\{|X| \le s\}}\right] + \mathbb{E}\left[(s - |X|)^2 \mathbb{1}_{\{|X| > s\}}\right] \tag{12}$$

No approximation was made. Indeed, (4) and (12) are identical by virtue of the relationship between integrals of distributions and expected values of indicator functions. More generally, we used:

$$\int_{\mathcal{R}} g(x) f_X(x) dx = \mathbb{E}\left[g(X)\mathbb{1}_{\{X \in \mathcal{R}\}}\right]$$

for a given interval $\mathcal{R}$ and function $g()$. We now note that the expression for $J(s)$ in (12) is easier to differentiate. Nonetheless, it requires differentiating the indicator function ($\mathbb{1}_{\{|X|\leq s\}}$ and $\mathbb{1}_{\{|X|>s\}}$). This function is technically non-differentiable; however, it is customary in deep learning to treat it as a piecewise linear function. For instance, such is the way the ReLU function is handled for training neural networks using the back-propagation algorithm. In our case, we do make the piecewise linear approximation to obtain: $\frac{\partial \mathbb{1}_{\{|X|\leq s\}}}{\partial s} = \frac{\partial \mathbb{1}_{\{|X|>s\}}}{\partial s} = 0$. Because the derivative is taken inside an expectation operation in (12), this approximation is generally valid. The validity of the approximation would fail if one of the candidates $\{s_n\}_{n>1}$ coincides with a point of positive mass. Thus, to be certain that the algorithm converges correctly, one requirement on the distribution $f_X()$ would be to have no points of positive mass in the vicinity of s_1. This is by no means a strong condition, and we expect all data of interest to satisfy this condition anyway. Next, we simply compute the first and second order derivatives of $J(s)$ in (12) to obtain:

$$J'(s) = \frac{2 \cdot 4^{-B}}{3} s\mathbb{E}\left[\mathbb{1}_{\{|X|\leq s\}}\right] + 2\mathbb{E}\left[(s-|X|)\mathbb{1}_{\{|X|>s\}}\right]$$
$$J''(s) = \frac{2 \cdot 4^{-B}}{3} \mathbb{E}\left[\mathbb{1}_{\{|X|\leq s\}}\right] + 2\mathbb{E}\left[\mathbb{1}_{\{|X|>s\}}\right]$$

which we insert into (11) to obtain (5). Observe that the second derivate $J''(s)$ is a positive quantity, proving the convexity of $J(s)$ and justifying the use of the Newton-Raphson method. It might be useful to notice the following:

$$J'(s) = s \cdot J''(s) - 2\mathbb{E}\left[|X| \cdot \mathbb{1}_{\{|X|>s_n\}}\right]$$

C Proof of Proposition 4.1

Note that $\text{Var}\left(\Delta X_l\right) = k_l \mathbb{E}\left[\left|\frac{\partial \mathbb{Q}(X_l)}{\partial X_l}\right|^2\right]$ where k_l accumulates the back-propagated gradient variance until the quantization step at layer l. It follows that:

$$\frac{\text{Var}\left(\Delta^{(\text{STE})} X_l\right)}{\text{Var}\left(\Delta X_l\right)} = \prod_{i=l}^{L} \frac{\mathbb{E}\left[\left|\frac{\partial^{(\text{STE})} \mathbb{Q}(X_i)}{\partial X_i}\right|^2\right]}{\mathbb{E}\left[\left|\frac{\partial \mathbb{Q}(X_i)}{\partial X_i}\right|^2\right]}$$

Indeed, at every layer, the gradient estimator replaces the true gradient. Further, note that $\mathbb{E}\left[\left|\frac{\partial^{(\text{STE})} \mathbb{Q}(X_i)}{\partial X_i}\right|^2\right] = 1$ by definition of the STE (see Section 4). By virtue of the law of total expectation, we have:

$$\mathbb{E}\left[\left|\frac{\partial \mathbb{Q}(X_i)}{\partial X_i}\right|^2\right]$$
$$= \mathbb{E}\left[\left|\frac{\partial \mathbb{Q}(X_i)}{\partial X_i}\right|^2 \middle| |X_i| \leq s_i\right] \Pr\left(|X_i| \leq s_i\right)$$
$$+ \mathbb{E}\left[\left|\frac{\partial \mathbb{Q}(X_i)}{\partial X_i}\right|^2 \middle| |X_i| > s_i\right] \Pr\left(|X_i| > s_i\right)$$

In the discretization region, and for sufficiently small quantization step, which is in line with our paper's assumptions, we have: $\mathbb{E}\left[\left|\frac{\partial \mathbb{Q}(X_i)}{\partial X_i}\right|^2 \middle| |X_i| \leq s_i\right] = 1$. In contrast, in the clipping regions, we have $\mathbb{E}\left[\left|\frac{\partial \mathbb{Q}(X_i)}{\partial X_i}\right|^2 \middle| |X_i| > s_i\right] = 0$. We obtain:

$$\mathbb{E}\left[\left|\frac{\partial \mathbb{Q}(X_i)}{\partial X_i}\right|^2\right] = \Pr\left(|X_i| \leq s_i\right)$$
$$< 1 = \mathbb{E}\left[\left|\frac{\partial^{(\text{STE})} \mathbb{Q}(X_i)}{\partial X_i}\right|^2\right]$$

The strict inequality occurs when using any clipped quantizer, but not the max-scaled quantizer. Thus, we obtain:

$$\frac{\mathbb{E}\left[\left|\frac{\partial^{(\text{STE})} \mathbb{Q}(X_i)}{\partial X_i}\right|^2\right]}{\mathbb{E}\left[\left|\frac{\partial \mathbb{Q}(X_i)}{\partial X_i}\right|^2\right]} = 1 + \delta_i$$

where $\delta_i > 0$. We assign $\delta = \min_{i=1\ldots L} \delta_i$ to obtain the desired result in (7) of Proposition 4.1.

D Proof of Proposition 4.2

In Proposition 4.2, static clipped quantization is assumed, thus, the quantization scalar satisfies $s < s_{\max}$. It follows that, there exists at least one weight element $w_\circ \in \vec{w}$ such that its initial value at iteration 1 satisfies $\left|w_\circ^{(1)}\right| > s$. Thus, its first PWL gradient estimate at iteration 1 is given by: $\Delta^{(\text{PWL})} w_\circ^{(1)} = 0$. Consequently, $w_\circ^{(2)} = w_\circ^{(1)}$ and $\Delta^{(\text{PWL})} w_\circ^{(2)} = 0$. More generally, we may use induction to obtain that $\forall i, \Delta^{(\text{PWL})} w_\circ^{(i)} = 0$ and the weight value is 'stuck at initialization', i.e., $w_\circ^{(i)} = w_\circ^{(1)}$. This means that $w_\circ$ is never learned and therefore $N_{\vec{w}} > \tilde{N}_{\vec{w}}^{(i)}$, which is the first inequality in Proposition 4.2. This first part of the result also applies to dynamic quantization. Furthermore, if at iteration i there is at least one $w_* \in \vec{w}$ such that $\left|w_*^{(i)}\right| < s$ and $\Delta^{(\text{PWL})} w_*^{(i)}$ causes the next iteration weight magnitude

to be $\left|w_*^{(i+1)}\right| > s$, then $\tilde{N}_{\vec{w}}^{(i)} > \tilde{N}_{\vec{w}}^{(i+1)}$. If no such weight exists, then $\tilde{N}_{\vec{w}}^{(i)} = \tilde{N}_{\vec{w}}^{(i+1)}$. This completes the proof of the second inequality in Proposition 4.2. At any iteration i, at best the number of learnable parameters remains constant, and at worst, it decreases.

E Experimental Implementations Details

In this section, we discuss all details behind our implementations in Section 5. These include training recipes, tensor quantization specifics, and static quantization calibration details. All experiments were implemented using DGX-1 Volta machines.

E.1 Baseline Training Recipes

The following recipes were used for obtaining ImageNet **training-from-scratch** results included in Section 5.1. All models were trained using momentum SGD using 8 V100 GPUs. The following parameters are the defaults suggested in the 'Deep Learning Examples' repository [4].

ResNet training used the following parameters: an initial learning of 0.1, a per-GPU batch-size of 64, a momentum factor of 0.9, a weight decay factor of 1e-4, and a learning rate decay factor of 0.1 every 30 epochs. ResNet-50 and ResNet-18 were trained for 150 epochs, while ResNet-101 was trained for 80 epochs.

MobileNet training used the following parameters: an initial learning rate of 0.2, a per-GPU batch-size of 128, a momentum factor of 0.9, a learning weight decay factor of 0.1 every 100 epochs, and a total number of training epochs of 300. Weight decay factors of 4e-5 and 1e-5 were used for MobileNet-V2 and MobileNet-V3, respectively.

Shortened versions of the above recipes were used for **retraining** discussed in Section 5.2. Furthermore, models were retrained using 4 V100 GPUs. More specifically, ResNets and MobileNets were retrained using an initial learning rate of 1e-4 and 1e-2, respectively. ResNet-50 and ResNet-101 were retrained for 15 epochs with a learning rate decay of 0.1 every 5 epochs. All other models were retrained for 30 epochs with a learning rate decay of 0.1 every 10 epochs.

For **fine-tuning** of BERT models on Squad, we used 1 V100 GPU and referred to the default parameters suggested in the 'Deep Learning Examples' repository [5]. Specifically, we fine-tune for 2 epochs and use: a batch of 4, a learning rate of 3e-5, and a maximum sequence length of 384.

[4]Code retrieved from `https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Classification/ConvNets`.

[5]Code retrieved from `https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/LanguageModeling/BERT`.

E.2 Tensor Quantization Specifics

It is important to explicitly describe various options for quantization granularity. The following are commonly employed methods:

- **Tensor scaling**, where one scalar s is used to quantize all elements in a tensor $\vec{t}$.
- **Per-output-channel (POC) scaling**, which applies to weights in convolutional layers. A convolutional weight tensor $\vec{w}$ has a dimensionality of (K, C, R, S) where K and C are the output and input number of channels, respectively, and R and S are the kernel height and width, respectively. A vector $\vec{s}$ of K scalars is used to quantize $C \times R \times S$ elements at a time. Specifically, the tensor $\vec{w}$ is flattened into a matrix of dimensionality $(K, C \times R \times S)$ and every row vector is quantized using the scalar in $\vec{s}$ of appropriate index.
- **Per-output-feature (POF) scaling**, which applies to weights in linear (fully connected) layers. A weight matrix $\vec{w}$ has dimensionality (O, I) where O and I are the output and input number of features, respectively. A vector $\vec{s}$ of O scalars is used to quantize the row vectors of $\vec{w}$ in a similar fashion as the POS scaling case.

The advantage of POS and POF scaling is the finer granularity of quantization, which can boost accuracy. The above three methods are well-adapted to GPU and deep learning accelerator datapaths (Wu et al., 2020). In addition, the overhead of quantization metadata is small enough that in our work we assume all scalars to be in full precision (Dai et al., 2021).

In our experiments, we use POC scaling for convolutional weight quantization in all ResNets and MobileNets, and POF scaling for the weight layers in BERT models. Tensor scaling is used for all other tensors, which include: activations and fully connected layers in ResNets and MobileNets, and activations in BERT models. For the last, we define activations as being inputs to the various linear layers in the transformer blocks, as well as inputs to the Batch Matrix Multiply (BMM) operations.

Finally, we note that first (convolutional) and last (fully connected) layers in ResNets and MobileNets are always quantized to 8-bit, as is customary for such networks (Choi et al., 2018). However, projection layers (also known shortcut connections) are quantized to low precision (4-bit, 6-bit, and 8-bit, depending on the experiment).

E.3 Static Quantization Calibration Details

In this section, we describe the methodology to calibrate clipping scalars for retraining and fine-tuning experiments

in Section 5.2 and 5.3, respectively.

Weight calibration is done by simply using the pretrained weights at various layers and computing various clipping scalar candidates. Specifically, the candidates used are the OCTAV and MSE sweep predicted scalars, as well as the 99.9^{th}, 99.99^{th}, and 99.999^{th} percentiles.

For activations, we sample 5 random input batches from the training set. For each, we compute the various candidates above at every layer. Thus, for each strategy, we obtain a collection of 5 candidates for every activation tensor. The average of these candidates is used as a calibrated clipping scalar. For BERT-Base, we sample and average over 10 random inputs from the training set.

We define the MSE Sweep as being a 100-point sweep. Specifically, for every tensor, we evaluate the sweep over $\{\frac{k}{100} \cdot s_{\max}\}_{k=1}^{100}$. In the case of POC and POF scaling, we run the sweep over $\{\frac{k}{100} \cdot \vec{s}_{\max}\}_{k=1}^{100}$ where $\vec{s}_{\max}$ is the vector of row-wise maxima in the tensors. This is an approximation leading to a weaker solution than OCTAV, which, thanks to broadcasting, can perform row-wise optimization with no overhead. Nonetheless, we find both to yield similar results, as highlighted in our results in Section 5.2.

F OCTAV vs. Brute Force Sweep Speed Comparison

We compare times taken to calibrate BERT-Base tensors when using OCTAV vs. the MSE sweep. The calibration was done on an Intel Xeon CPU, using the NumPy package, and no code optimization was performed for either. We choose to measure time on a CPU to provide the fairest and most accurate comparison possible. Indeed, on a GPU, significant noise in time measurement can occur due to various communication of data between GPU and host that may be required. As OCTAV only requires tensor operations, its GPU speedup over MSE sweeps is expected to be even greater than what we report here.

In BERT-Base, there are 74 weight layers and thus 74 weight tensors to be processed. For each of these layers, we also process an input activation tensor. As we use 10 random input batches, we obtain 740 activation tensors to process. Furthermore, there are 24 BMM operations, each of which has two operands that are added to the list of activations. Thus, we get another 48 activation tensors as BMM operands per input batch, for a total of 480 additional activation tensors. Thus, we process 1220 activation tensors overall.

As we perform calibration for $B = 4, 5, 6, 7, 8$, we report average times per-precision. For each tensor, we start a timer as soon as we invoke our calibration (OCTAV vs. MSE sweep) routine and stop it as soon as execution terminates.

Our measurements are listed in Table 6. We find that OCTAV processes one weight tensor in 0.191 sec., on average completing all weight calibration in only 14 sec. In contrast, the MSE sweep requires over 2 mins. and is 10.2 $\times$ slower. For activations, OCTAV processes one tensor in 0.497 sec. on average, and completes all activation calibration in just over 10 mins. This corresponds to the calibration of 1220 tensors on a CPU. In contrast, the MSE sweep is 6.3 $\times$ slower and requires over 1 hour to terminate. Interestingly, activation calibration speed-up is less pronounced than that of weights. We speculate that this is due to the activation tensors being much larger so that much of the execution time goes into data movement from memory, which amortizes the speed-up.

Table 6: Calibration times comparison of OCTAV vs. MSE Sweep for BERT-Base

		Weight	Activation
Per-Tensor Avg. (seconds)	OCTAV	0.191	0.497
	MSE Sweep	1.943	3.141
Total Calib. (h:mm:ss)	OCTAV	0:00:14	0:10:06
	MSE Sweep	0:02:24	1:03:52
Speed-up		10.2×	6.3×

G When is MSE not Convex?

Comparing static quantization using OCTAV vs. MSE sweep results for BERT models in Section 5.3 reveals unexpected results. As shown in Table 5, OCTAV calibration at a low precision yields a significantly higher accuracy than the MSE sweep. As both methods are supposed to return similar solutions, we investigated the reason of this discrepancy.

In Figure 5(a) we plot the calibrated clipping scalar for activations (linear layers input only, BMM operands excluded) as a function of layer index when using OCTAV and the MSE sweep. We observe that the solution is identical almost everywhere, except at a few layers. Specifically, at layers 28, 33, 40, and 64, the OCTAV-calibrated scalar is noticeably smaller than its MSE sweep counterpart. To understand this phenomenon, we carefully compared the data at layers 63 and 64. For the former, the two calibrated strategies converge to the same solution, but for the latter, OCTAV returns a clipping scalar of $\sim$8 while the MSE sweep returns $\sim$38.

In Figure 5(b,c), we plot the empirical probability distribution function (PDF) of the tensors used for calibration at layer 63 and 64, respectively. These tensors correspond to the 10 random input batches selected for calibration, each of which is represented by one color. A stark contrast in distribution is observed: layer 63 appears to be typical with most of the density concentrated around zero, while layer 64 data has large outliers around the value 350.

In Figure 5(d,e), we plot the 4-bit clipped quantization MSE as a function of clipping scalars for layers 63 and 64, re-

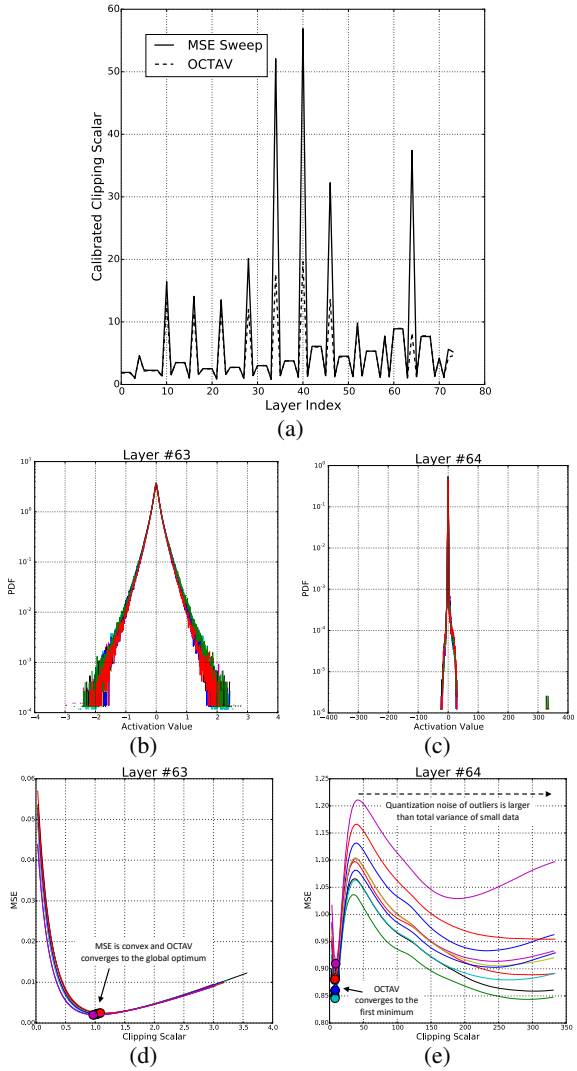


Figure 5: Investigation of 4-bit clipped quantization MSE convexity in BERT-Base: (a) calibrated clipping scalars returned by OCTAV and MSE sweep for activation layers, (b) PDF of activation tensors at layer 63, (c) PDF of activation tensors at layer 64, (c) MSE vs. clipping scalar at layer 63, and (d) MSE vs. clipping scalar at layer 64. Different colors correspond to the 10 different input batches used for calibration. Activation tensors considered are input to the linear layers in BERT-Base, while BMM operands are not shown.

spectively. We also pinpoint the predicted clipping scalar by OCTAV for each of the 10 tensors. As expected, for layer 63, the MSE is a convex function and OCTAV accurately converges to its global optimum. In contrast, layer 64 exhibits a more complex behavior. The MSE does include a local minimum close to zero, which corresponds to the trade-off between discretization and clipping noise, and to which OCTAV converges to. Beyond that point, the MSE increases as the clipping scalar increases, and this is due to an increase in the discretization step. At some point, non-outlier data is bound to become smaller than the magnitude of the least significant bit and thus be quantized to zero. At this point, the quantization noise related to non-outlier data quantization is equal to its total variance and no longer increases when the clipping scalar increases. However, with larger clipping scalar, the quantization noise of outliers themselves decreases, which leads to the MSE decreasing again. The MSE therefore has a second minimum, which is close to the maximum scalar and only caters for outlier representation. Furthermore, for one of the 10 tensors processed, in this case, the one corresponding to the green line in Figure 5(d), the MSE at this second minimum is smaller than that of the first one. This second minimum is thus selected by the MSE sweep and skews the calibrated scalar towards the outliers.

In conclusion, when tensors have large outliers, the MSE sweep may return a calibrated scalar that caters only for outliers and zeroes out all small values. In contrast, OCTAV converges to the minimum closest to zero, which balances discretization and clipping noise of all data in the tensor. This is in fact the only minimum identified by $J(s)$ in (4), due to the additive noise model assumption. Thus, OCTAV is guaranteed to converge to this first minimum. Intuitively, the solution returned by OCTAV is desired, as it caters for *all* the data in the tensor. This intuition concurs with our experimental results in Section 5.3. Indeed, static-OCTAV was found to yield a noticeably higher accuracy than the MSE sweep for both BERT models at low precision (4-to-6-bit).