A Survey of Quantization Methods for Efficient Neural Network Inference

Amir Gholami*, Sehoon Kim*, Zhen Dong*, Zhewei Yao*, Michael W. Mahoney, Kurt Keutzer
University of California, Berkeley
{amirgh, sehoonkim, zhendong, zheweiy, mahoneymw, keutzer}@berkeley.edu

***Abstract*—As soon as abstract mathematical computations were adapted to computation on digital computers, the problem of efficient representation, manipulation, and communication of the numerical values in those computations arose. Strongly related to the problem of numerical representation is the problem of quantization: in what manner should a set of continuous real-valued numbers be distributed over a fixed discrete set of numbers to minimize the number of bits required and also to maximize the accuracy of the attendant computations? This perennial problem of quantization is particularly relevant whenever memory and/or computational resources are severely restricted, and it has come to the forefront in recent years due to the remarkable performance of Neural Network models in computer vision, natural language processing, and related areas. Moving from floating-point representations to low-precision fixed integer values represented in four bits or less holds the potential to reduce the memory footprint and latency by a factor of 16x; and, in fact, reductions of 4x to 8x are often realized in practice in these applications. Thus, it is not surprising that quantization has emerged recently as an important and very active sub-area of research in the efficient implementation of computations associated with Neural Networks. In this article, we survey approaches to the problem of quantizing the numerical values in deep Neural Network computations, covering the advantages/disadvantages of current methods. With this survey and its organization, we hope to have presented a useful snapshot of the current research in quantization for Neural Networks and to have given an intelligent organization to ease the evaluation of future research in this area.**

I. Introduction

Over the past decade, we have observed significant improvements in the accuracy of Neural Networks (NNs) for a wide range of problems, often achieved by highly over-parameterized models. While the accuracy of these over-parameterized (and thus very large) NN models has significantly increased, the sheer size of these models means that it is not possible to deploy them for many resource-constrained applications. This creates a problem for realizing pervasive deep learning, which requires real-time inference, with low energy consumption and high accuracy, in resource-constrained environments. This pervasive deep learning is expected to have a significant impact on a wide range of applications such as real-time intelligent healthcare monitoring, autonomous driving, audio analytics, and speech recognition.

Achieving efficient, real-time NNs with optimal accuracy requires rethinking the design, training, and deployment of NN models [71]. There is a large body of literature that has focused on addressing these issues by making NN models more efficient (in terms of latency, memory footprint, and energy consumption, etc.), while still providing optimal accuracy/generalization trade-offs. These efforts can be broadly categorized as follows.

*a) **Designing efficient NN model architectures**:* One line of work has focused on optimizing the NN model architecture in terms of its micro-architecture [100, 110, 125, 163, 164, 205, 245, 272] (e.g., kernel types such as depth-wise convolution or low-rank factorization) as well as its macro-architecture [99, 100, 103, 109, 207, 225] (e.g., module types such as residual, or inception). The classical techniques here mostly found new architecture modules using manual search, which is not scalable. As such, a new line of work is to design Automated machine learning (AutoML) and Neural Architecture Search (NAS) methods. These aim to find in an automated way the right NN architecture, under given constraints of model size, depth, and/or width [158, 188, 224, 237, 244, 282]. We refer interested reader to [54] for a recent survey of NAS methods.

*b) **Co-designing NN architecture and hardware together**:* Another recent line of work has been to adapt (and co-design) the NN architecture for a particular target hardware platform. The importance of this is because the overhead of a NN component (in terms of latency and energy) is hardware-dependent. For example, hardware

*Equal contribution.

with a dedicated cache hierarchy can execute bandwidth bound operations much more efficiently than hardware without such cache hierarchy. Similar to NN architecture design, initial approaches at architecture-hardware co-design were manual, where an expert would adapt/change the NN architecture [70], followed by using automated AutoML and/or NAS techniques [22, 23, 99, 244].

*c) **Pruning**:* Another approach to reducing the memory footprint and computational cost of NNs is to apply pruning. In pruning, neurons with small *saliency* (sensitivity) are removed, resulting in a sparse computational graph. Here, neurons with small saliency are those whose removal minimally affects the model output/loss function. Pruning methods can be broadly categorized into unstructured pruning [49, 85, 137, 141, 186, 249], and structured pruning [90, 105, 153, 162, 266, 267, 271]. With unstructured pruning, one removes neurons with with small saliency, wherever they occur. With this approach, one can perform aggressive pruning, removing most of the NN parameters, with very little impact on the generalization performance of the model. However, this approach leads to sparse matrix operations, which are known to be hard to accelerate, and which are typically memory-bound [21, 66]. On the other hand, with structured pruning, a group of parameters (e.g., entire convolutional filters) is removed. This has the effect of changing the input and output shapes of layers and weight matrices, thus still permitting dense matrix operations. However, aggressive structured pruning often leads to significant accuracy degradation. Training and inference with high levels of pruning/sparsity, while maintaining state-of-the-art performance, has remained an open problem [16]. We refer the interested reader to [66, 95, 132] for a thorough survey of related work in pruning/sparsity.

*d) **Knowledge distillation**:* Model distillation [3, 94, 148, 173, 189, 200, 261, 262] involves training a large model and then using it as a teacher to train a more compact model. Instead of using "hard" class labels during the training of the student model, the key idea of model distillation is to leverage the "soft" probabilities produced by the teacher, as these probabilities can contain more information about the input. Despite the large body of work on distillation, a major challenge here is to achieve a high compression ratio with distillation alone. Compared to quantization and pruning, which can maintain the performance with $\geq 4\times$ compression (with INT8 and lower precision), knowledge distillation methods tend to have non-negligible accuracy degradation with aggressive compression. However, the combination of knowledge distillation with prior methods (i.e., quantization and pruning) has shown great success [189].

*e) **Quantization**:* Finally, quantization is an approach that has shown great and consistent success in both training and inference of NN models. While the problems of numerical representation and quantization are as old as digital computing, Neural Nets offer unique opportunities for improvement. While this survey on quantization is mostly focused on inference, we should emphasize that an important success of quantization has been in NN training [10, 35, 57, 128, 239]. In particular, the breakthroughs of half-precision and mixed-precision training [41, 72, 78, 171] have been the main drivers that have enabled an order of magnitude higher throughput in AI accelerators. However, it has proven very difficult to go below half-precision without significant tuning, and most of the recent quantization research has focused on inference. This quantization for inference is the focus of this article.

*f) **Quantization and Neuroscience**:* Loosely related to (and for some a motivation for) NN quantization is work in neuroscience that suggests that the human brain stores information in a discrete/quantized form, rather than in a continuous form [167, 228, 232]. A popular rationale for this idea is that information stored in continuous form will inevitably get corrupted by noise (which is always present in the physical environment, including our brains, and which can be induced by thermal, sensory, external, synaptic noise, etc.) [27, 58]. However, discrete signal representations can be more robust to such low-level noise. Other reasons, including the higher generalization power of discrete representations [126, 136, 234] and their higher efficiency under limited resources [233], have also been proposed. We refer the reader to [220] for a thorough review of related work in neuroscience literature.

The goal of this work is to introduce current methods and concepts used in quantization and to discuss the current challenges and opportunities in this line of research. In doing so, we have tried to discuss most relevant work. It is not possible to discuss every work in a field as large as NN quantization in the page limit of a short survey; and there is no doubt that we have missed some relevant papers. We apologize in advance both to the readers and the authors of papers that we may have neglected.

In terms of the structure of this survey, we will first provide a brief history of quantization in Section II, and then we will introduce basic concepts underlying quantization in Section III. These basic concepts are

shared with most of the quantization algorithms, and they are necessary for understanding and deploying existing methods. Then we discuss more advanced topics in Section IV. These mostly involve recent state-of-the-art methods, especially for low/mixed-precision quantization. Then we discuss the implications of quantization in hardware accelerators in Section V, with a special focus on edge processors. Finally, we provide a summary and conclusions in Section VII.

II. General History of Quantization

Gray and Neuhoff have written a very nice survey of the history of quantization up to 1998 [76]. The article is an excellent one and merits reading in its entirety; however, for the reader's convenience we will briefly summarize some of the key points here. Quantization, as a method to map from input values in a large (often continuous) set to output values in a small (often finite) set, has a long history. Rounding and truncation are typical examples. Quantization is related to the foundations of the calculus, and related methods can be seen in the early 1800s (as well as much earlier), e.g., in early work on least-squares and related techniques for large-scale (by the standards of the early 1800s) data analysis [217]. An early work on quantization dates back to 1867, where discretization was used to approximate the calculation of integrals [199]; and, subsequently, in 1897, when Shappard investigated the impact of rounding errors on the integration result [213]. More recently, quantization has been important in digital signal processing, as the process of representing a signal in digital form ordinarily involves rounding, as well as in numerical analysis and the implementation of numerical algorithms, where computations on real-valued numbers are implemented with finite-precision arithmetic.

It was not until 1948, around the advent of the digital computer, when Shannon wrote his seminal paper on the mathematical theory of communication [208], that the effect of quantization and its use in coding theory were formally presented. In particular, Shannon argued in his lossless coding theory that using the same number of bits is wasteful, when events of interest have a non-uniform probability. He argued that a more optimal approach would be to vary the number of bits based on the probability of an event, a concept that is now known as *variable-rate quantization*. Huffman coding in particular is motivated by this [108]. In subsequent work in 1959 [209], Shannon introduced distortion-rate functions (which provide a lower bound on the signal distortion after coding) as well as the notion of vector quantization (also briefly discussed in Section IV-F). This concept was extended and became practical in [53, 55, 67, 201] for real communication applications. Other important historical research on quantization in signal processing in that time period includes [183], which introduced the Pulse Code Modulation (PCM) concept (a pulsing method proposed to approximate/represent/encode sampled analog signals), as well as the classical result of high resolution quantization [14]. We refer the interested reader to [76] for a detailed discussion of these issues.

Quantization appears in a slightly different way in algorithms that use numerical approximation for problems involving continuous mathematical quantities, an area that also has a long history, but that also received renewed interest with the advent of the digital computer. In numerical analysis, an important notion was (and still is) that of a *well-posed problem*—roughly, a problem is well-posed if: a solution exists; that solution is unique; and that solution depends continuously on the input data in some reasonable topology. Such problems are sometimes called *well-conditioned problems*. It turned out that, even when working with a given well-conditioned problem, certain algorithms that solve that problem "exactly" in some idealized sense perform very poorly in the presence of "noise" introduced by the peculiarities of roundoff and truncation errors. These roundoff errors have to do with representing real numbers with only finitely-many bits—a quantization specified, e.g., by the IEEE floating point standard; and truncation errors arise since only a finite number of iterations of an iterative algorithm can actually be performed. The latter are important even in "exact arithmetic," since most problems of continuous mathematics cannot even in principle be solved by a finite sequence of elementary operations; but the former have to do with quantization. These issues led to the notion of the *numerical stability* of an algorithm. Let us view a numerical algorithm as a function f attempting to map the input data x to the "true" solution y; but due to roundoff and truncation errors, the output of the algorithm is actually some other y^*. In this case, the *forward error* of the algorithm is $\Delta y = y^* - y$; and the *backward error* of the algorithm is the smallest Δx such that $f(x + \Delta x) = y^*$. Thus, the forward error tells us the difference between the exact or true answer and what was output by the algorithm; and the backward error tells us what input data the algorithm we ran actually solved exactly. The forward error and backward error for an algorithm are related by the condition number of the problem. We refer the interested reader to [229] for a detailed discussion of these issues.

A. Quantization in Neural Nets

No doubt thousands of papers have been written on these topics, and one might wonder: how is recent work on NN quantization different from these earlier works? Certainly, many of the recently proposed "novel algorithms" have strong connections with (and in some cases are essentially rediscoveries of) past work in the literature. However, NNs bring unique challenges and opportunities to the problem of quantization. First, inference and training of Neural Nets are both computationally intensive. So, the efficient representation of numerical values is particularly important. Second, most current Neural Net models are heavily over-parameterized, so there is ample opportunity for reducing bit precision without impacting accuracy. However, one very important difference is that NNs are very robust to aggressive quantization and extreme discretization. The new degree of freedom here has to do with the number of parameters involved, i.e., that we are working with over-parameterized models. This has direct implications for whether we are solving well-posed problems, whether we are interested in forward error or backward error, etc. In the NN applications driving recent developments in quantization, there is not a single well-posed or well-conditioned problem that is being solved. Instead, one is interested in some sort of forward error metric (based on classification quality, perplexity, etc.), but due to the over-parameterization, there are many very different models that exactly or approximately optimize this metric. Thus, it is possible to have high error/distance between a quantized model and the original non-quantized model, while still attaining very good generalization performance. This added degree of freedom was not present in many of the classical research, which mostly focused on finding compression methods that would not change the signal too much, or with numerical methods in which there was strong control on the difference between the "exact" versus the "discretized" computation. This observation that has been the main driver for researching *novel* techniques for NN quantization. Finally,the layered structure of Neural Net models offers an additional dimension to explore. Different layers in a Neural Net have different impact on the loss function, and this motivates a mixed-precision approach to quantization.

III. Basic Concepts of Quantization

In this section, we first briefly introduce common notations and the problem setup in Section III-A, and then we describe the basic quantization concepts and methods in Section III-B-III-F. Afterwards, we discuss the different fine-tuning methods in Section III-G, followed by stochastic quantization in Section III-H.

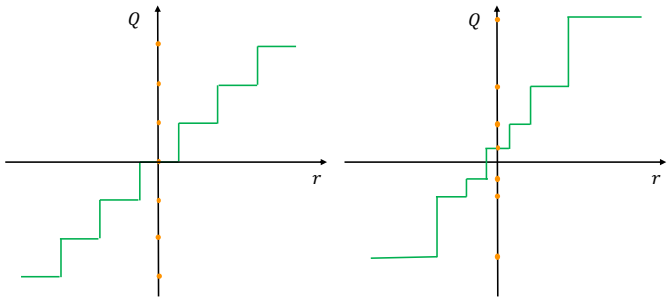

Figure 1: *Comparison between uniform quantization (left) and non-uniform quantization (right). Real values in the continuous domain r are mapped into discrete, lower precision values in the quantized domain Q, which are marked with the orange bullets. Note that the distances between the quantized values (quantization levels) are the same in uniform quantization, whereas they can vary in non-uniform quantization.*

A. Problem Setup and Notations

Assume that the NN has L layers with learnable parameters, denoted as $\{W_1, W_2, ..., W_L\}$, with θ denoting the combination of all such parameters. Without loss of generality, we focus on the supervised learning problem, where the nominal goal is to optimize the following empirical risk minimization function:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(x_i, y_i; \theta), \tag{1}$$

where (x, y) is the input data and the corresponding label, $l(x, y; \theta)$ is the loss function (e.g., Mean Squared Error or Cross Entropy loss), and N is the total number of data points. Let us also denote the input hidden activations of the i^{th} layer as h_i, and the corresponding output hidden activation as a_i. We assume that we have the trained model parameters θ, stored in floating point precision. In quantization, the goal is to reduce the precision of both the parameters (θ), as well as the intermediate activation maps (i.e., h_i, a_i) to low-precision, with minimal impact on the generalization power/accuracy of the model. To do this, we need to define a quantization operator that maps a floating point value to a quantized one, which is described next.

B. Uniform Quantization

We need first to define a function that can quantize NN weights and activations to a finite set of values. This

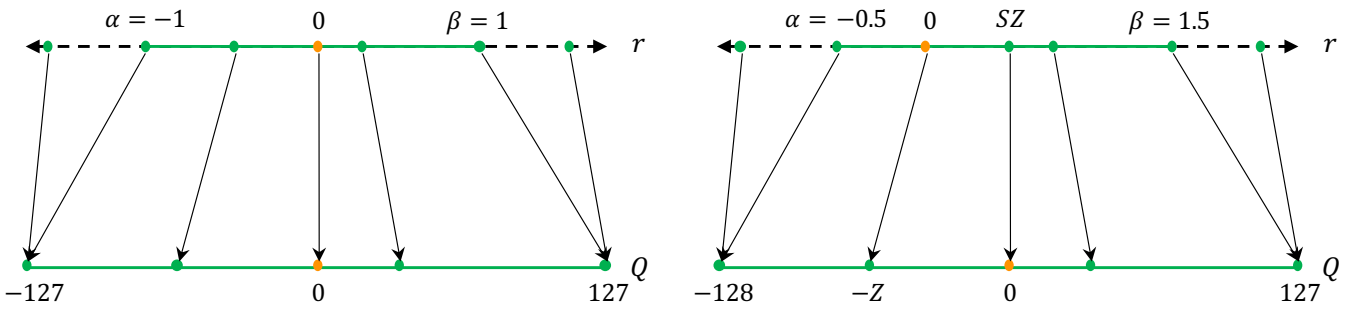

Figure 2: *Illustration of symmetric quantization and asymmetric quantization. Symmetric quantization with restricted range maps real values to [-127, 127], and full range maps to [-128, 127] for 8-bit quantization.*

function takes real values in floating point, and it maps them to a lower precision range, as illustrated in Figure 1. A popular choice for a quantization function is as follows:

$$Q(r) = \text{Int}\big(r/S\big) - Z, \tag{2}$$

where Q is the quantization operator, r is a real valued input (activation or weight), S is a real valued scaling factor, and Z is an integer zero point. Furthermore, the Int function maps a real value to an integer value through a rounding operation (e.g., round to nearest and truncation). In essence, this function is a mapping from real values r to some integer values. This method of quantization is also known as *uniform quantization*, as the resulting quantized values (aka quantization levels) are uniformly spaced (Figure 1, left). There are also *non-uniform quantization* methods whose quantized values are not necessarily uniformly spaced (Figure 1, right), and these methods will be discussed in more detail in Section III-F. It is possible to recover real values r from the quantized values $Q(r)$ through an operation that is often referred to as *dequantization*:

$$\tilde{r} = S(Q(r) + Z). \tag{3}$$

Note that the recovered real values $\tilde{r}$ will not exactly match r due to the rounding operation.

C. Symmetric and Asymmetric Quantization

One important factor in uniform quantization is the choice of the scaling factor S in Eq. 2. This scaling factor essentially divides a given range of real values r into a number of partitions (as discussed in [112, 131]):

$$S = \frac{\beta - \alpha}{2^b - 1}, \tag{4}$$

where $[\alpha, \beta]$ denotes the clipping range, a bounded range that we are clipping the real values with, and b is the quantization bit width. Therefore, in order for the scaling factor to be defined, the clipping range $[\alpha, \beta]$ should first be determined. The process of choosing the clipping range is often referred to as *calibration*. A straightforward choice is to use the min/max of the signal for the clipping range, i.e., $\alpha = r_{min}$, and $\beta = r_{max}$. This approach is an *asymmetric quantization* scheme, since the clipping range is not necessarily symmetric with respect to the origin, i.e., $-\alpha \neq \beta$, as illustrated in Figure 2 (Right). It is also possible to use a *symmetric quantization* scheme by choosing a symmetric clipping range of $\alpha = -\beta$. A popular choice is to choose these based on the min/max values of the signal: $-\alpha = \beta = \max(|r_{max}|, |r_{min}|)$. Asymmetric quantization often results in a tighter clipping range as compared to symmetric quantization. This is especially important when the target weights or activations are imbalanced, e.g., the activation after ReLU that always has non-negative values. Using symmetric quantization, however, simplifies the quantization function in Eq. 2 by replacing the zero point with $Z = 0$:

$$Q(r) = \text{Int}\left(\frac{r}{S}\right). \tag{5}$$

Here, there are two choices for the scaling factor. In "full range" symmetric quantization S is chosen as $\frac{2max(|r|)}{2^n - 1}$ (with floor rounding mode), to use the full INT8 range of [-128,127]. However, in "restricted range" S is chosen as $\frac{max(|r|)}{2^{n-1}-1}$, which only uses the range of [-127,127]. As expected, the full range approach is more accurate. Symmetric quantization is widely adopted in practice for quantizing weights because zeroing out the zero point can lead to reduction in computational cost during inference [247], and also makes the implementation more straightforward. However, note that for activation the cross terms occupying due to the offset in the asymmetric activations are a static data independent term

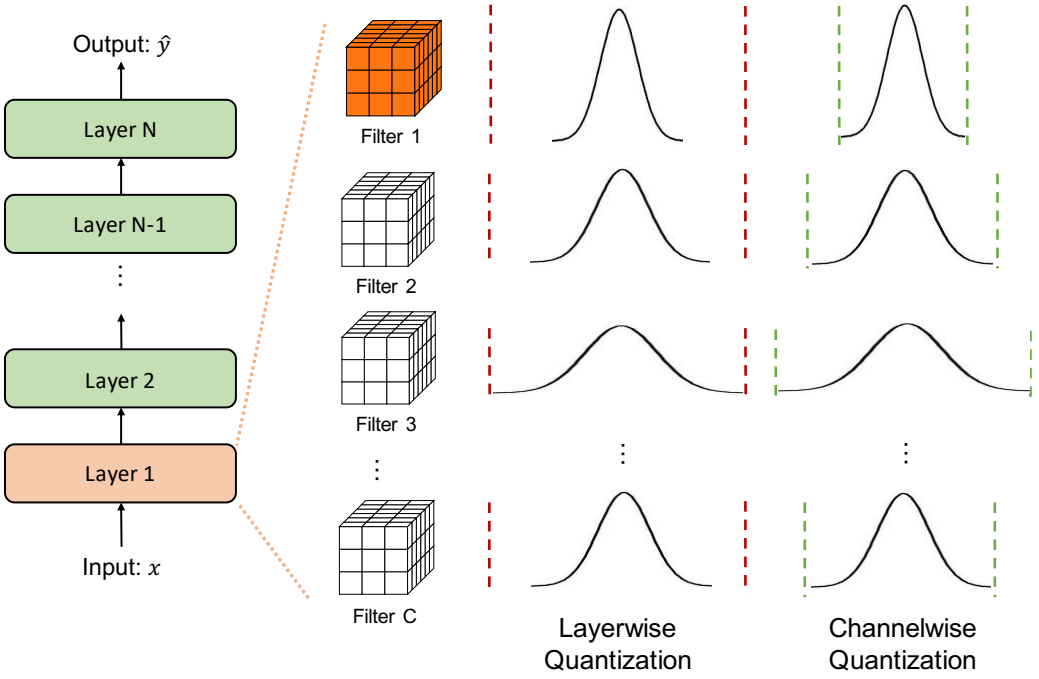

Figure 3: *Illustration of different quantization granularities. In layerwise quantization, the same clipping range is applied to all the filters that belong to the same layer. This can result in bad quantization resolution for the channels that have narrow distributions (e.g., Filter 1 in the figure). One can achieve better quantization resolution using channelwise quantization that dedicates different clipping ranges to different channels.*

and can be absorbed in the bias (or used to initialize the accumulator) [15].

Using the min/max of the signal for both symmetric and asymmetric quantization is a popular method. However, this approach is susceptible to outlier data in the activations. These could unnecessarily increase the range and, as a result, reduce the resolution of quantization. One approach to address this is to use percentile instead of min/max of the signal [168]. That is to say, instead of the largest/smallest value, the i-th largest/smallest values are used as β/α. Another approach is to select α and β to minimize KL divergence (i.e., information loss) between the real values and the quantized values [172]. We refer the interested readers to [247] where the different calibration methods are evaluated on various models.

Summary (Symmetric vs Asymmetric Quantization). Symmetric quantization partitions the clipping using a symmetric range. This has the advantage of easier implementation, as it leads to $Z = 0$ in Eq. 2. However, it is sub-optimal for cases where the range could be skewed and not symmetric. For such cases, asymmetric quantization is preferred.

D. Range Calibration Algorithms: Static vs Dynamic Quantization

So far, we discussed different calibration methods for determining the clipping range of $[\alpha, \beta]$. Another important differentiator of quantization methods is *when* the clipping range is determined. This range can be computed statically for weights, as in most cases the parameters are fixed during inference. However, the activation maps differ for each input sample (x in Eq. 1). As such, there are two approaches to quantizing activations: *dynamic quantization*, and *static quantization*.

In dynamic quantization, this range is *dynamically* calculated for each activation map during runtime. This approach requires real-time computation of the signal statistics (min, max, percentile, etc.) which can have a very high overhead. However, dynamic quantization often results in higher accuracy as the signal range is exactly calculated for each input.

Another quantization approach is static quantization, in which the clipping range is pre-calculated and *static* during inference. This approach does not add any computational overhead, but it typically results in lower accuracy as compared to dynamic quantization. One popular method for the pre-calculation is to run a

series of calibration inputs to compute the typical range of activations [112, 259]. Multiple different metrics have been proposed to find the best range, including minimizing Mean Squared Error (MSE) between original unquantized weight distribution and the corresponding quantized values [40, 214, 221, 273]. One could also consider using other metrics such as entropy [184], although MSE is the most common method used. Another approach is to learn/impose this clipping range during NN training [36, 144, 268, 278]. Notable work here are LQNets [268], PACT [36], LSQ [56], and LSQ+ [15] which jointly optimizes the clipping range and the weights in NN during training.

Summary (Dynamic vs Static Quantization). Dynamic quantization dynamically computes the clipping range of each activation and often achieves the highest accuracy. However, calculating the range of a signal dynamically is very expensive, and as such, practitioners most often use static quantization where the clipping range is fixed for all inputs.

E. Quantization Granularity

In most computer vision tasks, the activation input to a layer is convolved with many different convolutional filters, as illustrated in Figure 3. Each of these convolutional filters can have a different range of values. As such, one differentiator for quantization methods is the granularity of how the clipping range $[\alpha, \beta]$ is calculated for the weights. We categorized them as follows.

a) Layerwise Quantization: In this approach, the clipping range is determined by considering all of the weights in convolutional filters of a layer [131], as shown in the third column of Figure 3. Here one examines the statistics of the entire parameters in that layer (e.g., min, max, percentile, etc.), and then uses the same clipping range for all the convolutional filters. While this approach is very simple to implement, it often results in sub-optimal accuracy, as the range of each convolutional filter can be vary a lot. For example, a convolutional kernel that has relatively narrower range of parameters may lose its quantization resolution due to another kernel in the same layer with a wider range.

b) Groupwise Quantization: One could group multiple different channels inside a layer to calculate the clipping range (of either activations or convolution kernels). This could be helpful for cases where the distribution of the parameters across a single convolution/activation varies a lot. For instance, this approach was found useful in Q-BERT [212] for quantizing Transformer [235] models that consist of fully-connected attention layers. However, this approach inevitably comes with the extra cost of accounting for different scaling factors.

c) Channelwise Quantization: A popular choice of the clipping range is to use a fixed value for each convolutional filter, independent of other channels [104, 112, 131, 215, 268, 276], as shown in the last column of Figure 3. That is to say, each channel is assigned a dedicated scaling factor. This ensures a better quantization resolution and often results in higher accuracy.

d) Sub-channelwise Quantization: The previous approach could be taken to the extreme, where the clipping range is determined with respect to any groups of parameters in a convolution or fully-connected layer. However, this approach could add considerable overhead, since the different scaling factors need to be taken into account when processing a single convolution or fully-connected layer. Therefore, groupwise quantization could establish a good compromise between the quantization resolution and the computation overhead.

Summary (Quantization Granularity). Channelwise quantization is currently the standard method used for quantizing convolutional kernels. It enables the practitioner to adjust the clipping range for each individual kernel with negligible overhead. In contrast, sub-channelwise quantization may result in significant overhead and is not currently the standard choice.

F. Non-Uniform Quantization

Some work in the literature has also explored non-uniform quantization [25, 38, 62, 74, 78, 98, 117, 123, 150, 156, 175, 184, 185, 230, 240, 248, 256, 258, 268, 275], where quantization steps as well as quantization levels are allowed to be non-uniformly spaced. The formal definition of non-uniform quantization is shown in Eq. 6, where X_i represents the discrete quantization levels and Δ_i the quantization steps (thresholds):

$$Q(r) = X_i, \text{ if } r \in [\Delta_i, \Delta_{i+1}). \tag{6}$$

Specifically, when the value of a real number r falls in between the quantization step Δ_i and Δ_{i+1}, quantizer Q projects it to the corresponding quantization level X_i. Note that neither X_i's nor Δ_i's are uniformly spaced.

Non-uniform quantization may achieve higher accuracy for a fixed bit-width, because one could better capture the distributions by focusing more on important value regions or finding appropriate dynamic ranges. For instance, many non-uniform quantization methods have been designed for bell-shaped distributions of the weights and activations that often involve long tails [12, 25, 61, 114, 145, 175]. A typical rule-based non-uniform quantization is to

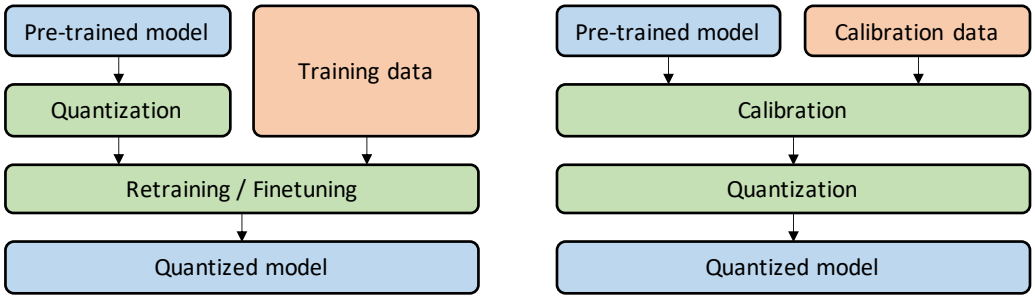

Figure 4: *Comparison between Quantization-Aware Training (QAT, Left) and Post-Training Quantization (PTQ, Right). In QAT, a pre-trained model is quantized and then finetuned using training data to adjust parameters and recover accuracy degradation. In PTQ, a pre-trained model is calibrated using calibration data (e.g., a small subset of training data) to compute the clipping ranges and the scaling factors. Then, the model is quantized based on the calibration result. Note that the calibration process is often conducted in parallel with the finetuning process for QAT.*

use a logarithmic distribution [175, 274], where the quantization steps and levels increase exponentially instead of linearly. Another popular branch is *binary-code-based* quantization [77, 106, 117, 250, 268] where a real-number vector $\mathbf{r} \in \mathbb{R}^n$ is quantized into m binary vectors by representing $\mathbf{r} \approx \sum_{i=1}^{m} \alpha_i \mathbf{b}_i$, with the scaling factors $\alpha_i \in \mathbb{R}$ and the binary vectors $\mathbf{b}_i \in \{-1, +1\}^n$. Since there is no closed-form solution for minimizing the error between $\mathbf{r}$ and $\sum_{i=1}^{m} \alpha_i \mathbf{b}_i$, previous research relies on heuristic solutions. To further improve the quantizer, more recent work [77, 226, 250] formulates non-uniform quantization as an optimization problem. As shown in Eq. 7, the quantization steps/levels in the quantizer Q are adjusted to minimize the difference between the original tensor and the quantized counterpart.

$$\min_{Q} \|Q(r) - r\|^2 \tag{7}$$

Furthermore, the quantizer itself can also be jointly trained with the model parameters. These methods are referred to as learnable quantizers, and the quantization steps/levels are generally trained with iterative optimization [250, 268] or gradient descent [123, 155, 256].

In addition to rule-based and optimization-based non-uniform quantization, clustering can also be beneficial to alleviate the information loss due to quantization. Some works [74, 248] use k-means on different tensors to determine the quantization steps and levels, while other work [38] applies a Hessian-weighted k-means clustering on weights to minimize the performance loss. Further discussion can be found in Section IV-F.

Summary (Uniform vs Non-uniform Quantization). Generally, non-uniform quantization enables us to better capture the signal information, by assigning bits and discreitizing the range of parameters non-uniformly. However, non-uniform quantization schemes are typically difficult to deploy efficiently on general computation hardware, e.g., GPU and CPU. As such, the uniform quantization is currently the de-facto method due to its simplicity and its efficient mapping to hardware.

G. Fine-tuning Methods

It is often necessary to adjust the parameters in the NN after quantization. This can either be performed by re-training the model, a process that is called Quantization-Aware Training (QAT), or done without re-training, a process that is often referred to as Post-Training Quantization (PTQ). A schematic comparison between these two approaches is illustrated in Figure 4, and further discussed below.

1) Quantization-Aware Training: Given a trained model, quantization may introduce a perturbation to the trained model parameters, and this can push the model away from the point to which it had converged when it was trained with floating point precision. It is possible to address this by re-training the NN model with quantized parameters so that the model can converge to a point with better loss. One popular approach is to use Quantization-Aware Training (QAT), in which the usual forward and backward pass are performed on the quantized model in floating point, but the model parameters are quantized after each gradient update (similar to projected gradient descent). In particular, it is important to do this projection after the weight update is performed in floating point precision. Performing the backward pass with floating point is important, as accumulating the gradients in quantized precision can result in zero-

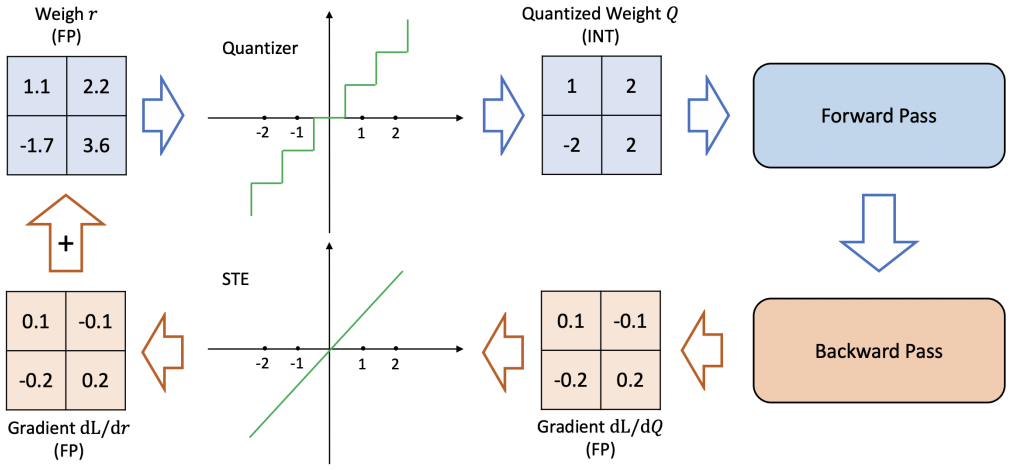

Figure 5: *Illustration of Quantization-Aware Training procedure, including the use of Straight Through Estimator (STE).*

gradient or gradients that have high error, especially in low-precision [42, 79, 80, 106, 156, 181, 198, 223].

An important subtlety in backpropagation is how the the non-differentiable quantization operator (Eq. 2) is treated. Without any approximation, the gradient of this operator is zero almost everywhere, since the rounding operation in Eq. 2 is a piece-wise flat operator. A popular approach to address this is to approximate the gradient of this operator by the so-called Straight Through Estimator (STE) [13]. STE essentially ignores the rounding operation and approximates it with an identity function, as illustrated in Figure 5.

Despite the coarse approximation of STE, it often works well in practice, except for ultra low-precision quantization such as binary quantization [8]. The work of [263] provides a theoretical justification for this phenomena, and it finds that the coarse gradient approximation of STE can in expectation correlate with population gradient (for a proper choice of STE). From a historical perspective, we should note that the original idea of STE can be traced back to the seminal work of [202, 203], where an identity operator was used to approximate gradient from the binary neurons.

While STE is the mainstream approach [218, 280], other approaches have also been explored in the literature [2, 25, 31, 59, 142, 160]. We should first mention that [13] also proposes a stochastic neuron approach as an alternative to STE (this is briefly discussed in Section III-H). Other approaches using combinatorial optimization [65], target propagation [138], or Gumbel-softmax [115] have also been proposed. Another different class of alternative methods tries to use regularization operators to enforce the weight to be quantized. This removes the need to use the non-differentiable quantization operator in Eq. 2. These are often referred to as *Non-STE* methods [4, 8, 39, 98, 142, 179, 274]. Recent research in this area includes ProxQuant [8] which removes the rounding operation in the quantization formula Eq. 2, and instead uses the so-called *W-shape*, non-smooth regularization function to enforce the weights to quantized values. Other notable research includes using pulse training to approximate the derivative of discontinuous points [45], or replacing the quantized weights with an affine combination of floating point and quantized parameters [161]. The recent work of [177] also suggests AdaRound, which is an adaptive rounding method as an alternative to round-to-nearest method. Despite interesting works in this area, these methods often require a lot of tuning and so far STE approach is the most commonly used method.

In addition to adjusting model parameters, some prior work found it effective to learn quantization parameters during QAT as well. PACT [36] learns the clipping ranges of activations under uniform quantization, while QIT [123] also learns quantization steps and levels as an extension to a non-uniform quantization setting. LSQ [56] introduces a new gradient estimate to learn scaling factors for non-negative activations (e.g., ReLU) during QAT, and LSQ+ [15] further extends this idea to general activation functions such as swish [196] and h-swish [99] that produce negative values.

Summary (QAT). QAT has been shown to work despite the coarse approximation of STE. However, the main disadvantage of QAT is the computational cost of

re-training the NN model. This re-training may need to be performed for several hundred epochs to recover accuracy, especially for low-bit precision quantization. If a quantized model is going to be deployed for an extended period, and if efficiency and accuracy are especially important, then this investment in re-training is likely to be worth it. However, this is not always the case, as some models have a relatively short lifetime. Next, we next discuss an alternative approach that does not have this overhead.

2) Post-Training Quantization: An alternative to the expensive QAT method is Post-Training Quantization (PTQ) which performs the quantization and the adjustments of the weights, without any fine-tuning [11, 24, 40, 60, 61, 68, 69, 88, 107, 140, 146, 170, 178, 273]. As such, the overhead of PTQ is very low and often negligible. Unlike QAT, which requires a sufficient amount of training data for retraining, PTQ has an additional advantage that it can be applied in situations where data is limited or unlabeled. However, this often comes at the cost of lower accuracy as compared to QAT, especially for low-precision quantization.

For this reason, multiple approaches have been proposed to mitigate the accuracy degradation of PTQ. For example, [11, 63] observe inherent bias in the mean and variance of the weight values following their quantization and propose bias correction methods; and [170, 178] show that equalizing the weight ranges (and implicitly activation ranges) between different layers or channels can reduce quantization errors. ACIQ [11] analytically computes the optimal clipping range and the channel-wise bitwidth setting for PTQ. Although ACIQ can achieve low accuracy degradation, the channel-wise activation quantization used in ACIQ is hard to efficiently deploy on hardware. In order to address this, the OMSE method [40] removes channel-wise quantization on activation and proposes to conduct PTQ by optimizing the L2 distance between the quantized tensor and the corresponding floating point tensor. Furthermore, to better alleviate the adverse impact of outliers on PTQ, an outlier channel splitting (OCS) method is proposed in [273] which duplicates and halves the channels containing outlier values. Another notable work is AdaRound [177] which shows that the naive round-to-nearest method for quantization can counter-intuitively results in sub-optimal solutions, and it proposes an adaptive rounding method that better reduces the loss. While AdaRound restricts the changes of the quantized weights to be within ± 1 from their full-precision counterparts, AdaQuant [107] proposes a more general method that allows the quantized weights to change as needed. PTQ schemes can be taken to the extreme, where neither training nor testing data are utilized during quantization (aka zero-shot scenarios), which is discussed next.

Summary (PTQ). In PTQ, all the weights and activations quantization parameters are determined without any re-training of the NN model. As such, PTQ is a very fast method for quantizing NN models. However, this often comes at the cost of lower accuracy as compared to QAT.

3) Zero-shot Quantization: As discussed so far, in order to achieve minimal accuracy degradation after quantization, we need access to the entire of a fraction of training data. First, we need to know the range of activations so that we can clip the values and determine the proper scaling factors (which is usually referred to as calibration in the literature). Second, quantized models often require fine-tuning to adjust the model parameters and recover the accuracy degradation. In many cases, however, access to the original training data is not possible during the quantization procedure. This is because the training dataset is either too large to be distributed, proprietary (e.g., Google's JFT-300M), or sensitive due to security or privacy concerns (e.g., medical data). Several different methods have been proposed to address this challenge, which we refer to as zero-shot quantization (ZSQ). Inspired by [178], here we first describe two different levels of zero-shot quantization:

- **Level 1:** No data and no finetuning (ZSQ + PTQ).
- **Level 2:** No data but requires finetuning (ZSQ + QAT).

Level 1 allows faster and easier quantization without any finetuning. Finetuning is in general time-consuming and often requires additional hyperparamenter search. However, Level 2 usually results in higher accuracy, as finetuning helps the quantized model to recover the accuracy degradation, particularly in ultra-low bit precision settings [84]. The work of [178] uses a Level 1 approach that relies on equalizing the weight ranges and correcting bias errors to make a given NN model more amenable to quantization without any data or finetuning. However, as this method is based on the scale-equivariance property of (piece-wise) linear activation functions, it can be sub-optimal for NNs with non-linear activations, such as BERT [46] with GELU [93] activation or MobileNetV3 [99] with swish activation [197].

A popular branch of research in ZSQ is to generate synthetic data similar to the real data from which the target pre-trained model is trained. The synthetic data is then used for calibrating and/or finetuning the quantized

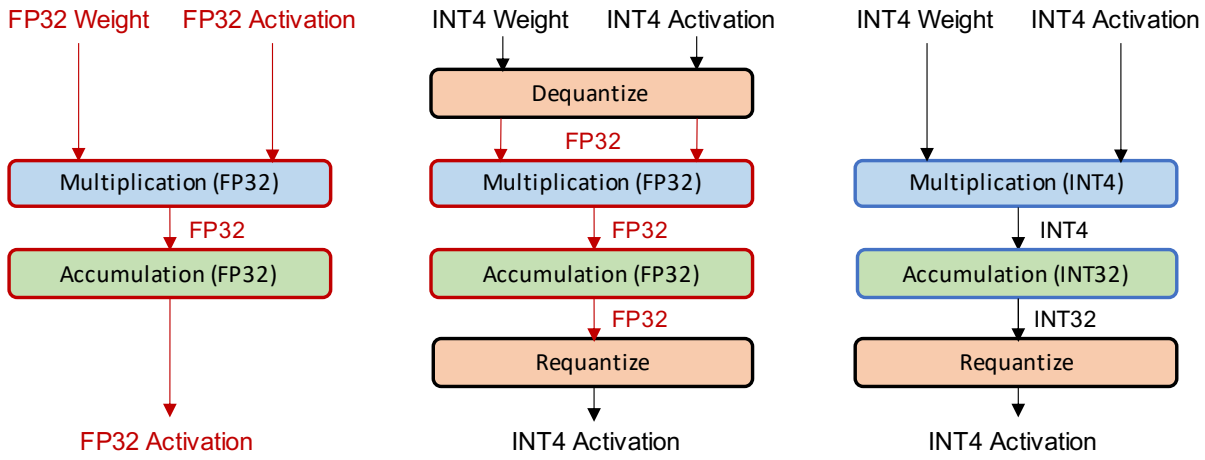

Figure 6: *Comparison between full-precision inference (Left), inference with simulated quantization (Middle), and inference with integer-only quantization (Right).*

model. An early work in this area [28] exploits Generative Adversarial Networks (GANs) [75] for synthetic data generation. Using the pre-trained model as a discriminator, it trains the generator so that its outputs can be well classified by the discriminator. Then, using the synthetic data samples collected from the generator, the quantized model can be finetuned with knowledge distillation from the full-precision counterpart (see Section IV-D for more details). However, this method fails to capture the internal statistics (e.g., distributions of the intermediate layer activations) of the real data, as it is generated only using the final outputs of the model. Synthetic data which does not take the internal statistics into account may not properly represent the real data distribution [84]. To address this, a number of subsequent efforts use the statistics stored in Batch Normalization (BatchNorm) [111], i.e., channel-wise mean and variance, to generate more realistic synthetic data. In particular, [84] generates data by directly minimizing the KL divergence of the internal statistics, and it uses the synthetic data to calibrate and finetune the quantized models. Furthermore, ZeroQ [24] shows that the synthetic data can be used for sensitivity measurement as well as calibration, thereby enabling mixed-precision post-training quantization without any access to the training/validation data. ZeroQ also extends ZSQ to the object detection tasks, as it does not rely on the output labels when generating data. Both [84] and [24] set the input images as trainable parameters and directly perform backpropagation on them until their internal statistics become similar to those of the real data. To take a step further, recent research [37, 89, 251] finds it effective to train and exploit generative models that can better capture the real data distribution and generate more realistic synthetic data.

Summary (ZSQ). Zero Shot (aka data free) quantization performs the entire quantization without any access to the training/validation data. This is particularly important for Machine Learning as a Service (MLaaS) providers who want to accelerate the deployment of a customer's workload, without the need to access their dataset. Moreover, this is important for cases where security or privacy concerns may limit access to the training data.

H. Stochastic Quantization

During inference, the quantization scheme is usually deterministic. However, this is not the only possibility, and some works have explored stochastic quantization for quantization aware training as well as reduced precision training [13, 78]. The high level intuition has been that the stochastic quantization may allow a NN to explore more, as compared to deterministic quantization. One popular supporting argument has been that small weight updates may not lead to any weight change, as the rounding operation may always return the same weights. However, enabling a stochastic rounding may provide the NN an opportunity to *escape*, thereby updating its parameters.

More formally, stochastic quantization maps the floating number up or down with a probability associated to the magnitude of the weight update. For instance, in [29, 78], the Int operator in Eq. 2 is defined as

$$\text{Int}(x) = \begin{cases} \lfloor x \rfloor & \text{with probability } \lceil x \rceil - x, \\ \lceil x \rceil & \text{with probability } x - \lfloor x \rfloor. \end{cases} \tag{8}$$

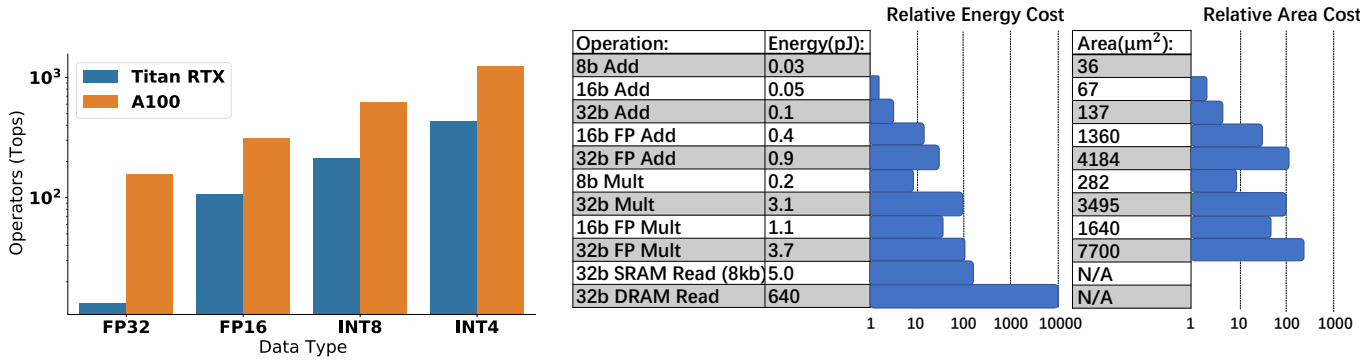

Figure 7: *(Left) Comparison between peak throughput for different bit-precision logic on Titan RTX and A100 GPU. (Right) Comparison of the corresponding energy cost and relative area cost for different precision for 45nm technology [96]. As one can see, lower precision provides exponentially better energy efficiency and higher throughput.*

However, this definition cannot be used for binary quantization. Hence, [42] extends this to

$$\text{Binary}(x) = \begin{cases} -1 & \text{with probability } 1-\sigma(x), \\ +1 & \text{with probability } \sigma(x), \end{cases} \tag{9}$$

where Binary is a function to binarize the real value x, and $\sigma(\cdot)$ is the sigmoid function.

Recently, another stochastic quantization method is introduced in QuantNoise [59]. QuantNoise quantizes a different random subset of weights during each forward pass and trains the model with unbiased gradients. This allows lower-bit precision quantization without significant accuracy drop in many computer vision and natural language processing models. However, a major challenge with stochastic quantization methods is the overhead of creating random numbers for every single weight update, and as such they are not yet adopted widely in practice.

IV. Advanced Concepts: Quantization Below 8 bits

In this section, we will discuss more advanced topics in quantization which are mostly used for sub-INT8 quantization. We will first discuss simulated quantization and its difference with integer-only quantization in Section IV-A. Afterward, we will discuss different methods for mixed-precision quantization in Section IV-B, followed by hardware-aware quantization in Section IV-C. Then we will describe how distillation can be used to boost the quantization accuracy in Section IV-D, and then we will discuss extremely low bit precision quantization in Section IV-E. Finally, we will briefly describe the different methods for vector quantization in Section IV-F.

A. Simulated and Integer-only Quantization

There are two common approaches to deploy a quantized NN model, *simulated quantization* (aka fake quantization) and *integer-only quantization* (aka fixed-point quantization). In simulated quantization, the quantized model parameters are stored in low-precision, but the operations (e.g. matrix multiplications and convolutions) are carried out with floating point arithmetic. Therefore, the quantized parameters need to be dequantized before the floating point operations as schematically shown in Figure 6 (Middle). As such, one cannot fully benefit from fast and efficient low-precision logic with simulated quantization. However, in integer-only quantization, all the operations are performed using low-precision integer arithmetic [112, 130, 151, 259], as illustrated in Figure 6 (Right). This permits the entire inference to be carried out with efficient integer arithmetic, without any floating point dequantization of any parameters or activations.

In general, performing the inference in full-precision with floating point arithmetic may help the final quantization accuracy, but this comes at the cost of not being able to benefit from the low-precision logic. Low-precision logic has multiple benefits over the full-precision counterpart in terms of latency, power consumption, and area efficiency. As shown in Figure 7 (left), many hardware processors, including NVIDIA V100 and Titan RTX, support fast processing of low-precision arithmetic that can boost the inference throughput and latency. Moreover, as illustrated in Figure 7 (right) for a 45nm technology [96], low-precision logic is significantly more efficient in terms of energy and area. For example, performing INT8 addition is 30× more energy efficient

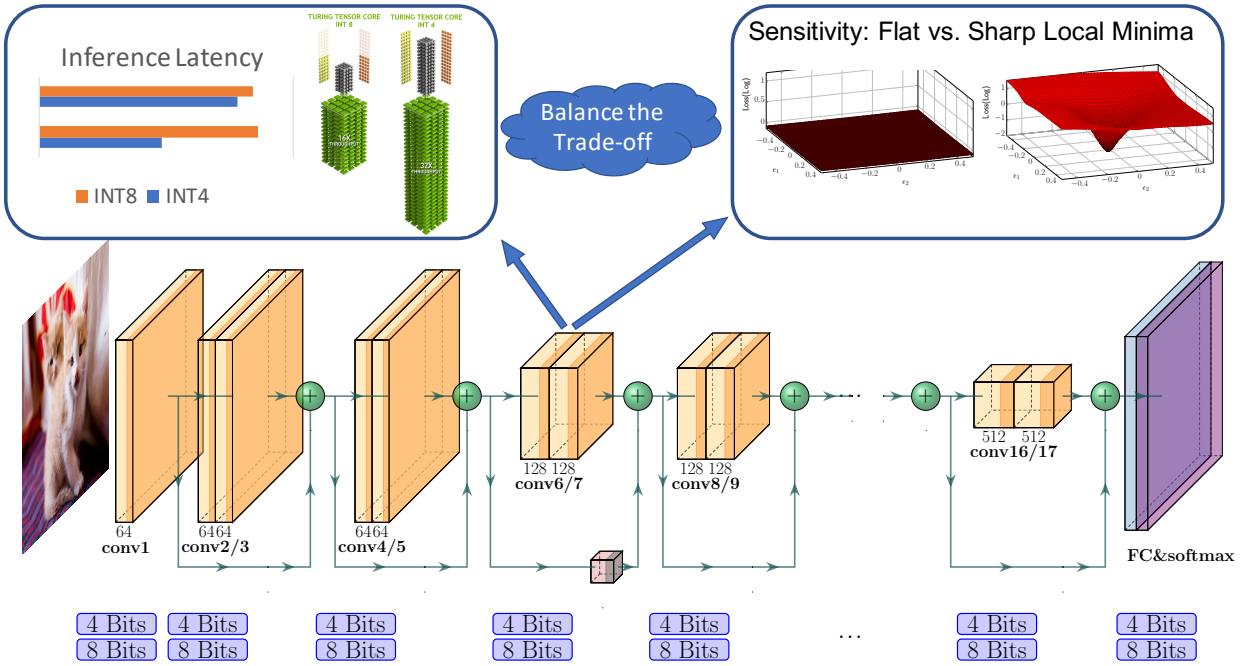

Figure 8: *Illustration of mixed-precision quantization. In mixed-precision quantization the goal is to keep sensitive and efficient layers in higher precision, and only apply low-precision quantization to insensitive and inefficient layers. The efficiency metric is hardware dependant, and it could be latency or energy consumption.*

and 116× more area efficient as compared to FP32 addition [96].

Notable integer-only quantization works include [151], which fuses Batch Normalization into the previous convolution layer, and [112], which proposes an integer-only computation method for residual networks with batch normalization. However, both methods are limited to ReLU activation. The recent work of [130] addresses this limitation by approximating GELU [93], Softmax, and Layer Normalization [6] with integer arithmetic and further extends integer-only quantization to Transformer [235] architectures.

Dyadic quantization is another class of integer-only quantization, where all the scaling is performed with dyadic numbers, which are rational numbers with integer values in their numerator and a power of 2 in the denominator [259]. This results in a computational graph that only requires integer addition, multiplication, bit shifting, but no integer division. Importantly, in this approach, all the additions (e.g. residual connections) are enforced to have the same dyadic scale, which can make the addition logic simpler with higher efficiency.

Summary (Simulated vs Integer-only Quantization). In general integer-only and dyadic quantization are more desirable as compared to simulated/fake quantization. This is because integer-only uses lower precision logic for the arithmetic, whereas simulated quantization uses floating point logic to perform the operations. However, this does not mean that fake quantization is never useful. In fact, fake quantization methods can be beneficial for problems that are bandwidth-bound rather than compute-bound, such as in recommendation systems [180]. For these tasks, the bottleneck is the memory footprint and the cost of loading parameters from memory. Therefore, performing fake quantization can be acceptable for these cases.

B. Mixed-Precision Quantization

It is easy to see that the hardware performance improves as we use lower precision quantization. However, uniformly quantizing a model to ultra low-precision can cause significant accuracy degradation. It is possible to address this with mixed-precision quantization [51, 81, 101, 182, 193, 204, 231, 238, 241, 255, 277]. In this approach, each layer is quantized with different bit precision, as illustrated in Figure 8. One challenge with this approach is that the search space for choosing this bit setting is exponential in the number of layers. Different approaches have been proposed to address this huge search space.

Selecting this mixed-precision for each layer is essentially a searching problem, and many different methods

have been proposed for it. The recent work of [238] proposed a reinforcement learning (RL) based method to determine automatically the quantization policy, and the authors used a hardware simulator to take the hardware accelerator's feedback in the RL agent feedback. The paper [246] formulated the mixed-precision configuration searching problem as a Neural Architecture Search (NAS) problem and used the Differentiable NAS (DNAS) method to efficiently explore the search space. One disadvantage of these exploration-based methods [238, 246] is that they often require large computational resources, and their performance is typically sensitive to hyperparameters and even initialization.

Another class of mixed-precision methods uses periodic function regularization to train mixed-precision models by automatically distinguishing different layers and their varying importance with respect to accuracy while learning their respective bitwidths [179].

Different than these exploration and regularization-based approaches, HAWQ [51] introduces an automatic way to find the mixed-precision settings based on second-order sensitivity of the model. It was theoretically shown that the trace of the second-order operator (i.e., the Hessian) can be used to measure the sensitivity of a layer to quantization [50], similar to results for pruning in the seminal work of Optimal Brain Damage [137]. In HAWQv2, this method was extended to mixed-precision activation quantization [50], and was shown to be more than 100x faster than RL based mixed-precision methods [238]. Recently, in HAWQv3, an integer-only, hardware-aware quantization was introduced [259] that proposed a fast Integer Linear Programming method to find the optimal bit precision for a given application-specific constraint (e.g., model size or latency). This work also addressed the common question about hardware efficiency of mixed-precision quantization by directly deploying them on T4 GPUs, showing up to 50% speed up with mixed-precision (INT4/INT8) quantization as compared to INT8 quantization.

Summary (Mixed-precision Quantization). Mixed-precision quantization has proved to be an effective and hardware-efficient method for low-precision quantization of different NN models. In this approach, the layers of a NN are grouped into sensitive/insensitive to quantization, and higher/lower bits are used for each layer. As such, one can minimize accuracy degradation and still benefit from reduced memory footprint and faster speed up with low precision quantization. Recent work [259] has also shown that this approach is hardware-efficient as mixed-precision is only used across operations/layers.

C. Hardware Aware Quantization

One of the goals of quantization is to improve the inference latency. However, not all hardware provide the same speed up after a certain layer/operation is quantized. In fact, the benefits from quantization is hardware-dependant, with many factors such as on-chip memory, bandwidth, and cache hierarchy affecting the quantization speed up.

It is important to consider this fact for achieving optimal benefits through hardware-aware quantization [86, 90, 238, 242, 246, 248, 257, 259]. In particular, the work [238] uses a reinforcement learning agent to determine the hardware-aware mixed-precision setting for quantization, based on a look-up table of latency with respect to different layers with different bitwidth. However, this approach uses simulated hardware latency. To address this the recent work of [259] directly deploys quantized operations in hardware, and measures the actual deployment latency of each layer for different quantization bit precisions.

D. Distillation-Assisted Quantization

An interesting line of work in quantization is to incorporate model distillation to boost quantization accuracy [124, 173, 189, 259]. Model distillation [3, 94, 148, 173, 189, 200, 260, 262, 280] is a method in which a large model with higher accuracy is used as a teacher to help the training of a compact student model. During the training of the student model, instead of using just the ground-truth class labels, model distillation proposes to leverage the soft probabilities produced by the teacher, which may contain more information of the input. That is the overall loss function incorporates both the student loss and the distillation loss, which is typically formulated as follows:

$$\mathcal{L} = \alpha \mathcal{H}(y, \sigma(z_s)) + \beta \mathcal{H}(\sigma(z_t, T), \sigma(z_s, T)) \qquad (10)$$

In Eq. 10, α and β are weighting coefficients to tune the amount of loss from the student model and the distillation loss, y is the ground-truth class label, $\mathcal{H}$ is the cross-entropy loss function, z_s/z_t are logits generated by the student/teacher model, σ is the softmax function, and T is its temperature defined as follows:

$$p_i = \frac{\exp \frac{z_i}{T}}{\sum_j \exp \frac{z_j}{T}} \qquad (11)$$

Previous methods of knowledge distillation focus on exploring different knowledge sources. [94, 148, 187] use logits (the soft probabilities) as the source of knowledge, while [3, 200, 261] try to leverage the knowledge

from intermediate layers. The choices of teacher models are also well studied, where [227, 265] use multiple teacher models to jointly supervise the student model, while [43, 269] apply self-distillation without an extra teacher model.

E. Extreme Quantization

Binarization, where the quantized values are constrained to a 1-bit representation, thereby drastically reducing the memory requirement by 32×, is the most extreme quantization method. Besides the memory advantages, binary (1-bit) and ternary (2-bit) operations can often be computed efficiently with bit-wise arithmetic and can achieve significant acceleration over higher precisions, such as FP32 and INT8. For instance, the peak binary arithmetic on NVIDIA V100 GPUs is 8x higher than INT8. However, a naive binarization method would lead to significant accuracy degradation. As such, there is a large body of work that has proposed different solutions to address this [18, 25, 47, 52, 77, 82, 91, 92, 118, 120, 122, 127, 129, 133, 139, 147, 152, 157, 190, 192, 210, 241, 243, 252, 254, 279, 281].

An important work here is BinaryConnect [42] which constrains the weights to either +1 or -1. In this approach, the weights are kept as real values and are only binarized during the forward and backward passes to simulate the binarization effect. During the forward pass, the real-value weights are converted into +1 or -1 based on the sign function. Then the network can be trained using the standard training method with STE to propagate the gradients through the non-differentiable sign function. Binarized NN [106] (BNN) extends this idea by binarizing the activations as well as the weights. Jointly binarizing weights and activations has the additional benefit of improved latency, since the costly floating-point matrix multiplications can be replaced with lightweight XNOR operations followed by bit-counting. Another interesting work is Binary Weight Network (BWN) and XNOR-Net proposed in [45], which achieve higher accuracy by incorporating a scaling factor to the weights and using $+\alpha$ or $-\alpha$ instead of +1 or -1. Here, α is the scaling factor chosen to minimize the distance between the real-valued weights and the resulting binarized weights. In other words, a real-valued weight matrix W can be formulated as $W \approx \alpha B$, where B is a binary weight matrix that satisfies the following optimization problem:

$$\alpha, B = \text{argmin}\|W - \alpha B\|^2. \tag{12}$$

Furthermore, inspired by the observation that many learned weights are close to zero, there have been attempts to ternarize network by constraining the weights/activations with ternary values, e.g., +1, 0 and -1, thereby explicitly permitting the quantized values to be zero [143, 156]. Ternarization also drastically reduces the inference latency by eliminating the costly matrix multiplications as binarization does. Later, Ternary-Binary Network (TBN) [236] shows that combining binary network weights and ternary activations can achieve an optimal tradeoff between the accuracy and computational efficiency.

Since the naive binarization and ternarization methods generally result in severe accuracy degradation, especially for complex tasks such as ImageNet classification, a number of solutions have been proposed to reduce the accuracy degradation in extreme quantization. The work of [191] broadly categorizes these solutions into three branches. Here, we briefly discuss each branch, and we refer the interested readers to [191] for more details.

a) Quantization Error Minimization: The first branch of solutions aims to minimize the quantization error, i.e., the gap between the real values and the quantized values [19, 34, 62, 102, 149, 155, 160, 165, 174, 211, 240]. Instead of using a single binary matrix to represent real-value weights/activations, HORQ [149] and ABC-Net [155] use a linear combination of multiple binary matrices, i.e., $W \approx \alpha_1 B_1 + \cdots + \alpha_M B_M$, to reduce the quantization error. Inspired by the fact that binarizing the activations reduces their representational capability for the succeeding convolution block, [174] and [34] show that binarization of wider networks (i.e., networks with larger number of filters) can achieve a good trade-off between the accuracy and the model size.

b) Improved Loss function: Another branch of works focuses on the choice of loss function [48, 97, 98, 243, 275]. Important works here are loss-aware binarization and ternarization [97, 98] that directly minimize the loss with respect to the binarized/ternatized weights. This is different from other approaches that only approximate the weights and do not consider the final loss. Knowledge distillation from full-precision teacher models has also been shown as a promising method to recover the accuracy degradation after binarization/ternarization [33, 173, 189, 252].

c) Improved Training Method: Another interesting branch of work aims for better training methods for binary/ternary models [5, 20, 44, 73, 157, 160, 276, 279]. A number of efforts point out the limitation of STE in backpropagating gradients through the sign function: STE only propagate the gradients for the weights and/or activations that are in the range of [-1, 1]. To address this,

BNN+ [44] introduces a continuous approximation for the derivative of the sign function, while [192, 253, 264] replace the sign function with smooth, differentiable functions that gradually sharpens and approaches the sign function. Bi-Real Net [160] introduces identity shortcuts connecting activations to activations in consecutive blocks, through which 32-bit activations can be propagated. While most research focuses on reducing the inference time latency, DoReFa-Net [276] quantizes the gradients in addition to the weights and activations, in order to accelerate the training as well.

Extreme quantization has been successful in drastically reducing the inference/training latency as well as the model size for many CNN models on computer vision tasks. Recently, there have been attempts to extend this idea to Natural Language Processing (NLP) tasks [7, 119, 270]. Considering the prohibitive model size and inference latency of state-of-the-art NLP models (e.g., BERT [46], RoBERTa [159], and the GPT family [17, 194, 195]) that are pre-trained on a large amount of unlabeled data, extreme quantization is emerging as a powerful tool for bringing NLP inference tasks to the edge.

Summary (Extreme Quantization). Extreme low-bit precision quantization is a very promising line of research. However, existing methods often incur high accuracy degradation as compared to baseline, unless very extensive tuning and hyperparameter search is performed. But this accuracy degradation may be acceptable for less critical applications.

F. Vector Quantization

As discussed in Section II, quantization has not been invented in machine learning, but has been widely studied in the past century in information theory, and particularly in digital signal processing field as a compression tool. However, the main difference between quantization methods for machine learning is that fundamentally we are not interested to compress the signal with minimum change/error as compared to the original signal. Instead, the goal is to find a reduced-precision representation that results in as small loss as possible. As such, it is completely acceptable if the quantized weights/activations are far away from the non-quantized ones.

Having said that, there are a lot of interesting ideas in the classical quantization methods in DSP that have been applied to NN quantization, and in particular vector quantization [9]. In particular, the work of [1, 30, 74, 83, 116, 166, 176, 184, 248] clusters the weights into different groups and use the centroid of each group as quantized values during inference. As shown in Eq. 13, i is the index of weights in a tensor, $c_1, ..., c_k$ are the k centroids found by the clustering, and c_j is the corresponding centroid to w_i. After clustering, weight w_i will have a cluster index j related to c_j in the codebook (look-up table).

$$\min_{c_1,...,c_k} \sum_i \|w_i - c_j\|^2 \tag{13}$$

It has been found that using a k-means clustering is sufficient to reduce the model size up to $8\times$ without significant accuracy degradation [74]. In addition to that, jointly applying k-means based vector quantization with pruning and Huffman coding can further reduce the model size [83].

Product quantization [74, 219, 248] is an extension of vector quantization, where the weight matrix is divided into submatrices and vector quantization is applied to each submatrix. Besides basic product quantization method, more fine-grained usage of clustering can further improve the accuracy. For example, in [74] the residuals after k-means product quantization are further recursively quantized. And in [184], the authors apply more clusters for more important quantization ranges to better preserve the information.

V. Quantization and Hardware Processors

We have said that quantization not only reduces the model size, but it also enables faster speed and requires less power, in particular for hardware that has low-precision logic. As such, quantization has been particularly crucial for edge deployment in IoT and mobile applications. Edge devices often have tight resource constraints including compute, memory, and importantly power budget. These are often too costly to meet for many deep NN models. In addition, many edge processors do not have any support floating point operations, especially in micro-controllers.

Here, we briefly discuss different hardware platforms in the context of quantization. ARM Cortex-M is a group of 32-bit RISC ARM processor cores that are designed for low-cost and power-efficient embedded devices. For instance, the STM32 family are the microcontrollers based on the ARM Cortex-M cores that are also used for NN inference at the edge. Because some of the ARM Cortex-M cores do not include dedicated floating-point units, the models should first be quantized before deployment. CMSIS-NN [134] is a library from ARM that helps quantizing and deploying NN models onto the ARM Cortex-M cores. Specifically, the library leverages

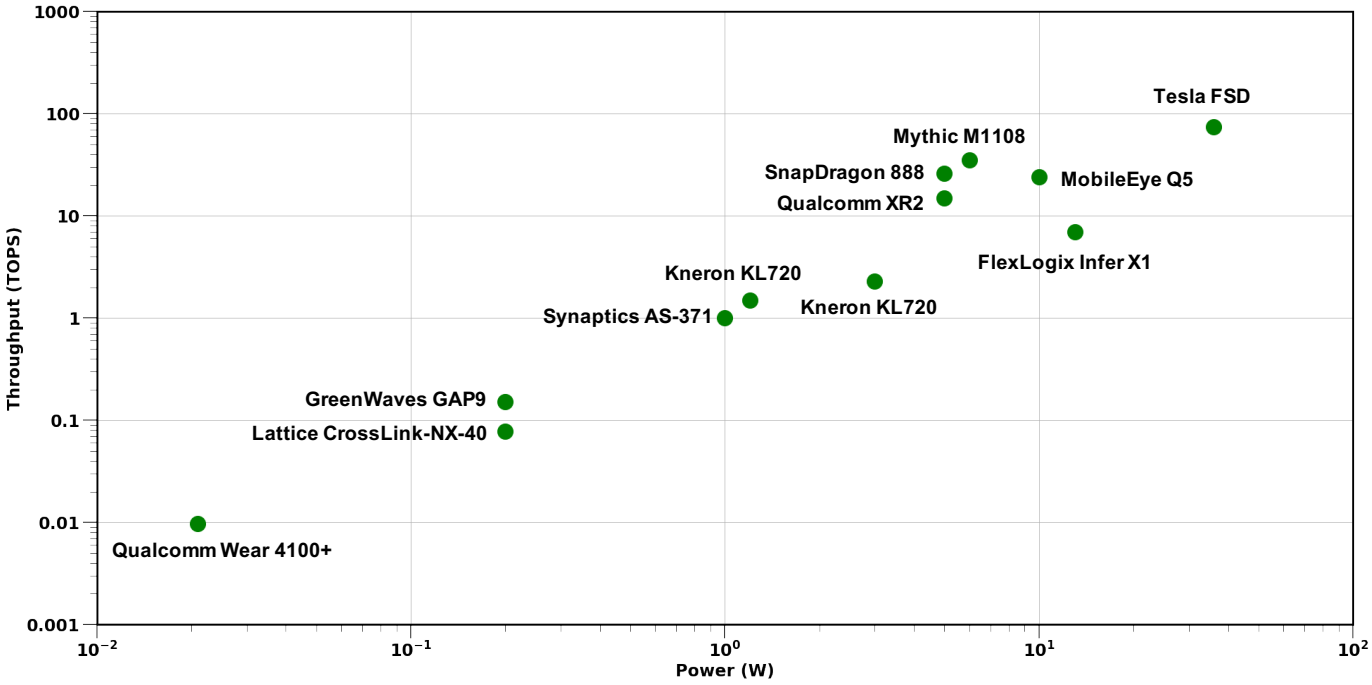

Figure 9: *Throughput comparison of different commercial edge processors for NN inference at the edge.*

fixed-point quantization [112, 151, 259] with power-of-two scaling factors so that quantization and dequantization processes can be carried out efficiently with bit shifting operations. GAP-8 [64], a RISC-V SoC (System on Chip) for edge inference with a dedicated CNN accelerator, is another example of an edge processor that only supports integer arithmetic. While programmable general-purpose processors are widely adopted due to their flexibility, Google Edge TPU, a purpose-built ASIC chip, is another emerging solution for running inference at the edge. Unlike Cloud TPUs that run in Google data centers with a large amount of computing resources, the Edge TPU is designed for small and low-power devices, and thereby it only supports 8-bit arithmetic. NN models must be quantized using either quantization-aware training or post-training quantization of TensorFlow.

Figure 9 plots the throughput of different commercial edge processors that are widely used for NN inference at the edge. In the past few years, there has been a significant improvement in the computing power of the edge processors, and this allows deployment and inference of costly NN models that were previously available only on servers. Quantization, combined with efficient low-precision logic and dedicated deep learning accelerators, has been one important driving force for the evolution of such edge processors.

While quantization is an indispensable technique for a lot of edge processors, it can also bring a remarkable improvement for non-edge processors, e.g., to meet Service Level Agreement (SLA) requirements such as 99th percentile latency. A good example is provided by the recent NVIDIA Turing GPUs, and in particular T4 GPUs, which include the Turing Tensor Cores. Tensor Cores are specialized execution units designed for efficient low-precision matrix multiplications.

VI. Future Directions for Research in Quantization

Here, we briefly discuss several high level challenges and opportunities for future research in quantization. This is broken down into quantization software, hardware and NN architecture co-design, coupled compression methods, and quantized training.

Quantization Software: With current methods, it is straightforward to quantize and deploy different NN models to INT8, without losing accuracy. There are several software packages that can be used to deploy INT8 quantized models (e.g., Nvidia's TensorRT, TVM, etc.), each with good documentation. Furthermore, the implementations are also quite optimal and one can easily observe speed up with quantization. However, the software for lower bit-precision quantization is not widely available, and sometimes it is non-existent. For instance, Nvidia's TensorRT does not currently support sub-INT8

quantization. Moreover, support for INT4 quantization was only recently added to TVM [259]. Recent work has shown that low precision and mixed-precision quantization with INT4/INT8 works in practice [51, 81, 101, 107, 182, 193, 204, 231, 238, 238, 241, 255, 259, 277]. Thus, developing efficient software APIs for lower precision quantization will have an important impact.

Hardware and NN Architecture Co-Design: As discussed above, an important difference between classical work in low-precision quantization and the recent work in machine learning is the fact that NN parameters may have very different quantized values but may still generalize similarly well. For example, with quantization-aware training, we might converge to a different solution, far away from the original solution with single precision parameters, but still get good accuracy. One can take advantage of this degree of freedom and also adapt the NN architecture as it is being quantized. For instance, the recent work of [34] shows that changing the width of the NN architecture could reduce/remove generalization gap after quantization. One line of future work is to adapt jointly other architecture parameters, such as depth or individual kernels, as the model is being quantized. Another line of future work is to extend this co-design to hardware architecture. This may be particularly useful for FPGA deployment, as one can explore many different possible hardware configurations (such as different micro-architectures of multiply-accumulate elements), and then couple this with the NN architecture and quantization co-design.

Coupled Compression Methods: As discussed above, quantization is only one of the methods for efficient deployment of NNs. Other methods include efficient NN architecture design, co-design of hardware and NN architecture, pruning, and knowledge distillation. Quantization can be coupled with these other approaches. However, there is currently very little work exploring what are the optimal combinations of these methods. For instance, pruning and quantization can be applied together to a model to reduce its overhead [86], and it is important to understand the best combination of structured/unstructured pruning and quantization. Similarly, another future direction is to study the coupling between these methods and other approaches described above.

Quantized Training: Perhaps the most important use of quantization has been to accelerate NN training with half-precision [41, 72, 78, 171]. This has enabled the use of much faster and more power-efficient reduced-precision logic for training. However, it has been very difficult to push this further down to INT8 precision training. While several interesting works exist in this area [10, 26, 121, 135, 169], the proposed methods often require a lot of hyperparameter tuning, or they only work for a few NN models on relatively easy learning tasks. The basic problem is that, with INT8 precision, the training can become unstable and diverge. Addressing this challenge can have a high impact on several applications, especially for training at the edge.

VII. Summary and Conclusions

As soon as abstract mathematical computations were adapted to computation on digital computers, the problem of efficient representation, manipulation, and communication of the numerical values in those computations arose. Strongly related to the problem of numerical representation is the problem of quantization: in what manner should a set of continuous real-valued numbers be distributed over a fixed discrete set of numbers to minimize the number of bits required and also to maximize the accuracy of the attendant computations? While these problems are as old as computer science, these problems are especially relevant to the design of efficient NN models. There are several reasons for this. First, NNs are computationally intensive. So, the efficient representation of numerical values is particularly important. Second, most current NN models are heavily over-parameterized. So, there is ample opportunity for reducing the bit precision without impacting accuracy. Third, the layered structure of NN models offers an additional dimension to explore. Thus, different layers in the NN have different impact on the loss function, and this motivates interesting approaches such mixed-precision quantization.

Moving from floating-point representations to low-precision fixed integer values represented in eight/four bits or less holds the potential to reduce the memory footprint and latency. [154] shows that INT8 inference of popular computer vision models, including ResNet50 [87], VGG-19 [216], and inceptionV3 [222] using TVM [32] quantization library, can achieve 3.89×, 3.32×, and 5.02× speedup on NVIDIA GTX 1080, respectively. [206] further shows that INT4 inference of ResNet50 could bring an additional 50-60% speedup on NVIDIA T4 and RTX, compared to its INT8 counterpart, emphasizing the importance of using lower-bit precision to maximize efficiency. Recently, [259] leverages mix-precision quantization to achieve 23% speedup for ResNet50, as compared to INT8 inference without accuracy degradation, and [130] extends INT8-only inference to BERT model to enable up to 4.0× faster inference than FP32.

While the aforementioned works focus on acceleration on GPUs, [113] also obtained 2.35× and 1.40× latency speedup on Intel Cascade Lake CPU and Raspberry Pi4 (which are both non-GPU architectures), respectively, through INT8 quantization of various computer vision models. As a result, as our bibliography attests, the problem of quantization in NN models has been a highly active research area.

In this work, we have tried to bring some conceptual structure to these very diverse efforts. We began with a discussion of topics common to many applications of quantization, such as uniform, non-uniform, symmetric, asymmetric, static, and dynamic quantization. We then considered quantization issues that are more unique to the quantization of NNs. These include layerwise, groupwise, channelwise, and sub-channelwise quantization. We further considered the inter-relationship between training and quantization, and we discussed the advantages and disadvantages of quantization-aware training as compared to post-training quantization. Further nuancing the discussion of the relationship between quantization and training is the issue of the availability of data. The extreme case of this is one in which the data used in training are, due to a variety of sensible reasons such as privacy, no longer available. This motivates the problem of zero-shot quantization.

As we are particularly concerned about efficient NNs targeted for edge-deployment, we considered problems that are unique to this environment. These include quantization techniques that result in parameters represented by fewer than 8 bits, perhaps as low as binary values. We also considered the problem of integer-only quantization, which enables the deployment of NNs on low-end microprocessors which often lack floating-point units.

With this survey and its organization, we hope to have presented a useful snapshot of the current research in quantization for Neural Networks and to have given an intelligent organization to ease the evaluation of future research in this area.

REFERENCES

[1] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. *arXiv preprint arXiv:1704.00648*, 2017.

[2] Eirikur Agustsson and Lucas Theis. Universally quantized neural compression. *Advances in neural information processing systems*, 2020.

[3] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9163–9171, 2019.

[4] Milad Alizadeh, Arash Behboodi, Mart van Baalen, Christos Louizos, Tijmen Blankevoort, and Max Welling. Gradient l1 regularization for quantization robustness. *arXiv preprint arXiv:2002.07520*, 2020.

[5] Milad Alizadeh, Javier Fernández-Marqués, Nicholas D Lane, and Yarin Gal. An empirical study of binary neural networks' optimisation. In *International Conference on Learning Representations*, 2018.

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[7] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*, 2020.

[8] Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. *arXiv preprint arXiv:1810.00861*, 2018.

[9] Dana Harry Ballard. *An introduction to natural computation*. MIT press, 1999.

[10] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. *Advances in neural information processing systems*, 2018.

[11] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Post-training 4-bit quantization of convolution networks for rapid-deployment. *arXiv preprint arXiv:1810.05723*, 2018.

[12] 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.

[13] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[14] William Ralph Bennett. Spectra of quantized sig-

nals. *The Bell System Technical Journal*, 27(3):446–472, 1948.

[15] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 696–697, 2020.

[16] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

[17] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[18] Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. High-capacity expert binary networks. *International Conference on Learning Representations*, 2021.

[19] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019.

[20] Adrian Bulat, Georgios Tzimiropoulos, Jean Kossaifi, and Maja Pantic. Improved training of binary networks for human pose estimation and image recognition. *arXiv preprint arXiv:1904.05868*, 2019.

[21] Aydin Buluc and John R Gilbert. Challenges and advances in parallel sparse matrix-matrix multiplication. In *2008 37th International Conference on Parallel Processing*, pages 503–510. IEEE, 2008.

[22] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[23] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[24] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020.

[25] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5918–5926, 2017.

[26] Léopold Cambier, Anahita Bhiwandiwalla, Ting Gong, Mehran Nekuii, Oguz H Elibol, and Hanlin Tang. Shifted and squeezed 8-bit floating point format for low-precision training of deep neural networks. *arXiv preprint arXiv:2001.05674*, 2020.

[27] Rishidev Chaudhuri and Ila Fiete. Computational principles of memory. *Nature neuroscience*, 19(3):394, 2016.

[28] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3514–3522, 2019.

[29] Jianfei Chen, Yu Gai, Zhewei Yao, Michael W Mahoney, and Joseph E Gonzalez. A statistical framework for low-bitwidth training of deep neural networks. *arXiv preprint arXiv:2010.14298*, 2020.

[30] Kuilin Chen and Chi-Guhn Lee. Incremental few-shot learning via vector quantization in deep embedded space. In *International Conference on Learning Representations*, 2021.

[31] Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[32] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. TVM: An automated end-to-end optimizing compiler for deep learning. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 578–594, 2018.

[33] Xiuyi Chen, Guangcan Liu, Jing Shi, Jiaming Xu, and Bo Xu. Distilled binary neural network for monaural speech separation. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[34] Ting-Wu Chin, Pierce I-Jen Chuang, Vikas Chandra, and Diana Marculescu. One weight bitwidth to rule them all. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[35] Brian Chmiel, Liad Ben-Uri, Moran Shkolnik, Elad Hoffer, Ron Banner, and Daniel Soudry. Neural

gradients are near-lognormal: improved quantized and sparse training. In *International Conference on Learning Representations*, 2021.

[36] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

[37] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 710–711, 2020.

[38] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543*, 2016.

[39] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Learning low precision deep neural networks through regularization. *arXiv preprint arXiv:1809.00095*, 2, 2018.

[40] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCV Workshops*, pages 3009–3018, 2019.

[41] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*, 2014.

[42] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[43] Elliot J Crowley, Gavin Gray, and Amos J Storkey. Moonshine: Distilling with cheap convolutions. In *NeurIPS*, pages 2893–2903, 2018.

[44] Sajad Darabi, Mouloud Belbahri, Matthieu Courbariaux, and Vahid Partovi Nia. Bnn+: Improved binary network training. 2018.

[45] Lei Deng, Peng Jiao, Jing Pei, Zhenzhi Wu, and Guoqi Li. Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Networks*, 100:49–58, 2018.

[46] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[47] James Diffenderfer and Bhavya Kailkhura. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. In *International Conference on Learning Representations*, 2021.

[48] Ruizhou Ding, Ting-Wu Chin, Zeye Liu, and Diana Marculescu. Regularizing activation distribution for training binarized deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11408–11417, 2019.

[49] Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *arXiv preprint arXiv:1705.07565*, 2017.

[50] Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 2020.

[51] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019.

[52] Yueqi Duan, Jiwen Lu, Ziwei Wang, Jianjiang Feng, and Jie Zhou. Learning deep binary descriptor with multi-quantization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1183–1192, 2017.

[53] JG Dunn. The performance of a class of n dimensional quantizers for a gaussian source. In *Proc. Columbia Symp. Signal Transmission Processing*, pages 76–81, 1965.

[54] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

[55] William H Equitz. A new vector quantization clustering algorithm. *IEEE transactions on acoustics, speech, and signal processing*, 37(10):1568–1575, 1989.

[56] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

[57] Fartash Faghri, Iman Tabrizian, Ilia Markov, Dan Alistarh, Daniel Roy, and Ali Ramezani-Kebrya. Adaptive gradient quantization for data-parallel sgd. *Advances in neural information processing*

systems, 2020.

[58] A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292–303, 2008.

[59] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Hervé Jégou, and Armand Joulin. Training with quantization noise for extreme model compression. *arXiv e-prints*, pages arXiv–2004, 2020.

[60] Jun Fang, Ali Shafiee, Hamzah Abdel-Aziz, David Thorsley, Georgios Georgiadis, and Joseph Hassoun. Near-lossless post-training quantization of deep neural networks via a piecewise linear approximation. *arXiv preprint arXiv:2002.00104*, 2020.

[61] 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.

[62] Julian Faraone, Nicholas Fraser, Michaela Blott, and Philip HW Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4300–4309, 2018.

[63] Alexander Finkelstein, Uri Almog, and Mark Grobman. Fighting quantization bias with bias. *arXiv preprint arXiv:1906.03193*, 2019.

[64] Eric Flamand, Davide Rossi, Francesco Conti, Igor Loi, Antonio Pullini, Florent Rotenberg, and Luca Benini. Gap-8: A risc-v soc for ai at the edge of the iot. In *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 1–4. IEEE, 2018.

[65] Abram L Friesen and Pedro Domingos. Deep learning as a mixed convex-combinatorial optimization problem. *arXiv preprint arXiv:1710.11573*, 2017.

[66] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[67] AE Gamal, L Hemachandra, Itzhak Shperling, and V Wei. Using simulated annealing to design good codes. *IEEE Transactions on Information Theory*, 33(1):116–123, 1987.

[68] Sahaj Garg, Anirudh Jain, Joe Lou, and Mitchell Nahmias. Confounding tradeoffs for neural network quantization. *arXiv preprint arXiv:2102.06366*, 2021.

[69] Sahaj Garg, Joe Lou, Anirudh Jain, and Mitchell Nahmias. Dynamic precision analog computing for neural networks. *arXiv preprint arXiv:2102.06365*, 2021.

[70] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. SqueezeNext: Hardware-aware neural network design. *Workshop paper in CVPR*, 2018.

[71] Amir Gholami, Michael W Mahoney, and Kurt Keutzer. An integrated approach to neural network design, training, and inference. *Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep*, 2020.

[72] Boris Ginsburg, Sergei Nikolaev, Ahmad Kiswani, Hao Wu, Amir Gholaminejad, Slawomir Kierat, Michael Houston, and Alex Fit-Florea. Tensor processing using low precision format, December 28 2017. US Patent App. 15/624,577.

[73] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4852–4861, 2019.

[74] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[75] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[76] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE transactions on information theory*, 44(6):2325–2383, 1998.

[77] Yiwen Guo, Anbang Yao, Hao Zhao, and Yurong Chen. Network sketching: Exploiting binary structure in deep cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5955–5963, 2017.

[78] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.

[79] Philipp Gysel, Mohammad Motamedi, and Soheil Ghiasi. Hardware-oriented approximation of convolutional neural networks. *arXiv preprint arXiv:1604.03168*, 2016.

[80] Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 29(11):5784–5789, 2018.

[81] Hai Victor Habi, Roy H Jennings, and Arnon Netzer. Hmq: Hardware friendly mixed precision quantization block for cnns. *arXiv preprint arXiv:2007.09952*, 2020.

[82] Kai Han, Yunhe Wang, Yixing Xu, Chunjing Xu, Enhua Wu, and Chang Xu. Training binary neural networks through learning with noisy supervision. In *International Conference on Machine Learning*, pages 4017–4026. PMLR, 2020.

[83] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[84] Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2020.

[85] Babak Hassibi and David G Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993.

[86] Benjamin Hawks, Javier Duarte, Nicholas J Fraser, Alessandro Pappalardo, Nhan Tran, and Yaman Umuroglu. Ps and qs: Quantization-aware pruning for efficient low latency neural network inference. *arXiv preprint arXiv:2102.11289*, 2021.

[87] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[88] Xiangyu He and Jian Cheng. Learning compression from limited unlabeled data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 752–769, 2018.

[89] Xiangyu He, Qinghao Hu, Peisong Wang, and Jian Cheng. Generative zero-shot network quantization. *arXiv preprint arXiv:2101.08430*, 2021.

[90] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[91] Zhezhi He and Deliang Fan. Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11438–11446, 2019.

[92] Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. *Advances in neural information processing systems*, 2019.

[93] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[94] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[95] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554*, 2021.

[96] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[97] Lu Hou and James T Kwok. Loss-aware weight quantization of deep networks. *arXiv preprint arXiv:1802.08635*, 2018.

[98] Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

[99] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobilenetV3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019.

[100] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[101] Peng Hu, Xi Peng, Hongyuan Zhu, Mohamed M Sabry Aly, and Jie Lin. Opq: Compressing deep neural networks with one-shot pruning-quantization. 2021.

[102] Qinghao Hu, Peisong Wang, and Jian Cheng. From hashing to cnns: Training binary weight

networks via hashing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[103] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[104] Qijing Huang, Dequan Wang, Zhen Dong, Yizhao Gao, Yaohui Cai, Tian Li, Bichen Wu, Kurt Keutzer, and John Wawrzynek. Codenet: Efficient deployment of input-adaptive object detection on embedded fpgas. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 206–216, 2021.

[105] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018.

[106] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[107] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518*, 2020.

[108] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[109] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[110] Yani Ioannou, Duncan Robertson, Roberto Cipolla, and Antonio Criminisi. Deep roots: Improving cnn efficiency with hierarchical filter groups. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1231–1240, 2017.

[111] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[112] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[113] Animesh Jain, Shoubhik Bhattacharya, Masahiro Masuda, Vin Sharma, and Yida Wang. Efficient execution of quantized deep learning models: A compiler approach. *arXiv preprint arXiv:2006.10226*, 2020.

[114] Shubham Jain, Swagath Venkataramani, Vijayalakshmi Srinivasan, Jungwook Choi, Kailash Gopalakrishnan, and Leland Chang. Biscaled-dnn: Quantizing long-tailed datastructures with two scale factors for deep neural networks. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2019.

[115] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[116] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[117] Yongkweon Jeon, Baeseong Park, Se Jung Kwon, Byeongwook Kim, Jeongin Yun, and Dongsoo Lee. Biqgemm: matrix multiplication with lookup table for binary-coding-based quantized dnns. *arXiv preprint arXiv:2005.09904*, 2020.

[118] Kai Jia and Martin Rinard. Efficient exact verification of binarized neural networks. *Advances in neural information processing systems*, 2020.

[119] Jing Jin, Cai Liang, Tiancheng Wu, Liqin Zou, and Zhiliang Gan. Kdlsq-bert: A quantized bert combining knowledge distillation with learned step size quantization. *arXiv preprint arXiv:2101.05938*, 2021.

[120] Qing Jin, Linjie Yang, and Zhenyu Liao. Adabits: Neural network quantization with adaptive bit-widths. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2146–2156, 2020.

[121] Jeff Johnson. Rethinking floating point for deep learning. *arXiv preprint arXiv:1811.01721*, 2018.

[122] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 19–28, 2017.

[123] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju

Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4350–4359, 2019.

[124] Prad Kadambi, Karthikeyan Natesan Ramamurthy, and Visar Berisha. Comparing fisher information regularization with distillation for dnn quantization. *Advances in neural information processing systems*, 2020.

[125] PP Kanjilal, PK Dey, and DN Banerjee. Reduced-size neural networks through singular value decomposition and subset selection. *Electronics Letters*, 29(17):1516–1518, 1993.

[126] Mel Win Khaw, Luminita Stevens, and Michael Woodford. Discrete adjustment to a changing environment: Experimental evidence. *Journal of Monetary Economics*, 91:88–103, 2017.

[127] Hyungjun Kim, Kyungsu Kim, Jinseok Kim, and Jae-Joon Kim. Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. *International Conference on Learning Representations*, 2020.

[128] Jangho Kim, KiYoon Yoo, and Nojun Kwak. Position-based scaled gradient for model quantization and sparse training. *Advances in neural information processing systems*, 2020.

[129] Minje Kim and Paris Smaragdis. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*, 2016.

[130] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. *arXiv preprint arXiv:2101.01321*, 2021.

[131] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[132] Andrey Kuzmin, Markus Nagel, Saurabh Pitre, Sandeep Pendyam, Tijmen Blankevoort, and Max Welling. Taxonomy and evaluation of structured compression of convolutional neural networks. *arXiv preprint arXiv:1912.09802*, 2019.

[133] Se Jung Kwon, Dongsoo Lee, Byeongwook Kim, Parichay Kapoor, Baeseong Park, and Gu-Yeon Wei. Structured compression by weight encryption for unstructured pruning and quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1909–1918, 2020.

[134] Liangzhen Lai, Naveen Suda, and Vikas Chandra. CMSIS-NN: Efficient neural network kernels for arm cortex-m cpus. *arXiv preprint arXiv:1801.06601*, 2018.

[135] Hamed F Langroudi, Zachariah Carmichael, David Pastuch, and Dhireesha Kudithipudi. Cheetah: Mixed low-precision hardware & software co-design framework for dnns on the edge. *arXiv preprint arXiv:1908.02386*, 2019.

[136] Kenneth W Latimer, Jacob L Yates, Miriam LR Meister, Alexander C Huk, and Jonathan W Pillow. Single-trial spike trains in parietal cortex reveal discrete steps during decision-making. *Science*, 349(6244):184–187, 2015.

[137] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[138] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer, 2015.

[139] Dongsoo Lee, Se Jung Kwon, Byeongwook Kim, Yongkweon Jeon, Baeseong Park, and Jeongin Yun. Flexor: Trainable fractional quantization. *Advances in neural information processing systems*, 2020.

[140] Jun Haeng Lee, Sangwon Ha, Saerom Choi, Won-Jo Lee, and Seungwon Lee. Quantization for rapid deployment of deep neural networks. *arXiv preprint arXiv:1810.05488*, 2018.

[141] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[142] Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[143] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[144] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[145] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144*, 2019.

[146] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *International Conference on Learning Representations*, 2021.

[147] Yuhang Li, Ruihao Gong, Fengwei Yu, Xin Dong, and Xianglong Liu. Dms: Differentiable dimension search for binary neural networks. *International Conference on Learning Representations*, 2020.

[148] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning from noisy labels with distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1910–1918, 2017.

[149] Zefan Li, Bingbing Ni, Wenjun Zhang, Xiaokang Yang, and Wen Gao. Performance guaranteed network acceleration via high-order residual quantization. In *Proceedings of the IEEE international conference on computer vision*, pages 2584–2592, 2017.

[150] Zhenyu Liao, Romain Couillet, and Michael W Mahoney. Sparse quantized spectral clustering. *International Conference on Learning Representations*, 2021.

[151] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International conference on machine learning*, pages 2849–2858. PMLR, 2016.

[152] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. *Advances in neural information processing systems*, 2020.

[153] Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *IJCAI*, pages 2425–2432, 2018.

[154] Wuwei Lin. Automating optimization of quantized deep learning models on cuda: https://tvm.apache.org/2019/04/29/opt-cuda-quantized, 2019.

[155] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. *arXiv preprint arXiv:1711.11294*, 2017.

[156] Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. *arXiv preprint arXiv:1510.03009*, 2015.

[157] Chunlei Liu, Wenrui Ding, Xin Xia, Baochang Zhang, Jiaxin Gu, Jianzhuang Liu, Rongrong Ji, and David Doermann. Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circulant back propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2019.

[158] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[159] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[160] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. 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)*, pages 722–737, 2018.

[161] Zhi-Gang Liu and Matthew Mattina. Learning low-precision neural networks without straight-through estimator (STE). *arXiv preprint arXiv:1903.01061*, 2019.

[162] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[163] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[164] Franck Mamalet and Christophe Garcia. Simplifying convnets for fast learning. In *International Conference on Artificial Neural Networks*, pages 58–65. Springer, 2012.

[165] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. *arXiv preprint arXiv:2003.11535*, 2020.

[166] Julieta Martinez, Shobhit Zakhmi, Holger H Hoos, and James J Little. Lsq++: Lower running time and higher recall in multi-codebook quantization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 491–506, 2018.

[167] Warren S McCulloch and Walter Pitts. A logical

calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[168] Jeffrey L McKinstry, Steven K Esser, Rathinakumar Appuswamy, Deepika Bablani, John V Arthur, Izzet B Yildiz, and Dharmendra S Modha. Discovering low-precision networks close to full-precision networks for efficient embedded inference. *arXiv preprint arXiv:1809.04191*, 2018.

[169] Naveen Mellempudi, Sudarshan Srinivasan, Dipankar Das, and Bharat Kaul. Mixed precision training with 8-bit floating point. *arXiv preprint arXiv:1905.12334*, 2019.

[170] Eldad Meller, Alexander Finkelstein, Uri Almog, and Mark Grobman. Same, same but different: Recovering neural network quantization error through weight factorization. In *International Conference on Machine Learning*, pages 4486–4495. PMLR, 2019.

[171] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[172] Szymon Migacz. Nvidia 8-bit inference with tensorrt. *GPU Technology Conference*, 2017.

[173] Asit Mishra and Debbie Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *arXiv preprint arXiv:1711.05852*, 2017.

[174] Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. Wrpn: Wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*, 2017.

[175] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

[176] Lopamudra Mukherjee, Sathya N Ravi, Jiming Peng, and Vikas Singh. A biresolution spectral framework for product quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3329–3338, 2018.

[177] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020.

[178] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019.

[179] Maxim Naumov, Utku Diril, Jongsoo Park, Benjamin Ray, Jedrzej Jablonski, and Andrew Tulloch. On periodic functions as regularizers for quantization of neural networks. *arXiv preprint arXiv:1811.09862*, 2018.

[180] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

[181] Renkun Ni, Hong-min Chu, Oscar Castañeda, Ping-yeh Chiang, Christoph Studer, and Tom Goldstein. Wrapnet: Neural net inference with ultra-low-resolution arithmetic. *arXiv preprint arXiv:2007.13242*, 2020.

[182] Lin Ning, Guoyang Chen, Weifeng Zhang, and Xipeng Shen. Simple augmentation goes a long way: {ADRL} for {dnn} quantization. In *International Conference on Learning Representations*, 2021.

[183] BM Oliver, JR Pierce, and Claude E Shannon. The philosophy of pcm. *Proceedings of the IRE*, 36(11):1324–1331, 1948.

[184] Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5456–5464, 2017.

[185] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. Value-aware quantization for training and inference of neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 580–595, 2018.

[186] Sejun Park, Jaeho Lee, Sangwoo Mo, and Jinwoo Shin. Lookahead: a far-sighted alternative of magnitude-based pruning. *arXiv preprint arXiv:2002.04809*, 2020.

[187] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019.

[188] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International*

Conference on Machine Learning, pages 4095–4104. PMLR, 2018.

[189] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[190] Haotong Qin, Zhongang Cai, Mingyuan Zhang, Yifu Ding, Haiyu Zhao, Shuai Yi, Xianglong Liu, and Hao Su. Bipointnet: Binary neural network for point clouds. *International Conference on Learning Representations*, 2021.

[191] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, 2020.

[192] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2250–2259, 2020.

[193] Zhongnan Qu, Zimu Zhou, Yun Cheng, and Lothar Thiele. Adaptive loss-aware quantization for multi-bit networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[194] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[195] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[196] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[197] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7:1, 2017.

[198] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[199] Bernhard Riemann. *Ueber die Darstellbarkeit einer Function durch eine trigonometrische Reihe*, volume 13. Dieterich, 1867.

[200] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[201] Kenneth Rose, Eitan Gurewitz, and Geoffrey Fox. A deterministic annealing approach to clustering. *Pattern Recognition Letters*, 11(9):589–594, 1990.

[202] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[203] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[204] Manuele Rusci, Marco Fariselli, Alessandro Capotondi, and Luca Benini. Leveraging automated mixed-low-precision quantization for tiny edge microcontrollers. In *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning*, pages 296–308. Springer, 2020.

[205] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE, 2013.

[206] Dave Salvator, Hao Wu, Milind Kulkarni, and Niall Emmart. Int4 precision for ai inference: https://developer.nvidia.com/blog/int4-for-ai-inference/, 2019.

[207] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobilenetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[208] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[209] Claude E Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec*, 4(142-163):1, 1959.

[210] Alexander Shekhovtsov, Viktor Yanush, and Boris Flach. Path sample-analytic gradient estimators for stochastic binary networks. *Advances in neural information processing systems*, 2020.

[211] Mingzhu Shen, Xianglong Liu, Ruihao Gong, and Kai Han. Balanced binary neural networks with gated residual. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4197–4201.

IEEE, 2020.

[212] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-BERT: Hessian based ultra low precision quantization of bert. In *AAAI*, pages 8815–8821, 2020.

[213] William Fleetwood Sheppard. On the calculation of the most probable values of frequency-constants, for data arranged according to equidistant division of a scale. *Proceedings of the London Mathematical Society*, 1(1):353–380, 1897.

[214] Sungho Shin, Kyuyeon Hwang, and Wonyong Sung. Fixed-point performance analysis of recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE, 2016.

[215] Moran Shkolnik, Brian Chmiel, Ron Banner, Gil Shomron, Yuri Nahshan, Alex Bronstein, and Uri Weiser. Robust quantization: One model to rule them all. *Advances in neural information processing systems*, 2020.

[216] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[217] S. M. Stigler. *The History of Statistics: The Measurement of Uncertainty before 1900*. Harvard University Press, Cambridge, 1986.

[218] Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. In *International Conference on Learning Representations*, 2021.

[219] Pierre Stock, Armand Joulin, Rémi Gribonval, Benjamin Graham, and Hervé Jégou. And the bit goes down: Revisiting the quantization of neural networks. *arXiv preprint arXiv:1907.05686*, 2019.

[220] John Z Sun, Grace I Wang, Vivek K Goyal, and Lav R Varshney. A framework for bayesian optimality of psychophysical laws. *Journal of Mathematical Psychology*, 56(6):495–501, 2012.

[221] Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of deep neural networks under quantization. *arXiv preprint arXiv:1511.06488*, 2015.

[222] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[223] Shyam A Tailor, Javier Fernandez-Marques, and Nicholas D Lane. Degree-quant: Quantization-aware training for graph neural networks. *International Conference on Learning Representations*, 2021.

[224] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[225] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[226] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[227] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.

[228] James Tee and Desmond P Taylor. Is information in the brain represented in continuous or discrete form? *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 6(3):199–209, 2020.

[229] L.N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[230] Frederick Tung and Greg Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7873–7882, 2018.

[231] Mart van Baalen, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling. Bayesian bits: Unifying quantization and pruning. *Advances in neural information processing systems*, 2020.

[232] Rufin VanRullen and Christof Koch. Is perception discrete or continuous? *Trends in cognitive sciences*, 7(5):207–213, 2003.

[233] Lav R Varshney, Per Jesper Sjöström, and Dmitri B Chklovskii. Optimal information storage in noisy synapses under resource constraints. *Neuron*, 52(3):409–423, 2006.

[234] Lav R Varshney and Kush R Varshney. Decision making with quantized priors leads to discrimination. *Proceedings of the IEEE*, 105(2):241–255,

2016.

[235] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[236] Diwen Wan, Fumin Shen, Li Liu, Fan Zhu, Jie Qin, Ling Shao, and Heng Tao Shen. Tbn: Convolutional neural network with ternary inputs and binary weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 315–332, 2018.

[237] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. *arXiv preprint arXiv:2011.09011*, 2020.

[238] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[239] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 2018.

[240] Peisong Wang, Qinghao Hu, Yifan Zhang, Chunjie Zhang, Yang Liu, and Jian Cheng. Two-step quantization for low-bit neural networks. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 4376–4384, 2018.

[241] Tianzhe Wang, Kuan Wang, Han Cai, Ji Lin, Zhijian Liu, Hanrui Wang, Yujun Lin, and Song Han. Apq: Joint search for network architecture, pruning and quantization policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2078–2087, 2020.

[242] Ying Wang, Yadong Lu, and Tijmen Blankevoort. Differentiable joint pruning and quantization for hardware efficiency. In *European Conference on Computer Vision*, pages 259–277. Springer, 2020.

[243] Ziwei Wang, Jiwen Lu, Chenxin Tao, Jie Zhou, and Qi Tian. Learning channel-wise interactions for binary convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 568–577, 2019.

[244] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[245] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9127–9135, 2018.

[246] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018.

[247] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

[248] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

[249] Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *Advances in Neural Information Processing Systems*, pages 13681–13691, 2019.

[250] Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. Alternating multi-bit quantization for recurrent neural networks. *arXiv preprint arXiv:1802.00150*, 2018.

[251] Shoukai Xu, Haokun Li, Bohan Zhuang, Jing Liu, Jiezhang Cao, Chuangrun Liang, and Mingkui Tan. Generative low-bitwidth data free quantization. In *European Conference on Computer Vision*, pages 1–17. Springer, 2020.

[252] Yinghao Xu, Xin Dong, Yudian Li, and Hao Su. A main/subsidiary network framework for simplifying binary neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7154–7162, 2019.

[253] Zhe Xu and Ray CC Cheung. Accurate and compact convolutional neural networks with trained binarization. *arXiv preprint arXiv:1909.11366*,

2019.

[254] Haichuan Yang, Shupeng Gui, Yuhao Zhu, and Ji Liu. Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2178–2188, 2020.

[255] Huanrui Yang, Lin Duan, Yiran Chen, and Hai Li. Bsq: Exploring bit-level sparsity for mixed-precision neural network quantization. *arXiv preprint arXiv:2102.10462*, 2021.

[256] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7308–7316, 2019.

[257] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300, 2018.

[258] Zhaohui Yang, Yunhe Wang, Kai Han, Chunjing Xu, Chao Xu, Dacheng Tao, and Chang Xu. Searching for low-bit weights in quantized neural networks. *Advances in neural information processing systems*, 2020.

[259] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael W Mahoney, et al. Hawqv3: Dyadic neural network quantization. *arXiv preprint arXiv:2011.10680*, 2020.

[260] Jianming Ye, Shiliang Zhang, and Jingdong Wang. Distillation guided residual learning for binary convolutional neural networks. *arXiv preprint arXiv:2007.05223*, 2020.

[261] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141, 2017.

[262] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

[263] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

[264] Penghang Yin, Shuai Zhang, Jiancheng Lyu, Stanley Osher, Yingyong Qi, and Jack Xin. Blended coarse gradient descent for full quantization of deep neural networks. *Research in the Mathematical Sciences*, 6(1):14, 2019.

[265] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1285–1294, 2017.

[266] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[267] Shixing Yu, Zhewei Yao, Amir Gholami, Zhen Dong, Michael W Mahoney, and Kurt Keutzer. Hessian-aware pruning and optimal neural implant. *arXiv preprint arXiv:2101.08940*, 2021.

[268] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *European conference on computer vision (ECCV)*, 2018.

[269] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019.

[270] Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. Ternarybert: Distillation-aware ultra-low bit bert. *arXiv preprint arXiv:2009.12812*, 2020.

[271] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2019.

[272] Qibin Zhao, Masashi Sugiyama, Longhao Yuan, and Andrzej Cichocki. Learning efficient tensor representations with ring-structured networks. In

ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 8608–8612. IEEE, 2019.

[273] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. *Proceedings of Machine Learning Research*, 2019.

[274] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[275] Aojun Zhou, Anbang Yao, Kuan Wang, and Yurong Chen. Explicit loss-error-aware quantization for low-bit deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9426–9435, 2018.

[276] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[277] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. *arXiv preprint arXiv:1712.01048*, 2017.

[278] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

[279] Shilin Zhu, Xin Dong, and Hao Su. Binary ensemble neural network: More bits per network or more networks per bit? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4923–4932, 2019.

[280] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7920–7928, 2018.

[281] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Structured binary neural networks for accurate image classification and semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 413–422, 2019.

[282] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.