Improving Neural Network Efficiency via Post-training Quantization with Adaptive Floating-Point

Fangxin Liu[1,2], Wenbo Zhao[1,2], Zhezhi He[1], Yanzhi Wang[3], Zongwu Wang[1], Changzhi Dai[4]
Xiaoyao Liang[1], and Li Jiang[1,2]*
1. Shanghai Jiao Tong University 2. Shanghai Qi Zhi Institute
3. Northeastern University 4. DeepBlue Technology (Shanghai) Co., Ltd.
{liufangxin, zhaowenbo, zhezhi.he, ljiang_cs}@sjtu.edu.cn

Abstract

Model quantization has emerged as a mandatory technique for efficient inference with advanced Deep Neural Networks (DNN) by representing model parameters with fewer bits. Nevertheless, prior model quantization either suffers from the inefficient data encoding method thus leading to noncompetitive model compression rate, or requires time-consuming quantization aware training process. In this work, we propose a novel Adaptive Floating-Point (AFP) as a variant of standard IEEE-754 floating-point format, with flexible configuration of exponent and mantissa segments. Leveraging the AFP for model quantization (i.e., encoding the parameter) could significantly enhance the model compression rate without accuracy degradation and model re-training. We also want to highlight that our proposed AFP could effectively eliminate the computationally intensive de-quantization step existing in the dynamic quantization technique adopted by the famous machine learning frameworks (e.g., pytorch, tensorRT, etc.). Moreover, we develop a framework to automatically optimize and choose the adequate AFP configuration for each layer, thus maximizing the compression efficacy. Our experiments indicate that AFP-encoded ResNet-50/MobileNet-v2 only has ∼0.04/0.6% accuracy degradation w.r.t its full-precision counterpart. It outperforms the state-of-the-art works by 1.1% in accuracy using the same bit-width while reducing the energy consumption by 11.2×, which is quite impressive for inference. Code is released at: https://github.com/MXHX7199/ICCV_2021_AFP

1. Introduction

The great success of deep learning depends on the availability of data and the continuous growth of deep learning systems' computing capability. However, deploying DNN on edge devices, such as Internet-of-Things devices or mobile phones, is challenging because of the limited computing and storage resources and the energy budget provided by these edge devices. These result in the large latency, which is the main concern during the inference. For instance, it takes 16 seconds on mobile to complete an image recognition using VGG16 [25], which is intolerable for most applications [18]. Therefore, it is essential to compress the DNN for lower storage requirements and simpler arithmetic operations to improve hardware efficiency.

Among various compression techniques, DNN quantization techniques map the continuous weights into discrete space. Thus weights can be encoded with lower bit-width

*Corresponding author: Li Jiang.

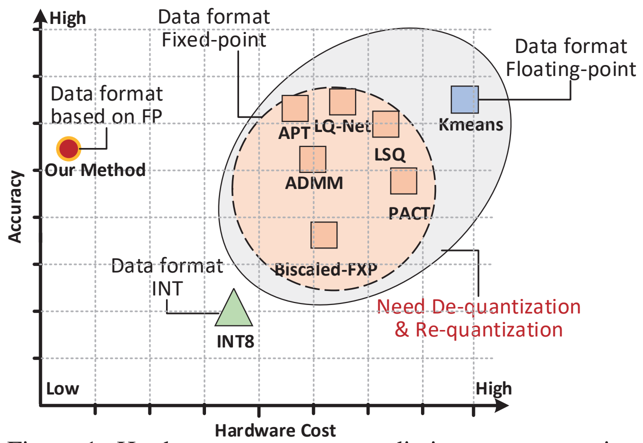


Figure 1: Hardware cost versus prediction accuracy, using various DNN quantization methods. The blue [9] and oranges represent quantization methods [8, 17, 4, 13, 14, 35], using floating-point format or fixed-point format, though with high prediction accuracy, have relatively high computation cost. In contrast, the green represents quantization method [12] using the integer data format, with relatively low accuracy but low-cost. The square indicates that the quantization method requires retraining or fine-tune, while the triangle and ours are not.

binary string, significantly reducing the model size. Following this idea, various quantization methods [8, 4, 12, 9, 17] have been proposed. These quantization methods can be generally separated into non-uniform methods, and uniform methods—the representative prior works should be aligned with those in Fig. 1. The non-uniform quantization method, represented by the deep compression [9], uses k-means to cluster the weights, and the quantized values are denoted as indexes. The uniform quantization method, represented by INT8 [12], maps the weight values into uniformly distributed integers. Meanwhile, power-of-two based quantization (e.g., INQ [35], APT [17]) maps the weight values to the exponential space, then simplify the expensive multiplication operation into shift operation.

In addition, for low-bit quantization, retraining DNN models is needed to mitigate the introduced quantization error (the square in Fig. 1). Since many users are incapable of retraining DNN due to the lack of computing-resource or retraining data, quantization without retraining becomes the most popular compression method in many real-world scenarios [20]. In most non-uniform quantization methods, the operands involved in the calculation belong to 32-bit Floating-Point format (FP32), while uniform quantization is generally in the integer format (INT). Moreover, from the hardware deployment perspective, FP32-based quantization methods have sufficient representation range and precision to make quantization errors small, but they are hardware-intensive [3] compared to INT-based quantization [10].

During inference, the main concerns include latency and energy consumption: low latency is critical for real-time interactions, while low energy consumption can help companies reduce cost in data-centers and improve the endurance of edge devices. Dynamic quantization allows lower energy math operations on hardware platforms and faster inference with only a small drop in accuracy. The key idea of the dynamic quantization is that we will dynamically determine the scaling factor for activation based on the range of data observed at runtime [20]. This means the scaling factor is "tuned" to retain as much information as possible for the activations of each layer. However, most dynamic quantization methods [12, 17, 4, 33, 13, 28] have to perform the de-quantization and re-quantization process to rescale parameters with the aim of ensuring accuracy, as TensorRT does. This adds the complex quantizer after each layer, leading to higher energy consumption and longer latency, thereby diminishing the efficiency improvements of quantization.

As the trade-off of prior quantization methods in terms of data format precision (i.e., quantization accuracy) and hardware efficiency, we develop a floating-point representation variant, named Adaptive Floating-Point (AFP). In contrast to the conventional 32-bit floating-point representation with fixed data format (i.e., the sign bit, exponent bits, and mantissa bits) and bit-widths for each segment, we inherit its data format but make the bit-width of AFP segments configurable. Therefore, to minimize the quantization-caused accuracy degradation and hardware-efficiency, the AFP representation can be optimized w.r.t specific target DNN. Our contribution in this work can be summarized as:

- We propose a novel Adaptive Floating-Point (AFP) representation whose segmentation (i.e., bit-widths of exponent and mantissa part) are fully configurable. Such AFP format is utilized to encode (i.e., quantize) DNN weights and activations for negligible accuracy degradation, while significantly reducing the computation cost. Meanwhile, we fuse dynamic quantization to improve performance by combining de-quantization and re-quantization with AFP format, where all of the calculations are done in AFP without having to convert to FP32, so there is less hardware overhead.

- To leverage the proposed AFP format for DNN quantization, we develop a comprehensive framework to automatically tune the AFP configuration (i.e., bit-widths of exponent and mantissa), using Bayesian optimization as the solver. Given a DNN to be quantized, our framework can provide the optimal setting in terms of the accuracy and computation cost.

- To demonstrate the efficacy of our proposed quantization method and corresponding framework, we conduct comprehensive experiments on the large-scale ImageNet dataset, with popular MobileNet-v2 and ResNet-50. The experiments show that our method outperforms prior ones. To be specific, the decrease in Top-1 accuracy of quantized ResNet-50 is 0.04%, MobileNet is 0.6% on ImageNet with 5-bit on average, exceeding state-of-the-art accuracy.

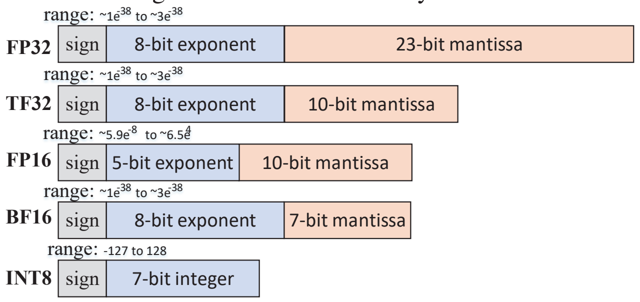


Figure 2: Comparison numeric format with INT8, FP32/16, BFP16 and TF32.

2. Background and Prior Works

2.1. Data Representation

There are five common precision formats in the domain of inference/training of neural networks: INT8 [12], FP32 [15], FP16 [15], BFP16 [16] and TF32 [22]. As depicted in Fig. 2, INT8 represents a signed integer number stored with 8-bit and the others follow the floating-point

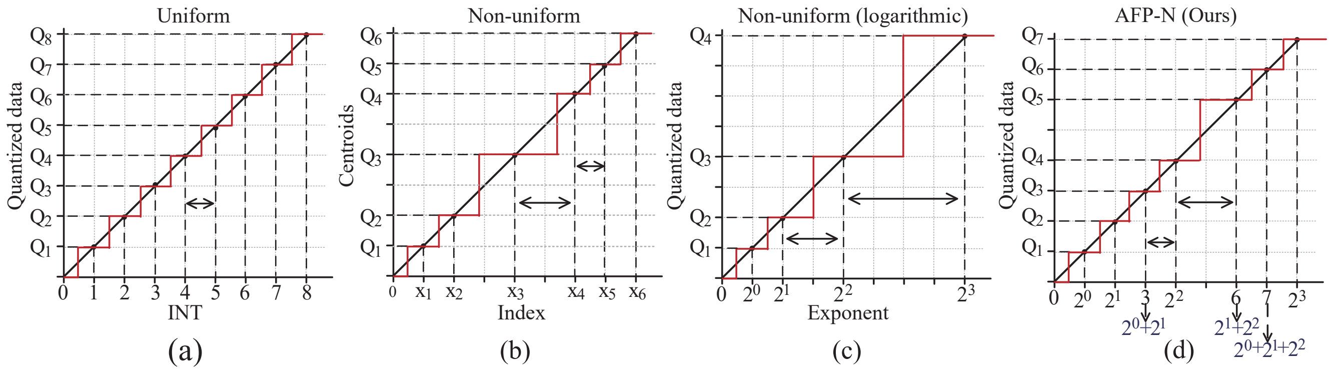


Figure 3: The Comparison of quantization methods: (a) uniform quantization, (b) and (c) are the non-uniform quantizations, (d) describes that AFP eliminates the above errors. (a) (e.g., INT8) rounds the weights up/down to the evenly distributed integers. (b) with the index for variable and adaptive representing resolution and (c) shows the quantized number in logarithmic quantization is distributed exponentially. Both uniform and non-uniform quantizations will cause massive quantization errors with lower bit-width.

format. Floating-point format [15] divides the bit string into three parts: the sign bit S, the exponent bits E whose bit-width is n_{exp} bits and the mantissa bits M whose length is n_{man} bits. Specifically, FP32 format has an 8-bit exponent and 23-bit mantissa, FP16 has a 5-bit exponent and 10-bit mantissa, BF16 [16] has an 8-bit exponent and 7-bit mantissa, and TF32 has an 8-bit exponent and 10-bit mantissa.

Consider the IEEE-754 [15] 32-bit floating-point representation $\boldsymbol{b}_{\text{fp32}} \in \{0,1\}^{32\times 1}$, which can be written as:

$$\boldsymbol{b}_{\text{fp32}} = \{\underbrace{b_{31}}_{\text{Sign}}, \underbrace{b_{30}, ..., b_{23}}_{\text{Exponent}}, \underbrace{b_{22}, ..., b_0}_{\text{Mantissa}}\};\ b_i \in \{0,1\} \tag{1}$$

The value represented by binary string can be mathematically expressed as:

$$g(\boldsymbol{b}_{\text{fp32}}) = \underbrace{(-1)^{b_{31}}}_{\text{Sign}} \times \underbrace{2^{\sum_{i=23}^{30} 2^{i-23}\cdot b_i - B_{\text{bias}}}}_{\text{Exponent}} \times \underbrace{\left(1 + \sum_{i=0}^{22} 2^{i-23}\cdot b_i\right)}_{\text{Mantissa}} \tag{2}$$

where the bias term $k = 127$ is the default setting for IEEE-754 FP32. Note that, the IEEE-754 floating-point ($\boldsymbol{b}_{\text{fp32}}$) can handle several special cases, including Not a Number (NaN), infinity ($\pm\infty$) and etc, which will not be specified here due to the space constraint. Besides, compared with floating-point, integer data format has a smaller hardware cost, as shown in Table 1.

2.2. Quantization

From the perspectives of quantizer adopted, prior works can be categorized in twofold: uniform and non-uniform counterparts.

Uniform Quantization and Encoding. Uniform quantization methods [2, 1, 8, 14, 32] define a range $[\min, \max]$ and quantize weights in the range by first projecting data to the closest discrete level (integer). For example, all the

Table 1: The comparison of energy- and area-cost of adder and multiplier for INT and FP data formats with different bit-width. Blue and orange bar show the relative cost for the energy and area, respectively, which are all normalized to INT8 adder [3].

OP	DataFormat	Bit-Width	Energy(pJ)	RelativeCosts	Area(μm2)	RelativeCosts
Adder	INT	8-bit	0.03	1	36	1
	INT	16-bit	0.05	2	67	2
	INT	32-bit	0.1	3	137	4
	FP	16-bit	0.4	13	1360	38
	FP	32-bit	0.9	30	4184	116
Multiplier	INT	8-bit	0.2	7	282	8
	INT	32-bit	3.1	103	3495	97
	FP	16-bit	1.1	37	1640	46
	FP	32-bit	3.7	123	7700	214

floating-point weights are quantized to an integer in the range $[-128, 127]$ in INT8 quantization [12]. However, there is an inborn shortage of uniform quantization. The distribution of the weights in a convolution layer is analogous to a Gaussian distribution, which means that few have a large absolute value. A majority of them aggregates near "0". This will lead to a large amount of small quantization errors for the weights with small absolute value. These small quantization errors are accumulated and consequently result in a significant accuracy drop [7]. Meanwhile, the quantized value generated by uniform quantization usually corresponds to the INT or fixed-point data format (i.e., INT8 [12], Biscaled-FxP [13], QIL [14], etc).

Non-uniform Quantization and Encoding. Non-uniform quantization methods [9, 35, 19, 17] usually has variable distribution intervals, making quantization more selective for significant data or more extensive dynamic ranges. Deep Compression [9] is a clustering-based quantization that aims to find some "centers" (i.e., the average value) that can represent the majority of the weight values in the original weight distribution. INQ [35] and APT [17] quantize weights in the form of exponents, allowing the distribution intervals to increases exponentially, giving a wide range with a low bit-width. Nevertheless, the above methods are very time-consuming, since they require retraining (or fine-tuning) to restore accuracy. For example, LSQ [14], LQ-Net [33] spent nearly 100 epochs for retraining, while PACT [4] costs 200 epochs to learn threshold parameters for the quantization. In the data format sense, after non-uniform quantization, the computation data still belongs to FP32, which is more expensive than the INT data format.

Limitations of previous works. Compared to uniform quantization in Fig. 3(a), which pursues even distribution intervals, non-uniform quantization is concerned with variable distribution intervals, resulting in smaller quantization errors within a limited bit-width, but introduces the index. In Fig. 3, quantization with 2-bit values can represent the range $[2^0, 2^3]$, but when the exponent is large, coarse-grained distribution intervals appear, and the quantization error is large. Therefore previous works are difficult to perform low bit-wide quantization without retraining. Meanwhile, most previous methods (BFP16 [16] is concerned with the training phase) for inference do not consider that the computation process during inference combined with encoding can only reduce the storage, while arithmetic and activation quantization on the fly still need to be optimized.

3. Approach

In this section, we present a floating-point representation variant, named AFP that realizes lower bit-width with acceptable accuracy loss of the quantized DNN system. AFP is designed to identify those redundant DNN parameters so that it can minimize the accuracy degradation with minimum bit-width. It is worth noting that our work focuses on hardware friendly encoding (i.e., quantization) on DNN with less resource-consuming computing units (adders or even shifters) instead of general multipliers.

3.1. Problem Definition

Consider a DNN consists of L parametric layer (i.e., convolutional/fully-connected layers), given the vectorized input $\boldsymbol{x}$ and corresponding ground-truth label $\boldsymbol{t}$, the neural network inference loss can be described as:

$$\mathcal{L}\big(f(\boldsymbol{x}, \{\mathbf{W}_l^{\text{fp}}\}_{l=1}^{L}); \boldsymbol{t}\big) \tag{3}$$

where $\mathbf{W}_l^{\text{fp}} \in \mathbb{R}$ denotes the full precision (i.e., FP32) weights of l-th layer. To quantize the model weights thus compressing the model size, our objective function can be expressed as:

$$\begin{aligned} \min_{Q(\mathbf{W}^{\text{fp}})} \; & \mathcal{L}\big(\boldsymbol{x}, \{\mathbf{W}_l^{\text{quan}}\}_{l=1}^{L}); \boldsymbol{t}\big) - \mathcal{L}\big(\boldsymbol{x}, \{\mathbf{W}_l^{\text{fp}}\}_{l=1}^{L}); \boldsymbol{t}\big) \\ s.t. \quad & \mathbf{W}_l^{\text{quan}} = Q(\mathbf{W}_l^{\text{fp}}) \end{aligned} \tag{4}$$

where $Q(\cdot)$ is the quantization function, and $\mathbf{W}_l^{\text{quan}}$ is the post-quantization weight tensor of l-th layer. To minimize the difference of inference loss between the full precision model and quantized one (parameterized by $\{\mathbf{W}^{\text{fp}}\}$ and $\{\mathbf{W}^{\text{quan}}\}$ respectively), the quantization function Q is required to be optimized.

3.2. Quantization with Adaptive Floating-Point

To solve the optimization problem defined in Eq. (4), we specify our proposed Adaptive Floating-Point (AFP) data representation first. It is noteworthy that, when applying AFP to represent a conventional 32-bit floating-point value (FP32), the quantization process is applied intrinsically.

Adaptive Floating-Point To encode the DNN parameters in a more hardware-friendly and efficient manner, we propose a novel data format called AFP, which is a variant of conventional floating-point presentation (e.g., IEEE-754 32-bit floating-point as described in Eq. (1) and Eq. (2)). Its binary data encoding $\boldsymbol{b}_{\text{AFP}}$ and represented value can be correspondingly expressed as follows:

$$\boldsymbol{b}_{\text{AFP}} = \{\underbrace{b_{n_{\text{exp}}+n_{\text{man}}}}_{\text{1-bit Sign}}, \underbrace{b_{(n_{\text{exp}}+n_{\text{man}}-1)}, ..., b_{n_{\text{man}}}}_{n_{\text{exp}}\text{-bit Exponent}}, \underbrace{b_{n_{\text{man}}-1}, ..., b_0}_{n_{\text{man}}\text{-bit Mantissa}}\} \tag{5}$$

$$g(\boldsymbol{b}_{\text{AFP}}) = \underbrace{(-1)^{b_{n_{\text{exp}}+n_{\text{man}}}}}_{\text{Sign}} \times \underbrace{2^{\sum_{i=n_{\text{man}}}^{n_{\text{exp}}+n_{\text{man}}-1} 2^{i-n_{\text{man}}} \cdot b_i - k}}_{\text{Exponent}} \times \underbrace{\left(1 + \sum_{i=0}^{n_{\text{man}}} 2^{(i-n_{\text{man}})} \cdot b_i\right)}_{\text{Mantissa}} \tag{6}$$

where our proposed AFP data format shares the similar segmentation ({sign, exponent, mantissa}) of IEEE-754 FP32 [15], but differs from it in the following perspective:

- AFP owns varying bitwidth for exponent and mantissa parts (n_{exp} and n_{man}), where the bit-width are chosen w.r.t the target application (i.e., DNN in our case).
- In contrast to the fixed bias term adopted by the FP32 (i.e., $k = 127$), we make such a bias term a tun-able as well; thus, $g(\boldsymbol{b}_{\text{AFP}})$ can still approximate the FP32 in terms of the absolute value. More details will be given in the following sections.

Approximate the FP32 value with AFP This work aims at finding the optimal AFP configuration (i.e., bit-widths of exponent and mantissa part) of weights to minimize the inference loss of DNN by the quantized parameters. We first propose two assumptions that help us solve the problem.

Assumption 1 *the weights of l-th layer are approximately following Gaussian distribution (i.e.,$\mathbf{W}_l^{FP} \sim \mathcal{N}\left(\mu_l, \sigma_l^2\right)$, where μ_l and σ_l are the calculated mean and standard deviation of the distribution of weight $\mathbf{W}_l^{FP}$.*

Assumption 2 *the smaller the quantization error is (e.g., minimize the KL-divergence between $\mathbf{W}_l^{FP}$ and $\mathbf{W}_l^{AFP}$), the smaller the accuracy loss of the quantized network [24] is.*

Such assumptions are the key to incorporate the quantization based on the data format into DNN inference. Then we introduce KL-divergence to describe the conversion error caused by the misfit of between the full-precision weights and quantized ones [11, 27], the optimization can be written as:

$$\begin{aligned} &\min_{Q_{\text{AFP}}} \sum_{l=1}^{L} \mathcal{D}_{\text{KL}}(\mathbf{W}_l^{\text{FP}} \parallel \mathbf{W}_l^{\text{AFP}}) \\ &where \quad \mathbf{W}_l^{\text{AFP}} = Q_{\text{AFP}}(\mathbf{W}_l^{\text{FP}}; k^l, n_{\text{exp}}^l, n_{\text{man}}^l) \\ &\qquad \forall l = 1, 2, \cdots, L \end{aligned} \tag{7}$$

where $Q(\cdot)$ is the quantization with bit-width parameter of the exponent n_{exp}^l, mantissa n_{man}^l, and flexible exponent bias k^l for each layer.

4. Algorithm Overview

In this section, we propose a quantization methodology with AFP for efficient inference. Then, to further improve the accuracy and computation efficiency, we adaptively select the configuration of quantization (Section 4.1) in the solver using Bayesian Optimization (BO) [29, 26] to remove the redundant bit-width, which can reduce the computation and storage. We support fine-grained choices regarding the bit-width of the exponent and mantissa part.

4.1. Layer-wise Quantization with AFP

The conversion from FP32 to our proposed AFP format (described by Q_{AFP}) can be viewed as a data quantization or approximation process. Given the l-th layer of the network, the weights represented in floating-point data $w_{\text{fp}} \in \mathbf{W}_l^{\text{FP}}$ to be converted to minimize the error introduced within the conversion process.

The problem we inevitably face in this work is how to adaptively choose the quantization bit-width and the allocation of the bit-width, which can be regarded as the problem of quantization parameter selection. BO has been applied to hyperparameter search in machine learning, but is beginning to be explored for NN compression techniques. To combine BO to implement adaptive parameter selection to improve search efficiency, we first need to define the objective and the quantization parameters to be optimized.

As the problem stated in Eq. (7) requires jointly optimizing $n_{\text{exp}}^{1:L}$, $n_{\text{man}}^{1:L}$ and $k^{1:L}$, which can be solved by BO. We minimize Eq. (7) by constructing a probabilistic model to determine the most promising candidate quantization parameters for the next step's evaluation. That is, the candidate quantization parameters is chosen by acquisition functions in each process of iteration of BO and then evaluated. The result of evaluation and parameters is used to update the probabilistic model.

We first determine the three quantities: the exponential bias $k^{1:L}$, the bit-width $n_{\text{exp}}^{1:L}$ determine the quantized exponent $\mathcal{E}$ and the bit-width $n_{\text{man}}^{1:L}$ determine the quantized mantissa $\mathcal{M}$.

1. **Determine k.** The bias k is chosen to allow the maximum value of quantized weights $\mathbf{W}_l^{\text{AFP}}$ and the maximum value of weights $\mathbf{W}_l^{\text{FP}}$ to be consistent, and the range of quantization can cover as much of the distribution of weights as possible.
$$k = \text{round}\left(\log_2(\max|\mathbf{W}_l^{\text{FP}}|)\right) \tag{8}$$
2. **Determine the bit-width of exponent n_{exp}.** The n_{exp} should be determined to enable that the range of exponent part $\mathcal{E}$ can adequately cover the distribution of the weights $\mathbf{W}_l^{\text{FP}}$.
$$\mathcal{E} = 2^k \sum_{i=n_{\text{man}}}^{n_{\text{exp}}+n_{\text{man}}} b_i \tag{9}$$
3. **Determine the bit-width of mantissa n_{man}.** The mantissa is a component of a finite floating-point number, with the radix point immediately following the first digit. Here, we try to optimize the quantization precision and minimize the quantization error by choosing an appropriate exponential offset and treat the mantissa $\mathcal{M}$ as the form of power-of-two.
$$\mathcal{M} = \text{round}\left[\left(\frac{w_{\text{fp}}/2^k}{2^{n_{\text{exp}}}} - 1\right) \cdot 2^{n_{\text{man}}}\right] \tag{10}$$

4.2. Search AFP Configuration

The bit-width determines the perturbation introduced by quantization. With the larger bit-width closer to the original value, the KL-divergence is smaller. Therefore, taking into considerations the hardware cost and the use of lower bit-width quantization, we introduce a penalty term defined in [30, 28] and reformulate it by combining bit-width, which is used to measure the number of bit operations needed by the multiplication of two operands. This considers both the number of operation and the bit-widths of operand, which can describe computation workloads more accurately. Therefore, we redefine the optimization problem, which consists of two components, entropy and cost, formalized as:

$$\begin{aligned} &\min_{\{Q_l^{\text{AFP}}\}_{l=1}^L} \mathcal{L}\left(\{\mathbf{W}_l^{\text{FP}}\}_{l=1}^L, \{\mathbf{W}_l^{\text{AFP}}\}_{l=1}^L\right) \\ &s.t. \ \{\mathbf{W}_l^{\text{AFP}}\}_{l=1}^L = \left\{Q^{\text{AFP}}\left(\{\mathbf{W}_l^{\text{FP}}\}, \{k, n_{\text{exp}}, n_{\text{man}}\}\right)\right\}_{l=1}^L \end{aligned} \tag{11}$$

$$\begin{aligned} \mathcal{L}\left(\mathbf{W}_l^{\text{FP}}, \mathbf{W}_l^{\text{AFP}}\right) &= Entropy \times Cost^{\lambda_l} \\ &= \mathcal{D}_{\text{KL}}(\mathbf{W}_l^{\text{FP}} || \mathbf{W}_l^{\text{AFP}}) \times \mathcal{C}(N_l^q, n_l^{\text{exp}})^{\lambda_l} \end{aligned} \tag{12}$$

where $\mathcal{C}(\cdot)$ denotes the cost of a candidate AFP data format and $\mathcal{D}_{\text{KL}}(\cdot||\cdot)$ represents the KL-divergence that characterizes the distance between two distributions. λ_l is a coefficient of the layer l to balance the entropy and cost term. Here, we limit the search range to less than 8-bit so that the loss of Eq.(12) to 0 will not occur.

$$\mathcal{D}_{\text{KL}}(\mathbf{W}_l^{\text{FP}}, \mathbf{W}_l^{\text{AFP}}) = \sum \mathbf{W}_l^{\text{FP}} \log \frac{\mathbf{W}_l^{\text{FP}}}{\mathbf{W}_l^{\text{AFP}}} \tag{13}$$

Algorithm 1 AFP Quantization Algorithm

Input: the full-precision weight matrix $\{\mathbf{W}_l^{\text{FP}}\}_{l=1}^L$ of a model, number of sampling data n , number of steps N

Output: The quantization parameters $\theta^* = \{k, n_{\text{exp}}, n_{\text{man}}\}_{l=1}^L$, which contains the flexible exponent bias k, the bit-width n_{exp} and n_{man} of exponent and mantissa part for each layer, and the quantized value $\{\mathbf{W}_l^{\text{AFP}}\}_{l=1}^L$

```
 1: Initiate sampling data $\mathcal{D}_n = \{(\theta_i, \mathcal{L}(\theta_i)), i = 1, \cdots, n\}$
 2: Initiate maximum iteration $N$.
 3: Initiate best result $J = +\infty$
 4: while iteration $< N$ and result does not converge do
 5:     $\theta^* \leftarrow \arg\max_{\Theta} \alpha(\Theta, \mathcal{D}_n)$    ▷ Determine the most promising compression hyperparameter $\theta^* = \{k', e', p'\}$ based on the acquisition function $\alpha$[29].
 6:     $N^{q'} \leftarrow 1 + e' + p'$
 7:     $\mathbf{W}_l^{\text{scaled}} \leftarrow \mathbf{W}_l^{\text{FP}} / 2^{k'_l}$    ▷ Scale weights
 8:     $\mathbf{W}_l^{\text{AFP}} \leftarrow Q_l^{\text{AFP}}(\mathbf{W}_l^{\text{scaled}}; e', p')$    ▷ Quantize weight
 9:     $\mathcal{D}_{\text{KL}} \leftarrow \sum_{l=1}^{L} \text{KL}(\mathbf{W}_l^{\text{FP}} || \mathbf{W}_l^{\text{AFP}})$
10:     $\mathcal{C} \leftarrow \mathcal{C}(N^{q'}, e')$    ▷ Eq.(14)
11:     $J_{new} \leftarrow \mathcal{D}_{\text{KL}} \cdot \mathcal{C}^{\lambda_l}$    ▷ Evaluate $\mathcal{L}(\theta^*)$
12:     $\mathcal{D}_{n+1} \leftarrow \{\mathcal{D}_n, (\theta^*, J_{new})\}$    ▷ Augment data
13:     $\mathcal{GP}_{new} \leftarrow \text{Update}(\mathcal{GP}, \mathcal{D}_{n+1})$    ▷ Update Gaussian process model
14:     if $J_{new} < J$ then
15:         $J \leftarrow \text{Update}(J_{new})$
16:     end if
17: end while
```

In addition, we formlate the cost $\mathcal{C}(N_q, n_{\text{exp}})$ in a bit operation manner: multiplying a floating-point number needs $n_{\text{exp}} + (1 + n_{\text{man}})^2$-bit OPs and adding them together in the vector multiplication needs a adder with $2^{n_{\text{exp}}} + n_{\text{man}}$ bit-width. Therefore, we define the cost function as:

$$\mathcal{C}(N_q, n_{\text{exp}}) = n_{\text{exp}} + (1 + n_{\text{man}})^2 + 2^{n_{\text{exp}}} + n_{\text{man}} \quad (14)$$

We model the optimization problem Eq. (12) as a Gaussian process, let $\mathcal{L} \sim \mathcal{GP}(\mu(\theta), K(\theta, \Theta))$. Where θ is a set of the quantization parameters, $\mu(\theta)$ and $K(\theta, \Theta)$ indicates the mean function and kernel of the GP model evaluated at θ, respectively. Given $\Theta = \{\theta_1, \theta_2, \cdots, \theta_n\}$, and the evaluation results $\mathcal{L} = \{\mathcal{L}(\theta_1)_1, \mathcal{L}_2(\theta_2), \cdots, \mathcal{L}_n(\theta_n)\}$. The overview is described in Algorithm 1.

4.3. Overall Inference Process

AFP is a number format that aims to reduce the energy consumption with minimum quantization error, compared to others, by adaptively tuning the configuration of bit-width and changing the process by which the values are quantized. After obtaining AFP-encoded model $\{\mathbf{W}_l^{\text{AFP}}\}_{l=1}^L$, the overall inference process takes the following steps: (1) quantize inputs $\mathbf{X}_l$ with Q_{AFP}; (2) obtain the output $\{\mathbf{Y}_l\}$ by performing the convolution operation with $\mathbf{X}_l^{\text{AFP}}$ and $\mathbf{W}_l^{\text{AFP}}$; (3) when the output is obtained, the de-quantization and re-quantization steps after convolution are conducted dynamically, resulting in an overall AFP-encoded neural network. For the dynamic quantization, we have to dynamically determine the AFP configuration on the fly and encode the activations to AFP, just before doing the computation. The network will thus be more accurate but will also introduce higher energy-consumption. To improve inference efficiency, based on the AFP configuration (n_{exp}^x and n_{man}^x) for each layer is obtained by static quantization (specifically, this is done by record activation distributions of different layers and perform AFP quantization based on these distributions), we simplify the quantizer as following:

- The shared exponent offset k^x of the layer is determined by comparing the exponent part of the output feature map.
- According to the AFP format, the length of the mantissa part $\mathcal{M}^x$ determines the computation's complexity, so we prioritize by reducing the mantissa part. Suppose the truncation position of the mantissa part is i. Then, if '1' exists at $i+1$, the mantissa part performs the carry operation and then truncates the excess part ($\geq i+1$) to enable the length of the mantissa part is n_{man}^x; otherwise, it is truncated directly.
- For the exponent part $\mathcal{E}^x$, after adjusting according to k^x, the part exceeding the bit-width n_{man}^x is truncated directly.

Inspired by calibration of INT8 static quantization in the industry, we use a batch of data to identify the appropriate AFP in which the allocation of bit-width allows adaptive precision adjustment for each layer. Thanks to adaptive format conversion and simplifying quantizer, the parameters and activations in the model can be stored with AFP format representation in N-bit format. This means that our quantization process, which requires only addition and truncation, is much more efficient than other quantization methods.

5. Experiments

In this section, we evaluate our AFP-based DNN quantization method on ImageNet datasets [6].

5.1. Comparison with Competing Methods

Performance with AFP on ImageNet Dataset. Before the ablation studies we first conduct the experiment on a large scale ImageNet dataset with ResNet-50/MobileNet-v2 networks. The experimental results with competing

Table 2: Validation accuracy (top1%) of ResNet-50/MobileNet-v2 on ImageNet using various quantization methods on weights and activations.

ImageNet-ILSVRC 2012								
Quan. scheme	Bit width	First layer	Last layer	Acc. Top-1(%)	Acc. loss Top-1(%)	Quan. type	No retrain	Data format
ResNet-50								
Full precision	32	32	32	76.13	-	-	-	-
INT8 [12]	8	8	8	74.9	-1.5	Uniform	✓	INT
V-Q [23]	7	7	7	75.89	-0.27	Uniform	×	FP
Biscaled-FxP [13]	6	6	6	70.46	-5.67	Non-uni.	✓	INT
ADMM [31]	6	6	6	75.93	-0.2	Non-uni.	×	FP
INQ [35]	5	32	32	74.81	-1.59	Non-uni.	×	FP
Focused-C. [34]	5	5	5	75.86	-1.54	Non-uni.	×	FP
APT [17]	4	32	32	75.95	-0.18	Non-uni.	×	FP
UNIQ [2]	4	4	4	74.84	-1.29	Non-uni.	×	FP
this work(dynamic)	**4.8**	**5**	**5**	**76.09**	**-0.04**	**Non-uni.**	✓	**FP**
this work(dynamic)	**3.9**	**4**	**4**	**75.27**	**-0.86**	**Non-uni.**	✓	**FP**
this work (static)	**4.8**	**5**	**5**	**76.00**	**-0.13**	**Non-uni.**	✓	**FP**
this work (static)	**3.9**	**4**	**4**	**75.11**	**-1.02**	**Non-uni.**	✓	**FP**
MobileNet-v2								
Full precision	32	32	32	71.88	-	-	-	-
INT8 [12]	8	8	8	70.9	-0.9	Uniform	✓	INT
DFQ [21]	8	8	8	71.2	-0.6	Uniform	×	INT
INQ [35]	5	32	32	69.17	-2.71	Non-uni.	×	FP
LQ-Net [33]	5	32	32	70.67	-1.21	Non-uni.	×	FP
Deep-Comp. [9]	5	5	5	70.15	-1.73	Non-uni.	×	FP
HAQ (linear) [28]	5	flex.	flex.	69.45	-2.43	Uniform	×	INT
PACT [5]	5	5	5	68.84	-3.04	Non-uni.	×	FP
HAQ (Kmeans) [28]	4	flex.	flex.	71.37	-0.51	Non-uni.	×	FP
this work(dynamic)	**5.7**	**6**	**6**	**71.59**	**-0.29**	**Non-uni.**	✓	**FP**
this work(dynamic)	**4.8**	**5**	**5**	**71.11**	**-0.77**	**Non-uni.**	✓	**FP**
this work (static)	**5.7**	**6**	**6**	**71.47**	**-0.41**	**Non-uni.**	✓	**FP**
this work (static)	**4.8**	**5**	**5**	**70.91**	**-0.97**	**Non-uni.**	✓	**FP**

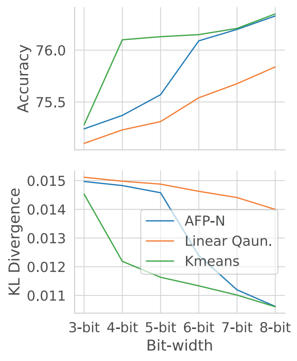
methods are listed in Table 2, together with the methods adopted in related works. The results show that our method can achieve state-of-the-art results with quantized networks. Specifically, for ResNet-50, We quantize the weights and activations to 4.7 and 4.8-bit on average with dynamic quantization, respectively, achieving 0.04% accuracy loss, where the maximum bit-width of all layers is 5-bit. The results of the static quantization use the method described in Section 4.3, and those with dynamic quantization means that the AFP configuration are calculated during inference.

5.2. Ablation studies

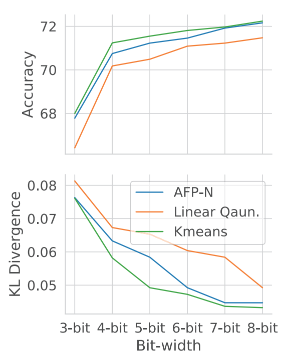
The ablation studies were performed with MobileNet-v2 and ResNet-50 on the ImageNet dataset, where the differences were significant enough to indicate the validity.

KL-divergence with accuracy. We evaluate the effectiveness of our assumption on different quantization methods, which as a validation. We compare the accuracy curve versus the KL-divergence with various bit-width. As shown in Fig. 4, the KL-divergence curve is consistent with the accuracy curve trend under different quantization methods. This proves experimentally that **Assumption 2** is (mentioned in Section 4.2) valid and that we can measure the change in precision by quantifying the resulting loss.

AFP versus other format-based quantization. We chose two classical quantization methods, INT8, a uniform quantization method based on INT format, and Kmeans, a non-uniform quantization based on FP format, for validation. From Fig. 4, we can see that AFP represents weights that cause less loss (in response to the KL-divergence) than the INT-based quantization methods; our approach gives a significant advantage in KL-divergence at all bit-width.



(a) ResNet-50



(b) MobileNet-v2

Figure 4: The accuracy (Top1) and KL-divergence curve versus the bit-with under quantization methods (AFP, INT8 [12], KMeans [9]), for ResNet-50/MobileNet-v2 on ImageNet dataset.

Compared to the quantization method based on FP format, our method has some slight differences with the FP format method at low bit-width (less than 6-bit). However, from a computational perspective, the FP format's quantization method still needs to be converted to the floating-point when participating in the computation. It does not reduce the computational overhead, which cannot be solved by the FP format-based quantization methods but can be effectively solved by our method, which will be discussed later.

Search time versus enumeration. In this section, we argue the necessity for a search algorithm. Performing enumeration is time-consuming and not convincing when finding the optimal solution (i.e., bit-widths of exponent and mantissa part) for each layer. The enumeration consumes 35 iterations to go through all the possibilities less than 8-bit for one layer. Only after that can the solution be chosen. The time to complete an iteration relies on model size, and enumeration is unacceptable if future model size increases. In Fig. 5, it can be seen that the BO algorithm's combination requires only 18 iterations on average for MobileNet-v2 and ResNet-50, reducing the time by 48.8%.

Batch size with accuracy. We compare the accuracy curve versus the batch size with various bit-width. For static quantization, we use a batch of images for calibration as in TensorRT, Pytorch. As shown in Fig. 6, as the batch size increases, the accuracy curve tends to go up, but when the batch size reaches a certain level (batch size = 32), the ac-

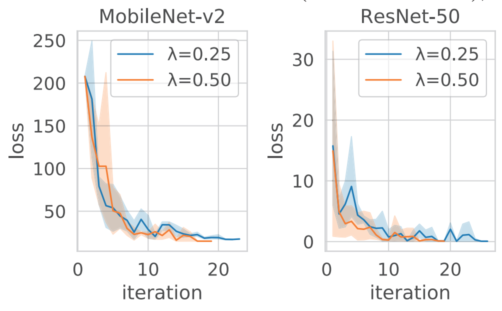


Figure 5: On ImageNet dataset, the loss of AFP-encoded model of MobileNet-v2 and ResNet-50 with bayesian optimization versus λ. Regions in shadow indicates the error band w.r.t 5 trials.

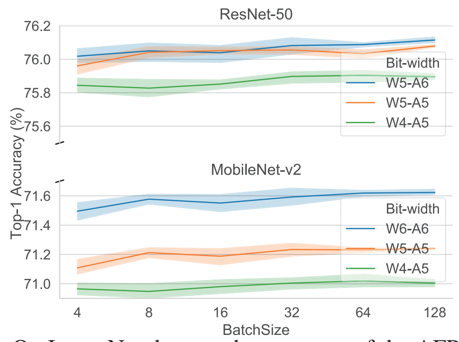


Figure 6: On ImageNet dataset, the accuracy of the AFP-encoded models versus batch size. Regions in shadow indicate the error band w.r.t 5 trials. Here, W5-A5 represents that the rounded bit-width of weights and activations is encoded to 5-bit on average.

curacy stabilizes and does not change anymore.

Hardware cost with the sweet-spot. We limit the search process in conjunction with the hardware cost. As shown in Fig. 5, gradually increasing λ will increase the portion of hardware cost in the loss function, leading to the loss curve converging more quickly to complete the search process to find the sweet points. When $\lambda = 0.50$, the most beneficial is bit-width $N_q = 5$ and $n_{exp} = 3$, the time spent is more significant than $\lambda = 0.25$ with bit-width $N_q = 7$ and $n_{exp} = 3$; With the similar accuracy, the smaller the λ, the smaller the impact of hardware loss on overall loss. It is more difficult to find an optimal solution where the quantization loss is very small, and the alternatives are very close. To exam the influence of the selection of the parameter λ on the search results, we set the parameter as $\lambda \in [0, 1]$. The results reported in Fig. 7 show that the choice of λ plays an essential role in the search. λ determines the share of hardware cost in the overall loss, and the hardware cost is more decisive when the quantization errors are very close. We also report the loss for each bit-width configuration when λ is equal to 0.5, and we can see that the sweet spot is where the bit-width N_q is equal to 5 and exponent part $n_{\text{exp}} = 3$, which matches the searched result.

Hardware efficiency. We modeled the behavior of the processing elements containing multiply-accumulator and quantizer in Verilog and synthesized it using Synopsys Design Compile with 90 nm technology node. In contrast to

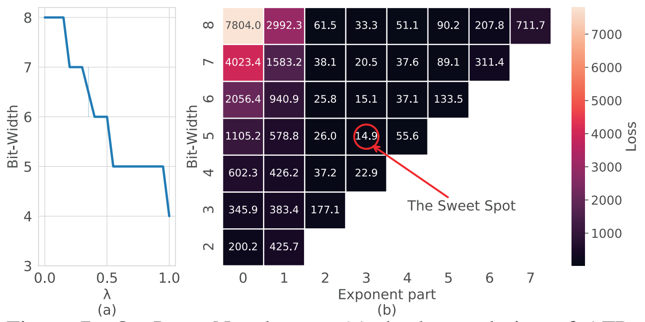


Figure 7: On ImageNet dataset, (a) the best solution of AFP-encoded model of MobileNet-v2 under our quantization with bayesian optimization algorithm versus λ. (b) Losses for each bit-width configuration with λ equal to 0.5.

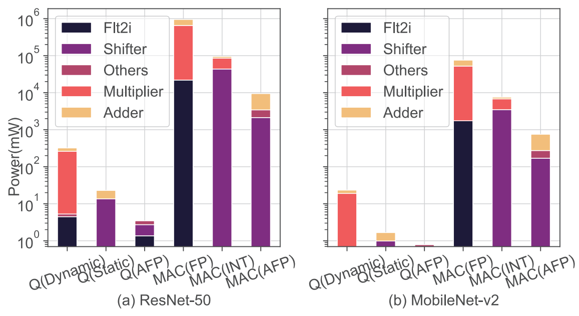


Figure 8: The energy breakdown of (a) ResNet-50 and (b) MobileNet-v2 for MAC and quantization (Q) operation.

the quantization based on FP format, which requires the conversion of bit-word stored in bit-width (e.g., index, etc.) to the floating-point value before computation. AFP directly "multiplies" two exponents composing the weight with the input in parallel by shift operations, whose results are added to the output. AFP is hardware friendly because the expensive multiplication operation is replaced by the simple bit shift and add operations. We also compare the quantization operation, processing each layer's activations during the inference, where the general quantization scheme requires two steps, dequantization and re-quantization, which consume much energy. Simplifying this process by calibration has become the mainstream, as INT8 inference done in Pytorch and TensorRT. Compared to these, our AFP largely keeps the NN accuracy and saves power through low bit-width integer addition, shifting, and some logic circuits. Besides, the impletation of MAC and quantization operations have been achieved as well. As shown in Fig. 8, we found that the AFP-encoded model saves power 9.3× for MAC and 13.2× on average for quantizer with calibration, which dramatically reduces the hardware cost.

6. Conclusion

In this paper, we introduced a novel quantization framework called AFP. It combines the compromise between the stringent resource constraint and the accuracy tolerance for NNs. Therefore, it can automatically find an optimal bit-width allocation scheme that makes the range and precision not redundant, but sufficient. This method can quantize both the activation and weight of a pre-trained model to low bit-width (less than 8-bit) without accuracy loss. Notably, the AFP is hardware-friendly. The evaluation show that our method has comprehensive advantages over existing solutions and is superior in terms of accuracy and power.

Acknowledgment

This work was partially supported by DeepBlue Technology (Shanghai) Co., Ltd., the National Natural Science Foundation of China (Grant No.61834006) and National Key Reserch and Development Program of China (2018YFB1403400).

References

[1] Chaim Baskin, Natan Liss, Yoav Chai, Evgenii Zheltonozhskii, Eli Schwartz, Raja Giryes, Avi Mendelson, and Alexander M Bronstein. Nice: Noise injection and clamping estimation for neural network quantization. *arXiv preprint arXiv:1810.00162*, 2018. 3

[2] 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. 3, 7

[3] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1468–1477, 2020. 2, 3

[4] 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. 1, 2, 3

[5] Yoni Choukroun, Eli Kravchik, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. *arXiv preprint arXiv:1902.06822*, 2019. 7

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition (CVPR)*, pages 248–255. Ieee, 2009. 6

[7] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020. 3

[8] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 1, 2, 3

[9] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations (ICLR)*, 2016. 1, 2, 3, 7

[10] Zhezhi He and Deliang Fan. Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11438–11446, 2019. 2

[11] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993. 5

[12] 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)*, pages 2704–2713, 2018. 1, 2, 3, 7

[13] 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. 1, 2, 3, 7

[14] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4350–4359, 2019. 1, 3

[15] William Kahan. Ieee standard 754 for binary floating-point arithmetic. *Lecture Notes on the Status of IEEE*, 754(94720-1776):11, 1996. 2, 3, 4

[16] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019. 2, 3, 4

[17] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2019. 1, 2, 3, 7

[18] Jiachen Mao, Xiang Chen, Kent W Nixon, Christopher Krieger, and Yiran Chen. Modnn: Local distributed mobile computing system for deep neural network. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1396–1401. IEEE, 2017. 1

[19] Bradley McDanel, Sai Qian Zhang, HT Kung, and Xin Dong. Full-stack optimization for accelerating cnns using powers-of-two weights with fpga validation. In *Proceedings of the ACM International Conference on Supercomputing (ICS)*, pages 449–460. ACM, 2019. 3

[20] Szymon Migacz. 8-bit inference with tensorrt. In *GPU technology conference*, volume 2, page 5, 2017. 2

[21] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1325–1334, 2019. 7

[22] NVIDIA. *NVIDIA A100 tensor core GPU architecture*, 2020. whitepaper v1.0. 2

[23] 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. 7

[24] John G Proakis, Masoud Salehi, Ning Zhou, and Xiaofeng Li. *Communication systems engineering*, volume 2. Prentice Hall New Jersey, 1994. 4

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[26] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25:2951–2959, 2012. 5

[27] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017. 5

[28] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 8612–8620, 2019. 2, 5, 7

[29] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006. 5, 6

[30] 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. 5

[31] Shaokai Ye, Xiaoyu Feng, Tianyun Zhang, Xiaolong Ma, Sheng Lin, Zhengang Li, Kaidi Xu, Wujie Wen, Sijia Liu, Jian Tang, et al. Progressive dnn compression: A key to achieve ultra-high weight pruning and quantization rates using admm. *arXiv preprint arXiv:1903.09769*, 2019. 7

[32] Shaokai Ye, Tianyun Zhang, Kaiqi Zhang, Jiayu Li, Jiaming Xie, Yun Liang, Sijia Liu, Xue Lin, and Yanzhi Wang. A unified framework of dnn weight pruning and weight clustering/quantization using admm. *arXiv preprint arXiv:1811.01907*, 2018. 3

[33] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018. 2, 3, 7

[34] Yiren Zhao, Xitong Gao, Daniel Bates, Robert Mullins, and Cheng-Zhong Xu. Focused quantization for sparse cnns. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5585–5594, 2019. 7

[35] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *5th International Conference on Learning Representations (ICLR)*, 2017. 1, 2, 3, 7