UNIQ: Uniform Noise Injection for Non-Uniform Quantization of Neural Networks

CHAIM BASKIN and NATAN LISS, Technion
ELI SCHWARTZ, Tel Aviv University
EVGENII ZHELTONOZHSKII, Technion
RAJA GIRYES, Tel Aviv University
ALEX M. BRONSTEIN and AVI MENDELSON, Technion

We present a novel method for neural network quantization. Our method, named UNIQ, emulates a non-uniform k-quantile quantizer and adapts the model to perform well with quantized weights by injecting noise to the weights at training time. As a by-product of injecting noise to weights, we find that activations can also be quantized to as low as 8-bit with only a minor accuracy degradation. Our non-uniform quantization approach provides a novel alternative to the existing uniform quantization techniques for neural networks. We further propose a novel complexity metric of number of bit operations performed (BOPs), and we show that this metric has a linear relation with logic utilization and power. We suggest evaluating the trade-off of accuracy vs. complexity (BOPs). The proposed method, when evaluated on ResNet18/34/50 and MobileNet on ImageNet, outperforms the prior state of the art both in the low-complexity regime and the high accuracy regime. We demonstrate the practical applicability of this approach, by implementing our non-uniformly quantized CNN on FPGA.

CCS Concepts: • Computing methodologies → Neural networks; Supervised learning by classification; • Computer systems organization → Neural networks; • Mathematics of computing → Quantile regression; • Hardware → Hardware accelerators;

Additional Key Words and Phrases: Deep learning, neural networks, quantization, efficient deep learning

ACM Reference format:
Chaim Baskin, Natan Liss, Eli Schwartz, Evgenii Zheltonozhskii, Raja Giryes, Alex M. Bronstein, and Avi Mendelson. 2021. UNIQ: Uniform Noise Injection for Non-Uniform Quantization of Neural Networks. ACM Trans. Comput. Syst. 37, 1–4, Article 4 (March 2021), 15 pages.
https://doi.org/10.1145/3444943

1 INTRODUCTION

Deep neural networks (DNNs) are widely used in many fields today, including computer vision, signal processing, computational imaging, image processing, and speech and language processing.
Yet a major drawback of these deep learning models is their storage and computational cost. Typical deep networks comprise millions of parameters and require billions of multiply-accumulate (MAC) operations. In many cases, this cost renders them infeasible for running on mobile devices with limited resources. Although some applications allow moving the inference to the cloud, such an architecture still incurs significant bandwidth and latency limitations.

A recent trend in research focuses on developing lighter deep models, both in their memory footprint and computational complexity. The main focus of most of these works, including this one, is on alleviating complexity at inference time rather than simplifying training. Although training a deep model requires even more resources and time, it is usually done offline with large computational resources.

One way of reducing computational cost is quantization of weights and activations. Quantization of weights also reduces storage size and memory access. The bitwidths of activations and weights affect linearly the amount of required hardware logic; reducing both bitwidths by a factor of 2 reduces the amount of hardware by roughly a factor of 4. (A more accurate analysis of the effects of quantization is presented in Section 4.2). Quantization allows fitting bigger networks into a device, which is especially critical for embedded devices and custom hardware. However, activations are passed between layers, and thus, if different layers are processed separately, activation size reduction reduces communication overheads.

DNNs are usually trained and operate with both the weights and the activations represented in single-precision (32-bit) floating point. A straightforward uniform quantization of the pre-trained weights to 16-bit fixed-point representation has been shown to have a negligible effect on the model accuracy [8]. In the majority of the applications, further reduction of precision quickly degrades performance; hence, non-trivial techniques are required to carry it out.

### 1.1 Neural Networks on Custom Hardware

When implementing systems involving arbitrary precision, FPGAs and ASICs are a natural selection as target device due to their customizable nature. It was already shown that there is a lot of redundancy when using floating-point representation in neural networks. Therefore, custom low-precision integer representation can be used with little impact to the accuracy. Due to the steadily increasing on-chip memory size (tens of megabytes) and the integration of high bandwidth memory (hundreds of megabytes), it is feasible to fit all of the parameters inside an ASIC or FPGA when using low bitwidth. Besides the obvious advantage of reducing the latency, this approach has several advantages: power consumption reduction and smaller resource utilization, which in addition to DSP blocks and LUTs, also includes routing resource. The motivation of quantizing the activations is similar to that of the parameters. Although activations are not stored during inference, their quantization can lead to major saving in routing resources, which in turn can increase the maximal operational frequency of the fabric, resulting in increased throughput.

**Contribution.** Most previous quantization solutions for neural networks assume a uniform distribution of the weights, which is non-optimal because these rather have bell-shaped distributions [9]. To facilitate this deficiency, we propose a $k$-quantile quantization method with balanced (equal probability mass) bins, which is particularly suitable for neural networks, where outliers or long tails of the bell curve are less important. We also show a simple and efficient way of reformulating this quantizer using a “uniformization trick.”

In addition, we introduce a novel method for training a network that performs well with quantized values. This is achieved by adding noise (at training time) to emulate the quantization noise introduced at inference time. The uniformization trick renders exactly the injection of uniform noise and alleviates the need to draw the noise from complicated and bin-dependent distributions.
Although we limit our attention to the $k$-quantile quantization, the proposed scheme can work with any threshold configuration while still keeping the advantage of uniformly distributed noise in every bin.

Since the calculations of low-precision networks are composed of integer operations, their computation complexity cannot be measured by FLOPS. Therefore, we propose a novel metric for complexity of fixed-point quantized neural network models, called bit operations (BOPs). We show that this metric has a linear relation with logic utilization and power. Measuring the complexity of each method by BOPs, our UNIQ strategy compares favorably, in the accuracy vs. complexity trade-off, to other leading methods such as Distillation [21] and MLQ [31]. In addition, our method performs well on smaller models targeting mobile devices, represented in the work by MobileNetV1. Those networks are known to be harder to quantize due to lower redundancy in parameters.

Finally, we suggest a look-up table (LUT) approach for hardware implementation of non-uniformly quantized CNNs. We implement it for an FPGA and show it to be more power efficient and with higher throughput compared to the usual low-bit integers arithmetic.

2 RELATED WORK

Quantization methods. Previous studies have investigated quantizing the network weights and the activations to as low as 1- or 2-bit representation [6, 14, 22, 26, 33]. Such extreme reduction of the range of parameter values greatly affects accuracy. Recent works proposed to use a wider network, such as one with more filters per layer, to mitigate the accuracy loss [25, 35]. In some approaches, such as those of Zhu et al. [35] and Zhou et al. [32], a learnable linear scaling layer is added after each quantized layer to improve expressiveness.

A general approach to learning a quantized model adopted in recent works [3, 13, 14, 26, 33] is to perform the forward pass using the quantized values while keeping another set of full-precision values for the backward pass updates. In this case, the backward pass is still differentiable, whereas the forward pass is quantized. In the aforementioned works, a deterministic or stochastic function is used at training for weight and activation quantization. Another approach introduced by Mishra and Marr [21] and Polino et al. [25] is based on a teacher-student setup for knowledge distillation of a full-precision (and usually larger) teacher model to a quantized student model. This method allows training highly accurate quantized models but requires training of an additional, larger network.

Most previous studies have used uniform quantization (i.e., all quantization bins are equally sized), which is attractive due to its simplicity. However, unless the values are uniformly distributed, uniform quantization is not optimal.

Unfortunately, neural network weights are not uniform but rather have bell-shaped distributions [1, 9] as we also show in the supplementary material.

Non-uniform quantization is utilized by Han et al. [9], where the authors replace the weight values with indexes pointing to a finite codebook of shared values. Ullrich et al. [30] propose to use clustering of weights as a way of quantization. They fit a Gaussian prior model to the weights and use the cluster centroids as the quantization codebook. Xu et al. [31] also use such an approach but in addition embody incremental quantization both on network and layer levels. Other examples of methods that take data distribution into account are Bayesian quantization [20] and value-aware quantization [24]. In the work of Louizos et al. [20], sparsity priors are imposed on the network weights, providing information as to the bits to be allocated per each layer. In the work of Molchanov et al. [23], variational dropout, which also relies on the Bayesian framework, is proposed to prune weights in the network. Park et al. [24] propose to leave a small part of weights and activations (with higher values) in full precision while using uniform quantization for the rest.
Another approach adopted by Zhou et al. [32] learns the quantizer thresholds by iteratively grouping close values and re-training the weights. Lee et al. [18] utilize a logarithmic scale quantization for an approximation of the $\ell_2$-optimal Lloyd quantizer [19]. Cai et al. [3] proposed to optimize expectation of $\ell_2$ error of quantization function to reduce the quantization error. Normally distributed weights and half-normally distributed activations were assumed, which enables using a pre-calculated $k$-means quantizer. In the work of Zhou et al. [34], balanced bins are used, so each bin has the same number of samples. In some sense, this idea is the closest to our approach; yet although Zhou et al. [34] force the values to have an approximately uniform distribution, we pose no such constraint. In addition, since calculating percentiles is expensive in this setting, Zhou et al. [34] estimate them with means, whereas our method allows using the actual percentiles as detailed in the sequel.

Hardware architectures for efficient DNN implementation. Among the quantization methods proposed for reducing the DNN complexity, there is a line of works that aim to design efficient hardware architectures by leveraging mixed precision representation. Chen et al. [5] proposed a CNN accelerator that optimizes the energy efficiency of the end-to-end processing pipeline: the main contribution was a dataflow processor that optimizes energy efficiency by reusing data locally to reduce expensive data movement such as DRAM accesses. Judd et al. [15] rely on bit-serial compute units and on the parallelism that is naturally present within CNNs to improve performance and energy with no accuracy loss. Sharma et al. [28] propose dynamic bit-level fusion/decomposition that constitutes an array of bit-level processing elements that dynamically fuse to match the bitwidth of individual DNN layers. Such flexible architectures enable minimizing the computation and the communication with no apparent loss in accuracy.

3 NON-UNIFORM QUANTIZATION BY UNIFORM NOISE INJECTION

To present UNIQ, our uniform noise injection quantization method for training a neural network amenable to operation in low-precision arithmetic, we start by outlining several common quantization schemes and discussing their suitability for DNNs. Then, we suggest a training procedure where during training time uniform random additive noise is injected into weights simulating the quantization error. The scheme aims at improving the quantized network performance at inference time, when regular deterministic quantization is used.

3.1 Quantization

Let $X$ be a random variable drawn from some distribution described by the probability density function $f_X$. Let $T = \{t_i\}$ with $t_0 = -\infty$, $t_k = \infty$ and $t_{i-1} < t_i$ be a set of thresholds partitioning the real line into $k$ disjoint intervals (bins) $[t_{i-1}, t_i]$, and let $Q = (q_i)^k_{i=1}$ be a set of $k$ representation levels. A quantizer $Q_{T,Q}$ is a function mapping each bin $[t_{i-1}, t_i]$ to the corresponding representation level $q_i$. We denote the quantization error by $E = X - Q_{T,Q}(X)$. The effect of quantization can be modeled as the addition of random noise to $X$; the noise added to the $i$-th bin admits the conditional distribution $(X - q_i)|X \in [t_{i-1}, t_i]$.

Most works on neural network quantization focus on the uniform quantizer, which has a constant bin width $t_i - t_{i-1} = \Delta$ and $q_i = (t_{i-1} + t_i)/2$, and it is known to be optimal (in the sense of the mean squared error $\mathbb{E}E^2$, where the expectation is taken with respect to the density $f_X$) in the case of uniform distribution. Yet since $X$ in neural networks is not uniform but rather bell shaped [9], in the general case the optimal choice, in the $\ell_2$ sense, is the $k$-means quantizer. Its name follows the property that each representation level $q_i$ coincides with the $i$-th bin centroid (mean w.r.t. $f_X$). Although finding the optimal $k$-means quantizer is known to be an NP-hard problem, heuristic procedures such as the Lloyd-Max algorithm [19] usually produce a good approximation. The $k$-means quantizer coincides with the uniform quantizer when $X$ is uniformly distributed.
Although being a popular choice in signal processing, the $k$-means quantizer encounters severe obstacles in our problem of neural network quantization. First, the Lloyd-Max algorithm has a prohibitively high complexity to be used in every forward pass. Second, it is not easily amenable to our scheme of modeling quantization as the addition of random noise, as the noise distribution at every bin is complex and varies with the change of the quantization thresholds. Finally, our experiments shown in Section 4.3 in the sequel suggest that the use of the $\ell_2$ criterion for quantization of deep neural classifier does not produce the best classification results. The weights in such networks typically assume a bell-shaped distribution with tails exerting a great effect on the mean squared error, yet having little impact on the classification accuracy.

Based on empirical observations, we conjecture that the distribution tails, which $k$-means is very sensitive to, are not essential for good model performance, at least in classification tasks. As an alternative to $k$-means, we propose the $k$-quantile quantizer characterized by the equiprobable bins property—that is, $\mathbb{P}(X \in [t_{i-1}, t_i]) = 1/k$. The property is realized by setting $t_i = F_X^{-1}(i/k)$, where $F_X$ denotes the cumulative distribution function of $X$, and, accordingly, its inverse $F_X^{-1}$ denotes the quantile function. The representation level of the $i$-th bin is set to the bin median, $q_i = \text{med}\{X|X \in [t_{i-1}, t_i]\}$. It can be shown that in the case of a uniform $X$, the $k$-quantile quantizer coincides with the $k$-level uniform quantizer.

The cumulative distribution $F_X$ and the quantile function $F_X^{-1}$ can be estimated empirically from the distribution of weights and updated in every forward pass. Alternatively, one can rely on the empirical observation that the $\ell_2$-regularized weights of each layer tend to follow an approximately normal distribution [2]. To confirm that this is the case for the networks used in the article, we analyzed the distribution of weights. An example of a layer-wise distribution of weights is shown in the supplementary material. Relying on this observation, we can estimate $\mu$ and $\sigma$ per each layer and use the CDF of the normal distribution (and its inverse, the normal quantile function).

Using the fact that applying a distribution function $F_X$ of $X$ to itself results in uniform distribution allows an alternative construction of the $k$-quantile quantizer. We apply the transformation $U = F_X(X)$ to the input converting it into a uniform random variable on the interval $[0, 1]$. Then, a uniform $k$-level quantizer (coinciding with the $k$-quantile quantizer) is applied to $U$ producing $\hat{U} = Q_{\text{uni}}(U)$; the result is transformed back into $\hat{X} = F_X^{-1}(\hat{U})$ using the inverse transformation. We refer to this procedure as the uniformization trick. Its importance will become evident in the next section.

### 3.2 Training Quantized Neural Networks by Uniform Noise Injection

The lack of continuity, let alone smoothness, of the quantization operator renders impractical its use in the backward pass. As an alternative, at training we replace the quantizer by the injection of random additive noise. This scheme suggests that instead of using the quantized value $\hat{w} = Q_{T,Q}(w)$ of a weight $w$ in the forward pass, $\hat{w} = w + \epsilon$ is used with $\epsilon$ drawn from the conditional distribution of $(W - q_i)|W \in [t_{i-1}, t_i]$ described by the density

$$f_\epsilon(\epsilon) = \frac{f_W(\epsilon + q_i)}{\int_{t_{i-1}}^{t_i} f_W(w)dw} \quad (1)$$

defined for $\epsilon \in [t_{i-1} - q_i, t_i - q_i]$ and vanishing elsewhere. The bin $i$ to which $w$ belongs is established according to its value and is fixed during the backward pass. Quantization of the network activations is performed in the same manner.

The fact that the parameters do not directly undergo quantization keeps the model differentiable. In addition, gradient updates in the backward pass have an immediate impact on the forward pass,
in contrast to the directly quantized model, where small updates often leave the parameter in the same bin, leaving it effectively unchanged.

Although it is customary to model the quantization error as noise with uniform distribution \( f_E \), this approximation breaks in the extremely low-precision regimes (small number of quantization levels \( k \)) considered here. Hence, the injected noise has to be drawn from a potentially non-uniform distribution that furthermore changes as the network parameters and the quantizer thresholds are updated.

To overcome this difficulty, we resort to the uniformization trick outlined in the previous section. Instead of the \( k \)-quantile quantizer \( w' = Q(w) \), we apply the equivalent uniform quantizer to the uniformized variable, \( \hat{w} = F_W^{-1}(Q_{\text{uni}}(F_W(w))) \). The effect of the quantizer can be again modeled using noise injection, \( \hat{w} = F_W^{-1}(F_W(w) + e) \), with the cardinal difference that now the noise \( e \) is uniformly distributed on the interval \( [-\frac{1}{2k}, \frac{1}{2k}] \) (estimating quantization error distribution).

Usually, quantization of neural networks is either used for training a model from scratch or applied post-training as a fine-tuning stage. Our method, as will be demonstrated in our experiments, works well in both cases. Our practice shows that best results are obtained when the learning rate is reduced as the noise is added; we explain this by the need to compensate for noisier gradients.

### 3.3 Gradual Quantization

The described method works well “as is” for small- to medium-sized neural networks. For deeper networks, the basic method does not perform as well, most likely due to errors arising when applying a long sequence of operations where more noise is added at each step. We found that applying the scheme gradually to small groups of layers works better in deep networks. To perform gradual quantization, we split the network into \( N \) blocks \( \{B_1, \ldots, B_N\} \), each containing about the same number of consecutive layers. We also split our budget of training epochs into \( N \) stages. At the \( i \)-th stage, we quantize and freeze the parameters in blocks \( \{B_1, \ldots, B_{i-1}\} \) and inject noise into the parameters of \( B_i \). For the rest of the blocks \( \{B_{i+1}, \ldots, B_N\} \) neither noise nor quantization is applied.

This approach is similar to one proposed by Xu et al. \[31\]. This way, the number of parameters into which the noise is injected simultaneously is reduced, which allows better convergence. The deeper blocks gradually adapt to the quantization error of previous ones and thus tend to converge relatively fast when the noise is injected into them. For fine-tuning a pre-trained model, we use the same scheme, applying a single epoch per stage. An empirical analysis of the effect of the different number of stages is presented in the supplementary material. This process can be performed iteratively, restarting from the beginning after the last layer has been trained. Since this allows earlier blocks to adapt to the changed values of the following ones, the iterative process yields an additional increase in accuracy. Two iterations were performed in the reported experiments.

### 3.4 Activations Quantization

Activations quantization is also beneficial in lowering the arithmetic complexity and can help in decreasing the communication overhead in the distributed model case. We observed that a by-product of training with noisy weights is that the model becomes less sensitive to a certain level of quantization of the activations. In training time, like in the case of the weights, we apply the CDF \( (F_A(a)) \) to uniformize the activations, but we do not apply the inverse CDF and treat the uniformization as part of the non-linear activation function. The reason for dropping the inverse CDF is that we observed similar accuracy performance with and without it. We observed that relying solely on the CDF as non-linearity cause accuracy performance degradation, so we apply both CDF and ReLU. In inference time, we again apply the CDF followed by uniform quantization.
We use the same quantizer for activations as used for weights with the slight modification that the first bin captures all of the negative values \([-\infty, 0]\) and assign then the value 0 (the effect of ReLU activation).

4 EXPERIMENTAL EVALUATION

We performed an extensive performance analysis of the UNIQ scheme compared to the current state-of-the-art methods for neural network quantization. The basis for comparison is the accuracy vs. the total number of BOPs in visual classification tasks on the ImageNet-1K [27] dataset. The CIFAR-10 dataset [16] is used to evaluate different design choices made, whereas the main evaluation is performed on ImageNet-1K.

MobileNet [12] and ResNet-18/34/50 [10] architectures are used as the baseline for quantization. MobileNet is chosen as a representative of lighter models, which are more suited for a limited hardware setting where quantization is also most likely to be used. ResNet-18/34/50 is chosen due to its near state-of-the-art performance and popularity, which makes it an excellent reference for comparison.

We adopted the number of BOPs metric to quantify the network arithmetic complexity. This metric is particularly informative about the performance of mixed-precision integer arithmetic especially in hardware implementations on FPGAs and ASICs.

Training details. For quantizing a pre-trained model, we train with SGD for a number of epochs equal to the number of trainable layers (convolution and fully connected). We also follow the gradual process described earlier with the number of stages set to the number of trainable layers. The learning rate is $10^{-4}$, momentum 0.9, and weight decay $10^{-4}$. In all experiments, unless otherwise stated, we fine-tuned a pre-trained model taken from an open-sourced repository of pre-trained models.

4.1 Performance of Quantized Networks on ImageNet

Table 1 compares the ResNet and MobileNet performance with weights and activations quantized to several levels using UNIQ and other leading approaches reported in the literature. For baseline, we use a full-precision model with 32-bit weights and activations. Note that the common practice of not quantizing first and last layers significantly increases the network complexity in BOPs. In contrast, our model performs quite well even when we quantize all of the layers and thus achieves lower-complexity requirements with a higher bitwidth. Note also the diminishing impact of the weights bitwidth on the BOP complexity as explained hereafter.

We found UNIQ to perform well also with the smaller MobileNet architecture. This is in contrast to most of the previous methods that resort to larger models and doubling the number of filters [25], thus quadrupling the number of parameters (e.g., from 11 to 44 million for ResNet-18).

4.2 Accuracy vs. Complexity Trade-off

Since custom precision data types are used for the network weights and activations, the number of MAC operations is not an appropriate metric to describe the computational complexity of the model. Therefore, we use the BOPs metric quantifying the number of BOPs. Given the bitwidth of two operands, it is possible to approximate the number of BOPs required for basic arithmetic operations such as addition and multiplication. The proposed metric is useful when the inference is performed with fixed-point operations on custom hardware like FPGAs or ASICs. Both are a

1https://github.com/Cadene/pretrained-models.pytorch.
| Architecture   | Method   | Bits (w,a) | Model Size (Mbit) | Complexity (GBOPs) | Accuracy (% Top-1) |
|----------------|----------|------------|-------------------|--------------------|-------------------|
| MobileNet      | UNIQ     | 4,8        | 16.8              | 25.1               | 66.00             |
| MobileNet      | UNIQ     | 5,8        | 20.8              | 30.5               | 67.50             |
| MobileNet      | QSM [29] | 8⁺,8⁺      | 33.6              | 46.7               | 68.01             |
| MobileNet      | UNIQ     | 8,8        | 33.6              | 46.7               | 68.25             |
| MobileNet      | Baseline | 32,32      | 135.2             | 626                | 68.20             |
| ResNet-18      | UNIQ     | 4,8        | 46.4              | 81.5               | 67.02             |
| ResNet-18      | Apprentice [21] | 2⁺,8⁺   | 39.2              | 83                 | 67.6              |
| ResNet-18      | UNIQ     | 5,8        | 58.4              | 99.5               | 68.00             |
| ResNet-18      | Apprentice [21] | 4⁺,8⁺   | 61.4              | 114                | 70.40             |
| ResNet-18      | UNIQ     | 4⁺,8⁺      | 61.4              | 114                | 69.12             |
| ResNet-18      | UNIQ     | 5⁺,8⁺      | 72.6              | 130                | 69.5              |
| ResNet-18      | Apprentice [21] | 2⁺,32   | 39.2              | 256                | 68.50             |
| ResNet-18      | IQN [32] | 5⁺,32      | 58.4              | 315                | 68.89             |
| ResNet-18      | MLQ [31] | 5⁺,32      | 58.4              | 315                | 69.09             |
| ResNet-18      | Distillation [25] | 4⁺,32   | 61.6              | 367                | 64.20             |
| ResNet-18      | Baseline | 32,32      | 374.4             | 1,924              | 69.60             |
| ResNet-34      | Apprentice [21] | 2⁺,8⁺   | 59.3              | 135                | 71.5              |
| ResNet-34      | UNIQ     | 4,8        | 87.2              | 166                | 71.09             |
| ResNet-34      | Apprentice [21] | 4⁺,8⁺   | 101.8             | 198                | 73.1              |
| ResNet-34      | UNIQ     | 4⁺,8⁺      | 101.8             | 198                | 72.19             |
| ResNet-34      | UNIQ     | 5⁺,8⁺      | 123.1             | 234                | 73.22             |
| ResNet-34      | Apprentice [21] | 2⁺,32   | 59.3              | 396                | 72.8              |
| ResNet-34      | UNIQ     | 4,32       | 101.8             | 516                | 73.1              |
| ResNet-34      | Baseline | 32,32      | 698               | 3,903              | 73.4              |
| ResNet-50      | Apprentice [21] | 2⁺,8⁺   | 112.9             | 113                | 72.8              |
| ResNet-50      | UNIQ     | 4,8        | 102.4             | 134                | 74.37             |
| ResNet-50      | UNIQ     | 4⁺,8⁺      | 102.4             | 167                | 75.1              |
| ResNet-50      | Apprentice [21] | 4⁺,8⁺   | 160               | 167                | 74.7              |
| ResNet-50      | Apprentice [21] | 2⁺,32   | 112.8             | 344                | 74.7              |
| ResNet-50      | UNIQ     | 4,32       | 102.4             | 423                | 75.09             |
| ResNet-50      | Baseline | 32,32      | 819.5             | 3,232              | 76.02             |

Complexity is reported in number of bit operations as explained in the text. Number of bits is reported as (weights, activations); (weights⁺, activations⁺) indicates that the first and last layers are not quantized (full precision). Model size is calculated as the sum of parameters sizes in bits. Accuracy is top-1 accuracy on ImageNet. For each DNN architecture, rows are sorted in increasing order complexity.

A natural choice for quantized networks, due to the use of LUTs and dedicated MAC (or more general DSP) units, which are efficient with custom data types.

An important phenomenon that can be observed in Table 1 is the non-linear relation between the number of activation and weight bits and the resulting network complexity in BOPs. To quantify this effect, let us consider a single convolutional layer with $b_w$-bit weights and $b_a$-bit activations containing $n$ input channels, $m$ output channels, and $k \times k$ filters. The maximum value of a single output is about $2^{b_w+b_a}nk^2$, which sets the accumulator width in the MAC operations.
Fig. 1. Performance vs. complexity of different quantized neural networks. Performance is measured as top-1 accuracy on ImageNet; complexity is estimated in number of BOPs. Network architectures are denoted by different marker shapes; quantization methods are marked in different colors. Multiple instances of the same method (color) and architecture (shape) represent different levels of quantization. Note that our method performs better for ResNet-50, comparable for ResNet-34, and worse for ResNet-18; this is due to the fact that in Apprentice [21], the baselines (full-precision) models are better for smaller models. Our model also performs well for the smaller and more efficient MobileNet architecture, which is known to be a harder problem.

The complexity of a single output calculation consists therefore of \( nk^2 b_a \)-wide \( \times b_w \)-wide multiplications and about the same amount of \( b_o \)-wide additions. This yields the total layer complexity of

\[
\text{BOPs} \approx mnk^2( b_a b_w + b_a + b_w + \log_2 nk^2 ).
\]

Note that the reduction of the weight and activation bitwidth decreases the number of BOPs as long as the factor \( b_a b_w \) dominates the factor \( \log_2 nk^2 \). Since the latter factor depends only on the layer topology, this point of diminishing return is network architecture dependent.

Another factor that must be incorporated into the BOPs calculation is the cost of fetching the parameters from an external memory. Three assumptions are made in the approximation of this cost: first, we assume that on-chip memory is used where memory access time is comparable to arithmetic operations; second, we assume that each parameter is only fetched once from an external memory; and third, the cost of fetching a \( b \)-bit parameter is assumed to be \( b \) BOPs. Given a neural network with \( n \) parameters all represented in \( b \) bits, the memory access cost is simply \( nb \).

Figure 1 and Table 1 display the performance-complexity trade-offs of various neural networks trained using UNIQ and other methods to different levels of weight and activation quantization. Notice that since we quantize the first and last layers, our ResNet-34 network has better computational complexity and better accuracy compared to all competing ResNet-18 networks. The same holds for our ResNet-50 compared to all competing ResNet-34.
### 4.3 Ablation Study

**Accuracy vs. quantization level.** We tested the effect of training ResNet-18 on CIFAR-10 with UNIQ for various levels of weight and activation quantization. Table 2 reports the results. We observed that for such a small dataset, the quantization of activations and weights helps avoid overfitting and the results for quantized model come very close to those with full precision.

**Comparison of different quantizers.** In the following experiment, different quantizers were compared within the uniform noise injection scheme.

The bins of the *uniform quantizer* were allocated evenly in the range $[-3\sigma, 3\sigma]$, with $\sigma$ denoting the standard deviation of the parameters. For both the $k$-quantile and the $k$-means quantizers, normal distribution of the weights was assumed and the normal cumulative distribution and quantile functions were used for the uniformization and deuniformization of the quantized parameter. The $k$-means and uniform quantizers used a pre-calculated set of thresholds translated to the uniformized domain. Since the resulting bins in the uniformized domain had different widths, the level of noise was different in each bin. This required an additional step of finding the bin index for each parameter, approximately doubling the training time.

The three quantizers were evaluated in a ResNet-18 network trained on the CIFAR-10 dataset with weights quantized to 3 bits ($k = 8$) and activations computed in full precision (32 bit). Table 3 reports the obtained top-1 accuracy. $k$-Quantile quantization outperforms other quantization methods and is only slightly inferior to the full-precision baseline. In terms of training time, the $k$-quantile quantizer requires about 60% more time to train for $k = 8$; this is compared to around a 280% increase in training time required for the $k$-means quantizer. In addition, $k$-quantile training time is independent on the number of quantization bins as the noise distribution is same for every bin while the other methods require separate processing of each bin, increasing the training time for higher bitwidths.

**Training from scratch vs. fine-tuning.** Both training from scratch (i.e., from random initialization) and fine-tuning have their advantages and disadvantages. Training from scratch takes more time but requires a single training phase with no extra training epochs, at the end of which a quantized model is obtained. Fine-tuning, however, is useful when a pre-trained full-precision model is already available; it can then be quantized with a short re-training.

Table 4 compares the accuracy achieved in the two regimes on a narrow version of ResNet-18 trained on CIFAR-10 and 100. A 5-bit quantization of weights only and 5-bit quantization of both weights and activations were compared. We found that both regimes work equally well, reaching accuracy close to the full-precision baseline.

**Accuracy vs. number of quantization stages.** We found that injecting noise to all layers simultaneously does not perform well for deeper networks. As described in Section 3.3, we suggest splitting the training into $N$ stages such that at each stage the noise is injected only into a subset of layers.

To determine the optimal number of stages, we fine-tuned ResNet-18 on CIFAR-10 with a fixed 18-epoch budget. Bitwidth was set to 4 for both the weights and the activations.
Table 4. Top-1 Accuracy (in Percentage) on CIFAR-10 and 100 of a Narrow Version on ResNet-18 Trained with UNIQ from Random Initialization vs. Fine-Tuning a Full-Precision Model

| Dataset  | Bits  | Full Training | Fine-Tuning | Baseline |
|----------|-------|---------------|-------------|----------|
| CIFAR-10 | 5,32  | 93.8          | 90.9        | 92.0     |
| CIFAR-10 | 5,5   | 91.56         | 91.21       |          |
| CIFAR-100| 5,32  | 66.54         | 65.73       | 66.3     |
| CIFAR-100| 5,5   | 65.29         | 65.05       |          |

Number of bits is reported as (weights, activations).

Figure 2 reports the classification accuracy as a function of the number of quantization stages. Based on these results, we conclude that the best strategy is injecting noise to a single layer at each stage. We follow this strategy in all experiments conducted in this article.

5 HARDWARE IMPLEMENTATION

5.1 MAC Operation for Non-Uniformly Quantized Data

From hardware implementation perspective, uniform quantization has low arithmetic complexity since the quantization is homomorphic and thus integer multiplication and addition can be used. However, non-uniform quantization is not homomorphic (e.g., the product of elements mapped to 2 and 3 is not equal to element mapped to 6), and thus it is impossible to use integer operations anymore. A simple but rather expensive approach would be to use floating-point operations. To still benefit from advantages of low-precision integer calculations along with non-uniform quantization, we suggest using the LUT approach: instead of calculating the product of two values during the inference, we pre-compute the result and store it in a table. Thus, any arithmetic operation requires a single memory access, at the expense of storing a multiplication table.

In the inference phase, each layer will have its own $2^{BW_w}$ discrete values for weights, represented in floating-point format. Each of these discrete levels is associated with an integer index. For activations, we use the training set statistics of a specific layer to pre-compute the thresholds and their mapping to an integer index representation. Let $BW_w$, $BW_a$, and $BW_p$ be number of bits for weights, activation, and fixed-point representation of their product, respectively. For each layer,
we construct a LUT with $2^{BW_w + BW_a}$ entries. Let $W_q$ and $A_q$ be two sets of discrete floating-point values of weights and activations, respectively. Let us also denote the outer product operator as $\otimes$. Table entries are $BW_p$ bits wide, and the table is calculated using MinMax scaling as follows:

$$Q(W_q \otimes A_q) = \left\lfloor \frac{W_q \otimes A_q}{\max(W_q \otimes A_q) - \min(W_q \otimes A_q)} (2^{BW_p} - 1) \right\rfloor,$$

where $\lfloor \cdot \rfloor$ denotes the rounding operation. Since the entries in the LUT are represented in fixed-point representation, they can be summed up like integers to form the MAC result before the activation function and quantization are applied (by using pre-computed thresholds and their mapping to integers indexes).

An illustration of this function, for activation with $\mu = 900$ and $\sigma = 900$, is presented in Figure 3. From this, we can clearly see that the distances on the y-axis are equal, which implies equiprobable bins. The intersection of the vertical green lines with the x-axis represents the integer thresholds that are loaded to the custom device, during inference, and define the boundaries between each bin.

5.2 Logic Utilization and Power Consumption on FPGA

It is not reasonable to implement popular architectures such as ResNet and MobileNet in a dataflow manner where all of the layers are implemented on a single FPGA. The more common and scalable approach is to implement a generic MAC calculation logic that can support multiple types of layers and multiple sizes. This MAC calculator is iteratively configured and called according to the current layer. We perform static power and logic utilization analysis for a single $256 \times 256 \times 3 \times 3$ representative convolution layer. We have implemented the proposed MAC calculation logic for both LUT-based (for non-uniform quantization) and DSP-based (for uniform quantization) approaches, on Intel’s Arria 10 FPGA. Table 5 reports logic utilization and power consumption, as reported by the Quartus Prime software for Intel FPGA synthesis and place & route. LUT-based realization outperforms the DSP-based one both in terms of maximal frequency (up to 60% higher). As for power consumption for the fixed frequency, the LUT-based solution shows an advantage for low
Table 5. Uniform (DSPs) and Non-Uniform (LUTs) Logic Utilization and Power Consumption on Intel’s Arria 10 FPGA

| Bits (w,a) | Method | LUTs   | DSPs | BRAM (Kbits) | Frequency (MHz) | Power (mW) |
|-----------|--------|--------|------|--------------|----------------|------------|
| (4,4)     | LUTs   | 13,353 | 0    | 2,236        | 240 (max)      | 7,133      |
|           |        |        |      |              | 150 (fixed)    | 6,297      |
|           | DSPs   | 15,781 | 1,152|              | 185 (max)      | 7,039      |
|           |        |        |      |              | 150 (fixed)    | 6,900      |
| (5,5)     | LUTs   | 13,353 | 0    | 8,945        | 240 (max)      | 7,145      |
|           |        |        |      |              | 150 (fixed)    | 6,304      |
|           | DSPs   | 18,086 | 1,152|              | 185 (max)      | 7,723      |
|           |        |        |      |              | 150 (fixed)    | 7,221      |
| (6,6)     | LUTs   | 13,353 | 0    | 35,782       | 240 (max)      | 11,369     |
|           |        |        |      |              | 150 (fixed)    | 8,905      |
|           | DSPs   | 21,016 | 1,152|              | 185 (max)      | 8,001      |
|           |        |        |      |              | 150 (fixed)    | 7,539      |

Fig. 4. Logic utilization and power consumption vs. number of BOPs. A significant correlation between the proposed BOPs metric and both logic utilization and power consumption supports the hypothesis that BOPs is a good proxy to both. Note that LUTs here refers to the basic building block of FPGA rather than our implementation of multiplication.

bitwidth. However, since the LUT size grows exponentially with the number of bits, the approach is less efficient for higher bitwidths and the power consumption growth significantly.

To justify the use of the BOPs metric, we investigate the correlation among BOPs, logic utilization, and power consumption. From Figure 4, we can observe a linear relation between our computational complexity metric and logic utilization and power consumption. This allows a more accurate system design, based on computational complexity vs. logic utilization and power trade-off.
6 CONCLUSION

We introduced UNIQ—a training scheme for quantized neural networks. The scheme is based on the uniformization of the distribution of the (intended to be quantized) parameters, injection of additive uniform noise, followed by the de-uniformization of the obtained result. The scheme is amenable to efficient training by backpropagation in full-precision arithmetic and achieves maximum efficiency with the $k$-quantile (balanced) quantizer that was investigated in this article.

To set a common basis for comparison, we proposed a novel measure of complexity called bit operations (BOPs). We have shown that this metric has a quasi-linear relation with logic utilization and power, which facilitates better accuracy-complexity trade-offs. We demonstrate that UNIQ achieves improved results for quantized neural networks on ImageNet. Our solution outperforms any other quantized network in terms of accuracy vs. complexity for MobileNetV1 and ResNet-50 architectures, and achieves comparable results to the current SOTA method [21] for ResNet-34.

In addition, we implemented our non-uniformly quantized network on FPGA and demonstrate that its power consumption is similar to that of a uniformly quantized network operating with integer arithmetics.

Although this work considered a setting in which all parameters have the same bitwidth, more complicated bit allocations will be explored in following studies. The proposed scheme is not restricted to the $k$-quantile quantizer discussed in this article but rather applies to any quantizer. In the general case, the noise injected into each bin is uniform, but its variance changes with the bin width.

REFERENCES

[1] Alexander G. Anderson and Cory P. Berg. 2018. The high-dimensional geometry of binary neural networks. In Proceedings of the International Conference on Learning Representations (ICLR’18).

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural network. In Proceedings of the 32nd International Conference on Machine Learning (PMLR’15). 1613–1622.

[3] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcellos. 2017. Deep learning with low precision by half-wave Gaussian quantization. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 2018. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence 40, 4 (April 2018), 834–848. DOI: https://doi.org/10.1109/TPAMI.2017.2699184

[5] Y. Chen, T. Krishna, J. S. Emer, and V. Sze. 2017. Eyeriers: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE Journal of Solid-State Circuits 52, 1 (Jan. 2017), 127–138. DOI: https://doi.org/10.1109/JSSC.2016.2616357

[6] Yinpeng Dong, Renkun Ni, Jianguo Li, Yurong Chen, Jun Zhu, and Hang Su. 2017. Learning accurate low-bit deep neural networks with stochastic quantization. In Proceedings of the British Machine Vision Conference (BMVC’17).

[7] Robert M. Gray and David L. Neuhoff. 1998. Quantization. IEEE Transactions on Information Theory 44, 6 (1998), 2325–2383.

[8] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In Proceedings of the 32nd International Conference on Machine Learning (ICML’15). 1737–1746.

[9] Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In Proceedings of the International Conference on Learning Representations (ICLR’16).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 770–778.

[11] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-Rahman Mohamed, Navdeep Jaitly, Andrew Senior, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29, 6 (Nov. 2012), 82–97. DOI: https://doi.org/10.1109/MSP.2012.2205597

[12] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861
UNIQ: Uniform Noise Injection for Non-Uniform Quantization of Neural Networks

13. Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In Advances in Neural Information Processing Systems. 4107–4115.

14. Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. Journal of Machine Learning Research 18, 1 (2017), 6869–6898.

15. P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos. 2016. Stripes: Bit-serial deep neural network computing. In Proceedings of the 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO’16). 1–12. DOI: https://doi.org/10.1109/MICRO.2016.7783722

16. Alex Krizhevsky. 2009. Learning Multiple Layers of Features From Tiny Images. Master’s Thesis. Department of Computer Science, University of Toronto.

17. Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI’15). 2267–2273.

18. Edward H. Lee, Daisuke Miyashita, Elaina Chai, Boris Murmann, and S. Simon Wong. 2017. LogNet: Energy-efficient neural networks using logarithmic computation. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP’17). IEEE, Los Alamitos, CA, 5900–5904.

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

20. Christos Louizos, Karen Ullrich, and Max Welling. 2017. Bayesian compression for deep learning. In Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Red Hook, NY, 3288–3298.

21. Asit Mishra and Debbie Marr. 2018. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In Proceedings of the International Conference on Learning Representations (ICLR’18).

22. Asit Mishra, Eriko Nurvitadhi, Jeffrey J. Cook, and Debbie Marr. 2018. WRPN: Wide reduced-precision networks. In Proceedings of the International Conference on Learning Representations (ICLR’18).

23. Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. 2017. Variational dropout sparsifies deep neural networks. In Proceedings of the International Conference on Machine Learning.

24. Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. 2018. Value-aware quantization for training and inference of neural networks. In Proceedings of the European Conference on Computer Vision (ECCV’18).

25. Antonio Polino, Razvan Pascaru, and Dan Alistarh. 2018. Model compression via distillation and quantization. In Proceedings of the International Conference on Learning Representations (ICLR’18).

26. Mohammad Rastegar, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. XNOR-Net: ImageNet classification using binary convolutional neural networks. In Proceedings of the European Conference on Computer Vision. 525–542.

27. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, et al. 2015. ImageNet large scale visual recognition challenge. International Journal of Computer Vision 115, 3 (2015), 211–252. DOI: https://doi.org/10.1007/s11263-015-0816-y

28. Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Vikas Chandra, and Hadi Esmailzadeh. 2018. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks. In Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA’18). IEEE, Los Alamitos, CA, 764–775. DOI: https://doi.org/10.1109/ISCA.2018.00069

29. Tao Sheng, Chen Feng, Shaojie Zhou, Xiaopeng Zhang, Liang Shen, and Mickey Alexske. 2018. A quantization-friendly separable convolution for MobileNets. In Proceedings of the 2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2’18). 14–18. DOI: https://doi.org/10.1109/EMC2.2018.00011

30. Karen Ullrich, Edward Meeds, and Max Welling. 2017. Soft weight-sharing for neural network compression. In Proceedings of the International Conference on Learning Representations (ICLR’17).

31. Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong. 2018. Deep neural network compression with single and multiple level quantization. arXiv:1803.03289

32. Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. 2017. Incremental network quantization: Towards lossless CNNs with low-precision weights. In Proceedings of the International Conference on Learning Representations (ICLR’17).

33. Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. 2016. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv:1606.06160

34. Shu-Chang Zhou, Yu-Zhi Wang, He Wen, Qin-Yao He, and Yu-Heng Zou. 2017. Balanced quantization: An effective and efficient approach to quantized neural networks. Journal of Computer Science and Technology 32, 4 (2017), 667–682.

35. Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. 2016. Trained ternary quantization. In Proceedings of the International Conference on Learning Representations (ICLR’16).

Received June 2020; accepted December 2020