Quantile Quantization

Ralph Estanboulieh
School of Engineering and Applied Sciences
Harvard University
Cambridge, Massachusetts
ralphestanboulie@college.harvard.edu

Ryan Kim
School of Engineering and Applied Sciences
Harvard University
Cambridge, Massachusetts
ryankim@college.harvard.edu

Luke Kenworthy
School of Engineering and Applied Sciences
Harvard University
Cambridge, Massachusetts
luke_kenworthy@college.harvard.edu

***Abstract*—Post-training quantization is an established technique to reduce the size of weights and activations in neural networks and thereby accelerate CPU and hardware latency. We propose and explore quantile quantization (QQ), a novel method of post-training quantization. QQ is designed to be generalizable to any dataset, dividing data into quantiles and minimizing loss based on the data in those quantiles. We found that this approach outperformed other quantization methods in minimizing quantization error on toy data. When we used QQ on more complex neural networks (GoogLeNet, MobileNet, ResNet), we found that QQ performed worse than its competing quantization methods in terms of classification accuracy. Nonetheless, QQ offers an exciting foray into potential innovations on post-training quantizations, especially with the potential for additional optimizations for higher bit-width quantization.**

1. Introduction

Efforts to scale up deep neural networks (DNNs) have been hindered by the feasibility to deploy these networks into systems with limited resources and a more distributed set of devices. The post-training quantization of full-precision weights and activations to lower precision integers by diminishing the bit-width offers an effective methodology to enable efficient and accurate re-deployment of DNNs. Unlike alternative quantization methods such as quantization-aware training, it avoids retraining or the requirement to have access to the full training dataset. Additionally post-training quantization has compatibility with pre-existing models and can greatly reduce the size of a network while maintaining high accuracy.

Nonetheless, the two main methods available for post-training quantization have some notable disadvantages and limitations in their approaches. Uniform quantization (UQ) [1] linearly quantizes all floating point values to unsigned, eight-bit integers. UQ fails to give any extra granularity in cases where the distribution of values is dense at a particular point, instead giving those values the same amount of differentiation as those in more sparse regions. This is usually the case in the distribution of weights in a neural network, which skew heavily around zero. From this disadvantage, however, we learn that dense regions need more granular quantization, or else too much information will be lost.

Piecewise linear quantization (PWLQ) [2] tries to take advantage of this observation, but theoretically falls short. PWLQ uses a piecewise linear function consisting of two lines, allocating more of the quantized number line to more dense regions of the distribution. While this approach is a step in the right direction, it is a more hacky form of quantization where the 'high-density' and 'low-density' regions are set without considerable attention and ad hoc to the specific circumstances of the DNN. The selection of breaklines for the quantization isn't especially regimented in nature. Further, it does not generalize well to more complicated, multimodal distributions since it only designates a high-density and low-density region.

1.1. Problem

To overcome the shortcomings of existing post-training quantizations methods (UQ and PWLQ), we seek to devise a quantization scheme that is *efficient* and as *accurate* as possible that can successfully *generalize* for any distribution of weights alongside enabling a layer-wise fine-tuning mechanism to improve the overall accuracy of the quantized DNN. We introduce this specific approach as **quantile quantization (QQ)**. QQ gives the most quantized precision to the highest density regions. Using the empirical CDF of the data we are quantizing, we divide the distribution into K regions of equal mass $1/K$. We then divide the quantized number space equally among these regions, making the quantized numbers approximate a uniform distribution. QQ makes no a priori assumptions about the distribution of data, since it relies solely on the empirical CDF.

We evaluated the performance of our approach in comparison to the existing methods such as UQ and PWLQ utilizing accuracy as our chief evaluation metric across a range of proposed bit-widths. We found that QQ has nice theoretical properties and performed better than PWLQ and UQ in some cases when minimizing the quantization error on toy data. However, when we applied QQ in a layer-wise fashion to reduce the size of more complex neural networks,

we found that it was unable to compete with UQ or PWLQ in terms of classification accuracy. Regardless, we think this is a valuable technique because it lays the groundwork for further exploration with similar quantization techniques, and shows how a reduction in quantization error does not necessarily result in a reduction of classification error with neural networks.

2. Related Work

2.1. Uniform Quantization

Uniform quantization uses a linear quantization function as follows:

$$Q(r) = \textbf{Int}(r/S) - Z,$$

where Q is the quantization function, r is the real input, S is the scaling factor, Z is the integer zero point, and **Int** converts a floating point number to an integer through some method such as truncation, rounding, or stochastic rounding. To un-quantize quantized values, you can simply multiply by S and add Z. Because the rounding **Int** function does not save any information about the original number, there is information loss inherent in the quantization process. This loss is known as quantization error, and is the force behind a loss in accuracy that occurs in neural networks once they are quantized. Though quantization error is usually a good proxy for accuracy loss, the relationship between the two is not well understood. It is possible, in some cases, for the accuracy of a neural network to increase after quantization because the quantization serves as a regularization factor [1].

2.2. Piecewise Linear Quantization

PWLQ considers, without loss of generality, a real-valued floating point range of numbers from $-m$ to m where $m > 0$. This range is divided into two regions, a center region $R_1 = [-p, p]$ and a tail region $R_2 = [-m, p) \cup (p, m]$. The optimal value of p, p^* is then picked such that p minimizes the quantization error contributed by each region for the distribution. p^* is found, in practice, by assuming an underlying Gaussian or Laplacian distribution and then performing gradient descent [2].

2.3. Uniform Noise Quantization

In their quantization scheme called Uniq, Baskin et al use a multi-quantile quantization scheme similar to ours [3]. However, their system assumes that their data follow a normal distribution, where they try to estimate its mean and standard deviation as they quantize the values. From this normal distribution, the authors derive their quantization function, so this assumption is critical to their process. While QQ similarly uses a multi-quantile quantization system, we do not assume anything about the underlying values themselves because we use the empirical distribution of the data to create our bins. This method of quantization gives us better results on non-standard multimodal distributions.

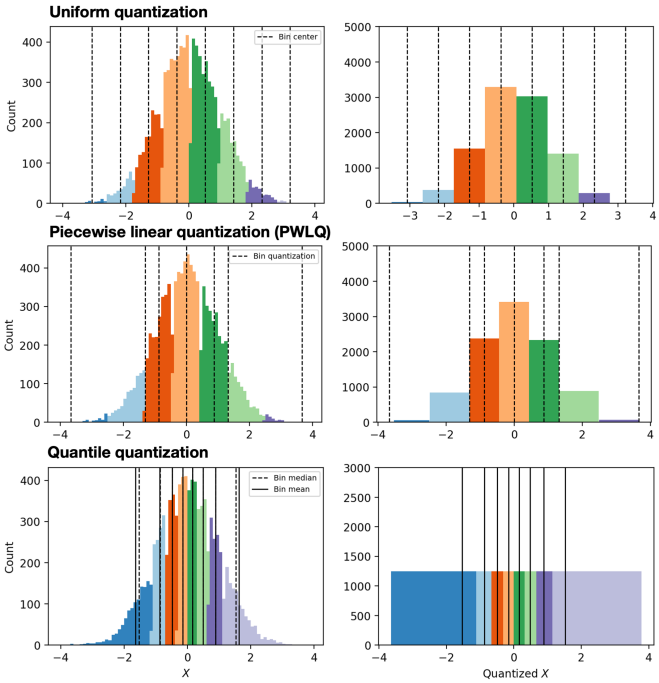
3. Proposed Approach

3.1. Quantile Quantization

As we mentioned earlier, QQ uses the empirical CDF of the data to divide the distribution into K regions of equal mass $1/K$. We then set the unquantized value that each quantile represents to the mean value in its respective bin, thus minimizing the L_2 loss of the data points in that region.

Using the universality of the Uniform principle (in which the CDF value of a random variable is uniformly distributed), this approach spreads the data equally over all of the quantiles, maximizing entropy as we see in Figure 1. While we use this quantization method for compressing neural network weights, it is fully generalizable and can work on any sample of data values since every sample has an empirical CDF. The empirical CDF serves as a consistent and unbiased estimator for the actual CDF of a distribution, and with millions of parameters per model, the estimation loss is infinitesimal.

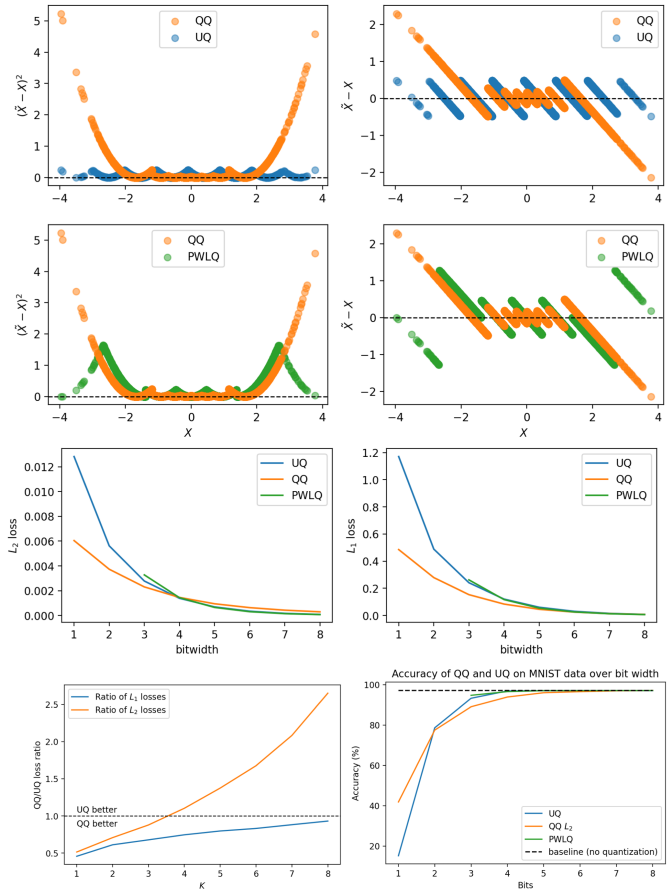
Figure 1. The three main quantization schemes discussed in this project.



We wanted to initially test the efficacy of QQ on toy data before we used it on a bigger dataset, like a neural network (see Figure 1). To do this, we compared the quantization losses of QQ, UQ, and PWLQ on Normally distributed weights. For every bit-width b (which allows 2^b quantization values), we evaluated the L_1 and L_2 quantization losses from each quantization scheme. In our results, shown in Figure 2, we found that QQ's supremacy depended on the type of loss we looked at. With L_2 loss, PWLQ outperformed UQ and QQ for $b > 4$. With L_1 loss, QQ dominated all

bit-widths, though its performance was nearly identical to the other methods for $b > 6$.

Figure 2. Quantization loss with UQ, PWLQ, and QQ. Top two pairs of plots compare the linear and quadratic quantization losses of QQ, UQ, and PWLQ. The third pair shows the the L_1 and L_2 losses with increasing bit-width. The bottom left plot compares the loss of UQ and QQ through their ratios, and the bottom right plot shows the effect of quantization on a small neural network trained on MNIST data.



4. Intellectual Points

QQ relies on the universality of the Uniform distribution, which ensures that the distribution of quantizations is uniform, thereby maximizes quantization entropy. This way the weights and activations will be spread equally over all quantizations, putting "equal pressure" on those values. Entropy increases from the UQ to PWLQ approach even with PWLQ centered around an optimal breakpoint; therefore, the results of QQ may help provide the foundations for a theoretical framework for quantization error and subsequent model accuracy.

We thought that the elegant theoretical properties of QQ and it's reliance on probability theory would yield the smallest quantization error and best accuracy. If these evaluation metrics can be improved, QQ can then serve as the primary state-of-art quantization method enabling the successful deployment of DNNs within resource-limited settings, which are typically a more realistic characterization of computational environments. Finally, while UQ and PWLQ utilize assumptions on the initial distributions following a Gaussian or Laplacian format, QQ utilizes the empirical CDF and does not make assumptions on the distribution of weights and activations. Our non-parametric approach also provides some simplicity to the problem.

5. Work Performed

We have now established our core theoretical framework for quantile quantization and considered its potential improvements upon existing post-training methodology in UQ and PWLQ. Next, we turn to our implementation of quantile quantization and explore the effects on accuracy for a more complex set of networks. For data, we utilized the ImageNet database of images to compare the performances of QQ, UQ, and PWLQ on well-established networks including ResNet, MobileNetV3, and GoogLeNet.

In testing our quantization approach against the current standards, we utilized Imagenette [4], a subset of the holistic ImageNet dataset that only contains images from 10 'easily-classified' categories. In light of the massive size of the full ImageNet dataset, these smaller vision datasets provide a more efficient methodology of training, which allows for a more rapid assessment of quantization performance.

To implement and subsequently run various standards of post-training quantization, we utilized PyTorch's *pretrained* models subpackage, which contains definitions for various model architectures for image classifications including our target neural networks ResNet, GoogLeNet, and MobileNetV3. These models were pretrained with ImageNet, hence the ImageNette-based testing. We then utilized our generalizable forms of quantization on the weights and biases of each network to compare between the methods. Our main result is the comparison of accuracy between various forms of quantization across different bit-widths to assess which approach realizes the best accuracy retention.

6. Results and Discussion

The results with the smaller MNIST-trained small neural network (bottom right plot in Figure 2) looked promising. QQ was better than UQ and PWLQ for very small bit-width. One could even notice a parallel between the MNIST-trained small neural network and L_2 loss (in the same Figure). Though, this is an extremely small network with few applications. Our main results concern the larger and more realistic networks.

As seen in Figures 3, 4, 5, and 6, QQ generally underperformed UQ and PWLQ in quantized model accuracy across a variety of neural networks. All of these models had similar distributions of weights, with a high density of weight values around zero. While the results were disappointing in that QQ did not compete favorably with the other forms of quantization, they give some insight into the interplay between quantization methods and neural networks.

First of all, UQ outperformed QQ in many cases. We hypothesize that part of this performance can be explained

Figure 3. ResNet quantized accuracy across bit-widths.

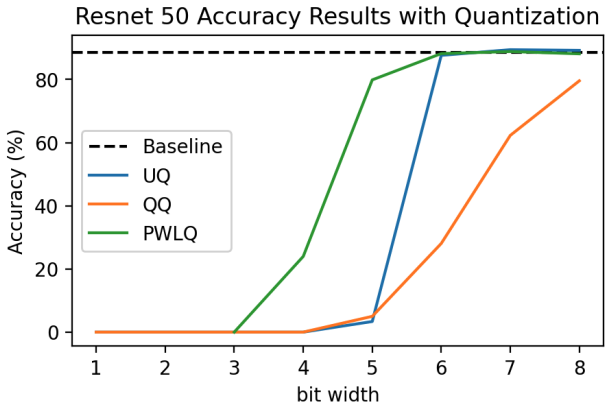


Figure 4. MobileNet V3 Small quantized accuracy across bit-widths.

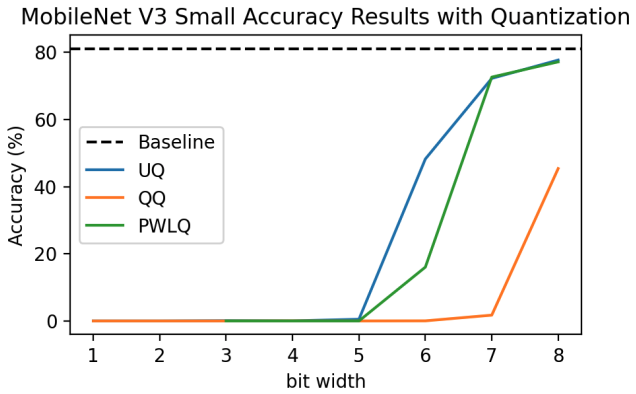


Figure 5. MobileNet V3 Large quantized accuracy across bit-widths.

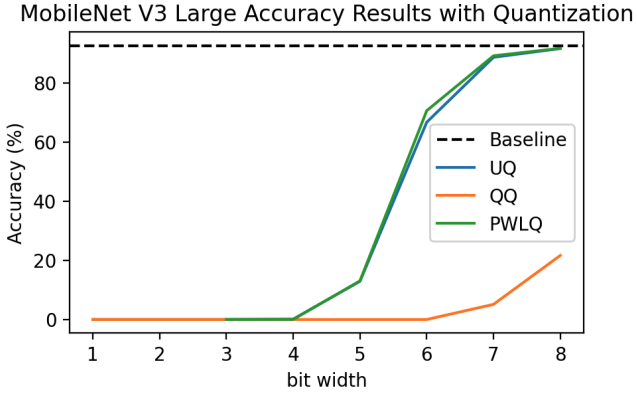
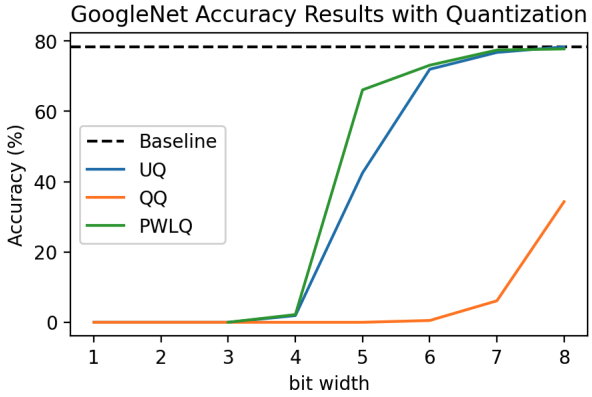


Figure 6. GoogLeNet quantized accuracy across bit-widths.



by the effect of outliers. During QQ, depending on the empirical distribution of data and the level of quantization, many of these outliers will get shrunk significantly, perhaps becoming nearly indistinguishable from weights that are closer to zero. With UQ, while there will be less differentiation among the very small weights, the outliers will still be represented as such in the data.

Second, PWLQ also did better than QQ in all of our trials. We believe that part of this has to do with the heavily optimized nature of the code used in Fang et al.'s paper [2]. We believed that one of the advantages of QQ was its simplicity, so we did not include such optimizations. We also hypothesize that PWLQ had the same advantage of UQ regarding outliers, but PWLQ also adds in another optimization by using a breakpoint, so it was able to outperform UQ in many cases as well. These results would be consistent with, for example, the phenomenon seen in Figure 5, where PWLQ and UQ had a similar level of performance, but PWLQ was slightly better.

These claims are supported by the top two pairs of plots Figure 2. QQ loss is much smaller than that of PWLQ and UQ at high-density parts of the weights distribution, while the loss skyrockets at the leftmost and rightmost bins — or at the tails of the distribution. This loss is not as large with PWLQ and UQ.

7. Conclusion

We believe that QQ is an exciting approach for quantization in neural networks. As seen in Figure 2, QQ was dominant in minimizing L_1 loss across all bit-widths. Though we did not get the results we had hoped for regarding quantized neural network accuracy, we believe that further modifications to QQ might be able to overcome some of the problems we encountered. We hope to explore more *deeply* where QQ does worse than its competitors, and how this could be improved by, again, optimizing QQ greedily. Perhaps we could dedicate a few of the quantiles on both ends of the distribution specifically for outliers, determining a cutoff similar to how PWLQ selects a breakpoint. While this would sacrifice some of the simplicity and generalizability of QQ, this tradeoff might be necessary to make QQ achieve high accuracy on neural networks. Our main contribution to the literature is the exploration of QQ on neural networks, and a brief exploration of the relationship between quantization loss and accuracy loss. This relationship was clearer for smaller networks but much more ambiguous for larger ones. Moreover, the entropy of the quantized distribution does not seem to be that important in determining the accuracy of a quantized neural network. Further work needs to be done to establish better predictors of accuracy than just quantization loss or entropy.

Acknowledgments

The authors would like to thank Professor HT "High Tech" Kung and all of the TFs in CS 242 for all of their guidance throughout this project, and for a great semester!

References

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

[2] Jun Fang, Ali Shafiee, Hamzah Abdel-Aziz, David Thorsley, Georgios Georgiadis, and Joseph H Hassoun. Post-training piecewise linear quantization for deep neural networks. In *European Conference on Computer Vision*, pages 69–86. Springer, 2020.

[3] Chaim Baskin, Eli Schwartz, Evgenii Zheltonozhskii, Natan Liss, Raja Giryes, Alex M Bronstein, and Avi Mendelson. Uniq: Uniform noise injection for non-uniform quantization of neural networks. *arXiv preprint arXiv:1804.10969*, 2018.

[4] fast.ai. Imagenette. https://github.com/fastai/imagenette, 2018.