# Edge Inference with Fully Differentiable Quantized Mixed Precision Neural Networks

Clemens JS Schaefer*†, Siddharth Joshi†, Shan Li‡ and Raul Blazquez‡
† University of Notre Dame, Notre Dame, IN, USA
‡ Google LLC, Mountain View, CA, USA
{*cschaef6, sjoshi2*}*@nd.edu*, {*lishanok, rblazquez*}*@google.com*

## Abstract

*The large computing and memory cost of deep neural networks (DNNs) often precludes their use in resource-constrained devices. Quantizing the parameters and operations to lower bit-precision offers substantial memory and energy savings for neural network inference, facilitating the use of DNNs on edge computing platforms. Recent efforts at quantizing DNNs have employed a range of techniques encompassing progressive quantization, step-size adaptation, and gradient scaling. This paper proposes a new quantization approach for mixed precision convolutional neural networks (CNNs) targeting edge-computing. Our method establishes a new Pareto frontier in model accuracy and memory footprint demonstrating a range of pre-trained quantized models, delivering best-in-class accuracy below 4.3 MB of weights and activations without modifying the model architecture. Our main contributions are: (i) a method for tensor-sliced learned precision with a hardware-aware cost function for heterogeneous differentiable quantization, (ii) targeted gradient modification for weights and activations to mitigate quantization errors, and (iii) a multi-phase learning schedule to address instability in learning arising from updates to the learned quantizer and model parameters. We demonstrate the effectiveness of our techniques on the ImageNet dataset across a range of models including EfficientNet-Lite0 (e.g., 4.14 MB of weights and activations at 67.66% accuracy) and MobileNetV2 (e.g., 3.51 MB weights and activations at 65.39% accuracy).*

## 1. Introduction

Deep neural networks (DNNs) demonstrate remarkable performance at computer vision tasks, notably being the defacto standard methods employed for large scale image recognition ( [6, 46]). However, modern deep learning models require substantial compute and memory resources. This presents a challenge in deploying DNNs on resource constrained edge hardware.

Techniques for developing edge-deployable DNNs include the design of hardware-friendly DNN models, the development of low-power/latency hardware, model pruning, and quantizing DNNs to operate at lower precision [35, 38, 40]. Since low-precision operations can simultaneously lower the memory footprint, increase throughput, and lower the latency for DNN inference, DNN quantization has become increasingly important [24]. In particular, with recent DNN accelerators [25, 45] and graphics processing units (GPUs) [36] offering support for mixed-precision computing, these benefits can be realized on existing hardware [43]. Furthermore, the compilers and hardware community are actively researching how to extend the support for multiple low-bit width mixed precision operations [14, 29, 30, 33, 34]. With additional compiler support for mixed precision quantization, such quantization could be leveraged for DNNs deployed on field programmable gate arrays (FPGAs). However, existing quantized DNN models do not fully leverage such hardware capabilities. In particular, most model quantization approaches focus on weight quantization, ignoring the high energy and latency costs of moving and storing activations. Benchmarking indicates that in intermediate layers, these costs dominate in accelerators [7].

Previous research [39, 41, 43] has shown promising results with heterogeneous quantization, allocating memory resources per layer. This form of quantization can facilitate a wider range of trade offs between networks size and accuracy. Building on this insight, we focus this paper on developing compact DNN models extremely low memory footprints. We report best-in-class results for models with a total memory footprint below 4.3 MB.

In this paper we: (i) present a hardware-aware mixed precision differentiable quantization formulation which includes per-tensor learned precision for activations and fine-grained per-channel quantization for weights, (ii) propose a novel gradient modification scheme which entails modifying weights and activation gradients differently and introduce

*Work partly conducted while interning at Google LLC

arctanh based gradient scaling together with comprehensive evaluations against other gradient scaling techniques, and (iii) introduce a multi-phase learning-rate schedule to address instability in learning arising from updates to the learned quantizers and perform extensive comparisons of this schedule with alternatives. We show the effectiveness of our methods on the ImageNet dataset using the EfficientNet-Lite0, MobileNetV2, wide SqueezeNext, and ResNet18 model architectures. We demonstrate state-of-the-art accuracy for multi bit-width models ranging from 2.89–4.3 MB total memory footprint. Across various model architectures, our quantization scheme forms the Pareto optimal frontier for model accuracy vs. model size.

## 2. Background

Uniform quantization is emulated in software using rounding and clipping of floating-point values, expressed as:

$$Q_u(x, d, q_+) = \text{round}\left(\text{clip}\left(\frac{x}{d}, -q_+, q_+\right)\right) \cdot d. \quad (1)$$

Where $Q_u$ is the quantization function, $x$ the value to be quantized, $d$ the step size and $q_+$ is the dynamic range. Achieving high model-accuracy given $b$ bits to represent values, entails a careful choice of the dynamic range ($q_+$) and (implicitly) the step size ($d$). The dynamic range relates to the step size and the number of bits as $b = \log(q_+/d) + 1$. In this paper, we use *calibration* for the process of determining these values during training. As part of calibration, $q_+$ is often smaller than $\max(\text{abs}(x))$ to further improve quantization efficiency. See supplementary materials A for a detailed study on the different scaling between $\max(\text{abs}(x)$ and $q_+$.

There are two main approaches to quantizing a floating point model: Post training quantization (PTQ) and Quantization Aware Training (QAT). PTQ does not typically require retraining or fine-tuning, with recent work by Dai et al. [9] demonstrating a 6 bit quantized ResNet-50 delivering 75.80% accuracy on ImageNet. However, the authors report limited success in but retaining accuracy in low-bit regimes, dropping from 75.80% with 6 bits to 7.11% with 3 bits. QAT takes into account quantization of weights and activations during training, and consequently has seen greater use in the low-bit regimes. However, QAT must address challenges arising from the non-differentiability of the quantization function which results in vanishing when propagated through multiple quantized layers.

### 2.1. Quantization Aware Training

The straight-through-estimator (STE) [2] is commonly used to avoid this problem, replacing the derivative of a discretizer (rounding) with that of an identity function. Essentially, ignoring the rounding function in the backward pass and preserving gradient flow.

To further enhance QAT performance, Choi et al. [8] introduce PACT which makes the dynamic range (see eq. (1) $q_+$) a trainable parameter. They achieve 75.3% evaluation accuracy on ImageNet with a ResNet50 quantized down to 3 bits while simultaneously stabilizing training. Jung et al. [26] examine the use of non-uniform quantization and study the effect of learned quantization levels. The resulting quantizer achieves 73.1% evaluation accuracy on ImageNet while using a ResNet34 quantized to 3 bits. However, non-uniform quantizers cannot always be mapped on to fixed-point arithmetic and can incur significant overhead when deployed or implemented in hardware [11]. Learning the step-size $d$ while learning a uniform quantization scheme, Esser et al. [13] developed LSQ which achieves an accuracy of 74.3% on ImageNet using a 3-bit ResNet34. In LSQ+, Bhalgat et al. [3] build on this technique by parameterizing the symmetry of the quantizer to accommodate modern activation functions such as swish, h-swish and mish, which have limited, but critical negative excursions. They demonstrate the effectiveness of their method on modern architectures by achieving 69.9% accuracy with a 3-bit Efficient-B0 and 66.7% accuraacy with a 3-bit MixNet on ImageNet. However, these methods do not quantize the first and last layers of the models, which typically incur significant performance degradation. In contrast, recent work using progressive-freezing and iterative training to quantize MobileNets [32] achieve 71.56% accuracy on ImageNet (note that MobileNets are significantly smaller than ResNets).

### 2.2. Heterogeneous Quantization

Model accuracy is not equally sensitive to quantization in different layers [32]. Prior work leverages this to allocate numerical precision on a per-layer basis [12, 32, 41, 44]. In HAQ [41], the authors use reinforcement learning with a hardware simulator generating energy and latency estimates to optimize the bit-width of every layer. They report a $1.9\times$ improvement to energy and latency while maintaining 8-bit levels of accuracy for MobileNet models. More recently, second order techniques like [11] use Hessian eigenvalues as a quantization sensitivity metric and assign layer bit-widths. This Hessian aware trace-weighted quantization (HAWQ) offers 75.76% accuracy (ImageNet) for ResNet50 with an average of 2 bit weights and 4 bit activations.

Uhlich et al. [39] (label Mixed in Figure 1) formulate a fully differentiable quantization scheme, where both the step-size and the dynamic range are trainable, using a symmetric uniform quantizer $Q_U(x, d, q_+)$. This formulation implicitly learns the bit-width. Additionally, the authors add an additional constraint to the loss function to target a network weight size and maximum feature map size. Taken together, their improvements result in a MobileNetV2 with a weight memory footprint of 1.55 MB and a maximum activation feature size of 0.57MB, while delivering an accuracy of

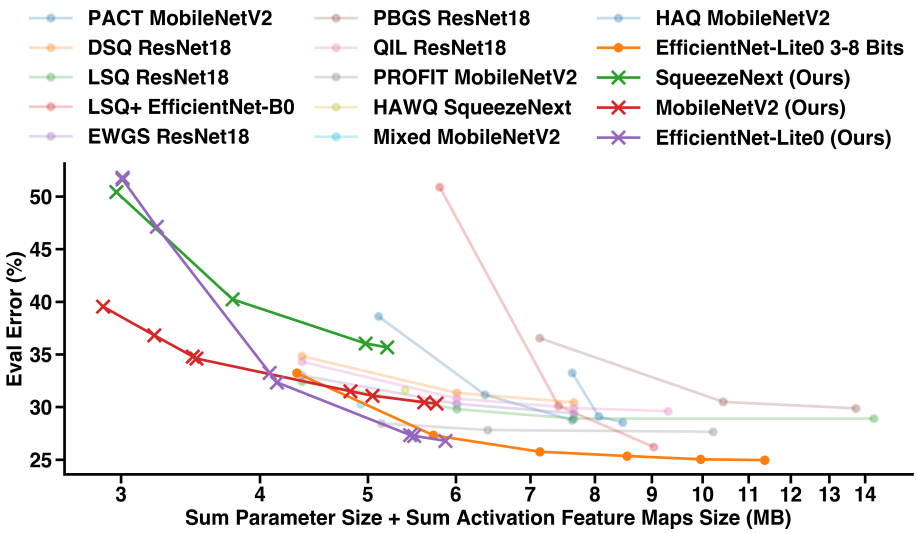


Figure 1. Our results quantizing different models compared to state-of-the-art. Network size (sum of parameters and activations) is compared to the evaluation error on the ImageNet dataset. Models quantized with our method occupy the Pareto-frontier, delivering smaller multi-bit networks at higher accuracy. We shows results from: PACT [8], DSQ [17], LSQ [13], LSQ+ [3], EWGS [28], PBGS [27], QIL [26], PROFIT [32], HAWQ [11], HAQ [41], Mixed [39]

69.74% on ImageNet (an estimated 4.93 MB for the weights and sum of activations).

### 2.3. Gradient Scaling for Quantization

Most models employ straight-through-estimators (STEs) in QAT for quantized neural networks essentially ignoring discretization in the backward pass. Alternatively some recently proposed techniques avoid this discrepancy by employing a smooth function, e.g., stacked $\tanh$, prior to the non-differentiable quantizer to emulate quantization effects facilitating gradient flow across layers (DSQ [17] in Figure 1). They demonstrate a 2-bit ResNet18 delivering 65.17% accuracy on ImageNet. Since the authors cascade this soft quantizer with a hard quantizer, they still employ an STE to propagate gradients in the backwards pass.

Kim et al. [27] (PBGS in Fig. 1) propose gradient scaling as a regularizer to learn 'easy to quantize' networks, training models with gradients scaled to induce values on the quantization grids. The authors demonstrate results for a 4-b quantized ResNet18 trained to 63.45% evaluation accuracy on ImageNet. Nguyen et al. [31] achieve the same effect through regularization, using the absolute cosine function for a 6-bit automatic speech recognition recurrent neural network with only a 2.68% accuracy degradation compared to the baseline model. Lee et al. [28] (EWGS in Figure 1) combine quantization in the forward pass and gradient scaling in the backward pass to account for discretization errors between inputs and outputs of the quantizer. They incorporate the sign and magnitude of the discretization error as well as second-order information to determine the gradient scaling factor. This method achieves 67% evaluation accuracy on ImageNet with ResNet18 quantized to 2 bits.

## 3. Methods

Given a pretrained floating point model we quantize it in three phases: (i) homogeneous pre-training phase, (ii) a phase to learn precisions, and (iii) a final finetuning phase where only model parameters are updated (see Fig. 3). During all those phases we employ gradient scaling to account for discretization in the backward pass. We implement an initial calibration phase for both weights and activations using Gaussian calibration for the weights and the $99.99^{\text{th}}$ percentile for the activations, which improves our homogeneous quantization performance by up to 1.22% for 3-bit EfficientNet compared to sample maximum calibration (see suppl. mat. Table 2 for details). During phase (ii) we employ penalty scheduling and reduce the frequency of quantizer parameter updates to combat learning instabilities. We also incorporate the weight and activation size penalty into the loss function like Uhlich et al. [39], resulting in:

$$\begin{aligned} L =& CE(x, y) + \beta \max\left(\left(\sum_{l=1}^{L}\sum_{c=1}^{C} b_{lc}^{w} \cdot s_{lc}^{w}\right) - t^{w}, 0\right)^2 \\ &+ \beta \max\left(\left(\sum_{l=1}^{L} b_{l}^{a} \cdot s_{l}^{a}\right) - t^{a}, 0\right)^2 . \end{aligned} \tag{2}$$

Here, $x$ is the input, $y$ the target, $CE$ stands for the cross

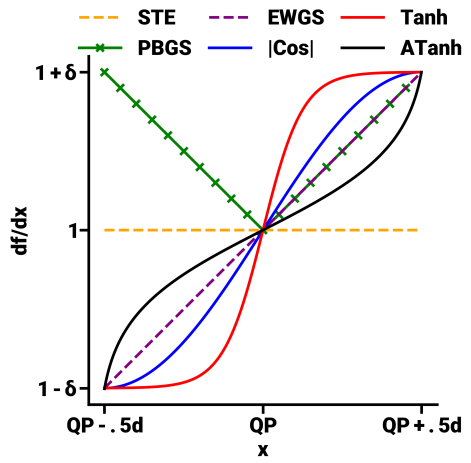


Figure 2. We illustrate different gradient scaling functions: straight-through-estimators (STE [2]), elementwise gradient scaling (EWGS [28]), position based gradient scaling (PBGS [27]), absolute cosine regularization (Acos [31]) as well as hyperbolic tangent function (Tanh) and its inverse (InvTanh). Note that $QP$ denotes quantization point, $d$ is the step size and $\delta$ is the magnitude control hyper parameter for gradient scaling.

entropy loss, $b_l$ represents the bitwidth of layer $l$, $s_l$ is the number of parameters of a given layer (the additional $c$ indicates the channel) and $t$ is the target size for the model. The superscripts $w$ and $a$ indicate weights and activations respectively. We use rectified quadratic penalties to enable an accuracy vs. model size trade-off during training, with the rectification used to prevent penalization once the size budget is met. A single modulating factor $\beta$ controls the penalty on model size.

We use the sum of weights and activation feature maps as our cost metric for efficiency. Data movement caused by memory accesses for weights and activations dominates other factors in modern edge accelerators [42]. Consequently, for such hardware, accounting for the total number of memory accesses for heterogenously quantized tensors, is crucial to determining model efficiency. Existing metrics like number of operations or parameter footprint, do not fully consider data movement or variable precision tensors. The Arithmetic Computation Effort (ACE) [47] is an alternative compute-focused metric that has been proposed for multi-precision edge accelerators. Our metric is strongly correlated with ACE (0.956) across multiple configurations, but offers more direct interpretability of the cost.

### 3.1. Quantization Training Dynamics

Learning both model parameters and bit-width allocations is necessarily a higher dimensional problem than just learning the parameters alone. This increases the search's sensitivity to initial conditions. To mitigate this, we de-couple these two elements by starting out training on a homogeneously quantized network (from the pre-training) and then training the per-layer quantizer with progressively increasing pressure from the model size constraints. As shown in Figure 3 (left), the model suffers from dramatic accuracy degradation when the model size penalty is imposed. Often, this results in catastrophic training failure due to the model size penalty dominating cross-entropy loss. We use a soft-transition on the size penalty ($\beta$) by linearly increasing $\beta$ to it's final value. This initial training can be viewed as enabling coarse navigation to the optimal in the solution space, followed by a more fine-grained descent towards the joint model-quantizer optima. As seen in Figure 3 (left), parameter updates can recover some lost accuracy after $\beta$ saturates.

We observed model instability and training failure when both bit-precision and model parameters were updated frequently. Infrequent updates prevented thorough exploration of the model search space, resulting in models that were still similar to their homogeneously quantized initializations. We avoid this by limiting the bit-precision update frequency, restricting updates to every $\Phi$ steps. In our experiments, an update frequency of $\Phi = 20$ provided sufficient time for model statistics (e.g. batch norm) to stabilize and model parameters to adapt to the new precision level.

Subsequently, we enable finer-grained quantization by adapting the precision of the convolution weight kernels at a per-output-channel granularity (tensor slices). Most existing hardware with mixed-precision support can compute a single channel at a given precision. However, finer grained quantization can entail significant hardware overhead [23]. Banner et al. [1] have previously shown flexible per-channel bit-widths by solving a layer-wise noise minimization problem, in contrast we learn the per-layer bit-widths based on the overall model loss function. Indeed, the variation in dynamic range across channels from a heterogeneously quantized EfficientNet-Lite0 and MobileNetV2 demonstrate the efficiency gains available using this technique (See Figure 3 right). To prevent the gains from quantized operation getting negated, we ensure that quantization granularity is not too fine-grained. After these quantizer parameters converge, we freeze them by setting $\beta = 0$ in eq. (2). This is followed by an additional fine-tuning period with a decaying learning rate to recover accuracy. In particular, our results suggest that batch normalization statistics are stabilized through this fine-tuning period, recovering accuracy.

### 3.2. Gradient Scaling

The STE operator is the de facto standard for enabling backpropagation through non-differentiable functions. We study how different gradient scaling techniques compare to STE across models and levels of precision. As shown in Figure 4, $\tanh$-based gradient scaling for activations and linear scaling for weights [28] outperforms other combinations.

We examine the effect of gradient scaling across multiple scaling functions ($f(x)$) enumerated below (and shown in Figure 2 (left)):

1. Position based gradient scaling (PBGS) [27]:
$$\text{scale} = 1 + \delta \cdot |x - \text{round}(x)|.$$

2. Element-wise gradient scaling (EWGS) [28]:
$$\text{scale} = 1 + \delta \cdot \text{sign}(g_x) \cdot (x - \text{round}(x)).$$

3. Modified absolute cosine (Acos) [31] gradient scaling:
$$\text{scale} = 1 + \delta \cdot \sin(\pi \cdot (x - \text{round}(x))).$$

4. Hyperbolic tangent (Tanh) gradient scaling:
$$\text{scale} = 1 + \delta \cdot \text{sign}(g_x) \cdot \tanh(\alpha \cdot (x - \text{round}(x))).$$

5. Inverse hyperbolic tangent (InvTanh) gradient scaling:
$$\text{scale} = 1 + \delta \cdot \text{sign}(g_x) \cdot \text{arctanh}(\alpha \cdot (x - \text{round}(x))).$$

Here, $\delta$ is a general hyperparameter to modulate the magnitude of gradient scaling and $g_x$ is the gradient. We also introduce an additional hyperparameter, $\alpha$, to control the steepness of the hyperbolic tangent functions. Figure 4 (right) shows the performance of various gradient scaling techniques for activations and weights when homogeneously quantizing EfficientNet-Lite0 to 3 bits. We examined the effect over 20 trials and show the resulting distribution in Figure 4. The horizontal dotted lines represent the baseline performance (when both acts. and wgts. use the same scaling technique) of the STE and EWGS method respectively. We observe that different gradient scaling schemes benefit the training performance for weights and activations, owing in part to different underlying distributions. Indeed, employing STEs for both weights and activations provides the performance baseline. We note that the improvements derived from gradient scaling on activations only (configurations *a*, *b*, *c*) lead to lower performance gains when compared to applying gradient scaling on weights only (configurations *d*, *e*, *f*). This suggests that the performance gains from gradient scaling can be primarily attributed to gradient scaling of weights. Although configuration *e* (weights use EWGS and activations use STE) delivered the best single-run result, as seen in Figure 4, these results were not consistent across trials. Indeed, in some trials, this configuration performed worse than the baseline. Indeed, the majority of the performance gains from applying EWGS area also observed in configuration *e*, where EWGS is only applied to weights while the activations use STEs. We observed that linear gradient scaling (EWGS) for weights and the inverse of the hyperbolic tangent scaling of activation gradients provides consistent improvement over the baseline, proving to be the more robust gradient-scaling technique. More comprehensive sensitivity analysis and analysis of computational overhead are provided in A.2 in the supplementary material.

# 4. Experiments

We demonstrate the effectiveness of our proposed recipe on the ImageNet [10] dataset across multiple models including EfficientNet-Lite0 [38], MobileNetv2 [35], wide SqueezeNext [15], and ResNet18 [18]. We use the Flax [19] and Jax [5] frameworks to implement the networks, quantization scheme, and training routine. All codes are available under https://github.com/Intelligent-Microsystems-Lab/HeterogeneousQuantization

## 4.1. Setting

We ran our experiments on TPUv3 accelerators using the Google Cloud. Each experiment ran on a single instance which comprises 8 cores and 32GB memory. We provide some hardware synthesis-based estimates of the latency impact of quantized models in Supplementary Materials A.5.

We implement standard input pre-processing for training, randomly cropping images to $224 \times 224$ with 3 channels (RGB), followed by input augmentation. Our augmentations include a random flip (left or right) and channel normalization (mean 127 and standard deviation 128). During evaluation the image is cropped around the center and no random flip is applied. Training uses RMSProp [20] with $0.9$ Nesterov momentum and a learning rate of $10^{-4}$. We increase the learning rate linearly from zero during the first two epochs and subsequently reduce it to zero in a cosine decay. Our training batch size is 1024, which is evenly split across the 8 cores of the TPU for single-program, multiple-data (SPMD) parallelism. We apply weight decay of $10^{-5}$ and label smoothing of $10^{-1}$ for improved accuracy. Each QAT training phase (pretraining, heterogenous training, and finetuning) lasts 50 epochs.

## 4.2. Gradient Scaling

We determine the gradient scale factor $\delta$ ($5 \cdot 10^{-3}$) through a grid search. Results of the gradient scaling can be seen in Figure 4 (right) and for non-mixed configurations in supplementary materials A.2. Both show data from 20 trials on a 3-bit homogeneously quantized EfficientNet-Lite0, illustrating the variance in accuracy from employing gradient scaling methods. Figure 8 (supplementary materials) shows a smaller grid search for ideal gradient scaling on the same network quantized to 2 and 4 bits. Notably, variance in final accuracy increases when quantizing to fewer bits whereas the difference in accuracy gets exacerbated.

## 4.3. Model Quantization

Figure 1 provides context for our results, comparing our techniques with state of the art quantization methods [3, 8, 11, 13, 17, 26–28, 32, 39, 41]. If unavailable, we computed the weight and activation sizes based on our repli-

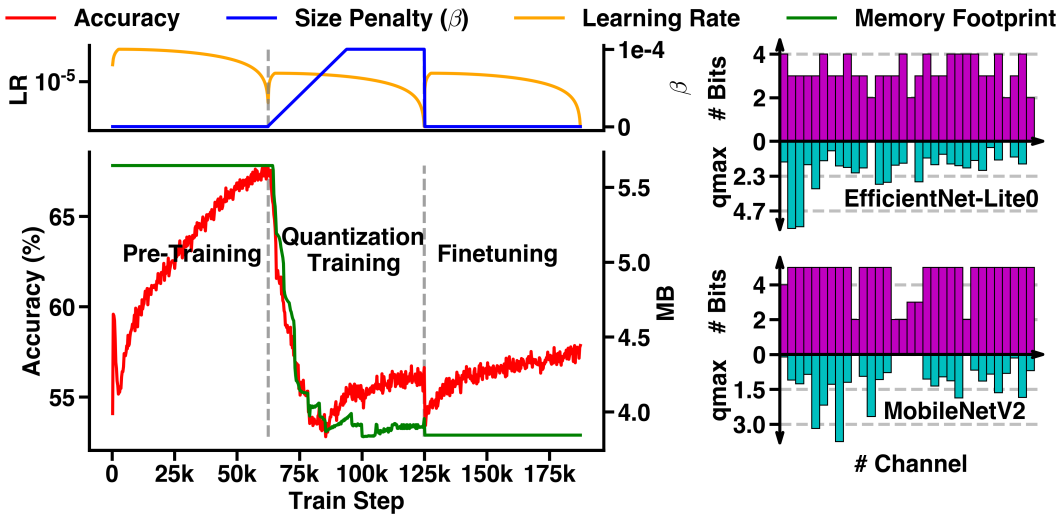


Figure 3. On the left illustrating the three phases of our mixed precision training method. The top left shows scheduled learning rate and size penalty ($\beta$) term meanwhile the bottom left shows the evolution of model accuracy and model size. On the right we show an example per-channel bit allocation in a weight kernel of an EfficientNet-Lite0 and MobileNetV2. Extreme quantization is correlated to low dynamic range ($q_+$). However, the contra does not necessarily hold.

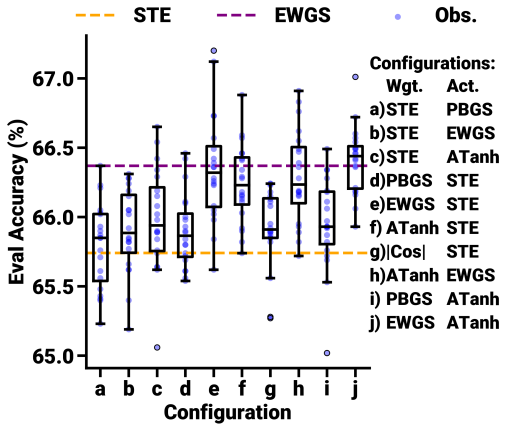


Figure 4. The performance of different mixed gradient scaling functions (different gradient scaling for weights and activations) on a homogeneous 3 bit EfficientNet-Lite0 on ImageNet.

cation of their work (all assumptions are shown in the supplementary materials Table 4). The total network size (x-axis in Fig. 1 includes batch norm parameters for which we assume a bfloat16 data type. Although no small-scale model (including our own) reports quantized batch normalization, some related work do report results from quantizing the batch norm layers for a 4-bit ResNet18 with a higher memory footprint than our reported results Yao et al. [43].

Our method improves the Pareto frontier for model error and model size, with the greatest improvement seen in the sub 6 MB region. Here, we define size to be the sum of weights and sum of activation feature maps. We specifically optimize the total size of the activations, because of the high energy cost arising from data movement for activations [7, 45]. To the best of our knowledge, we report best-in-class accuracy for multi-bit models in extreme constraint settings with less than 4.3 MB available for both weights and activations. Our efficient frontier (Figure 1) shows the degradation in performance across various target budgets and encapsulates the difference in robustness of various quantized model architectures. The EfficientNet-Lite0 architecture performs well for homogeneous quantization and heterogeneous quantizaton in a range between 4-6 MB total model size, meanwhile MobileNetV2 delivers better accuracy below 4 MB. We note the different scaling trends seen for quantized EfficientNet-Lite0 and MobileNetV2, suggesting that MobileNetV2 may be more suitable for ultra-low budget applications. We included a wide SqueezeNext model due to its small weight footprint, however due to its depth the activation size dominate the overall model budget and making it a strictly worse model across the target budgets. ResNet18 and SqueezeNext results are provided in A.3.

Figure 5 shows the detailed layer-wise bit-allocation of heterogeneously quantized EfficientNet-Lite0 with a total size of 3.43 MB and MobileNetV2 with 3.41 MB. The EfficientNet-Lite0 architecture imposes greater precision in weights at the early layers of the network compared to the later layers. However the activations follow a constant pattern throughout the network. Notably, the activation of the last pointwise convolutional layer of a mobile inverted bottleneck (MBConv block) are higher in compar-

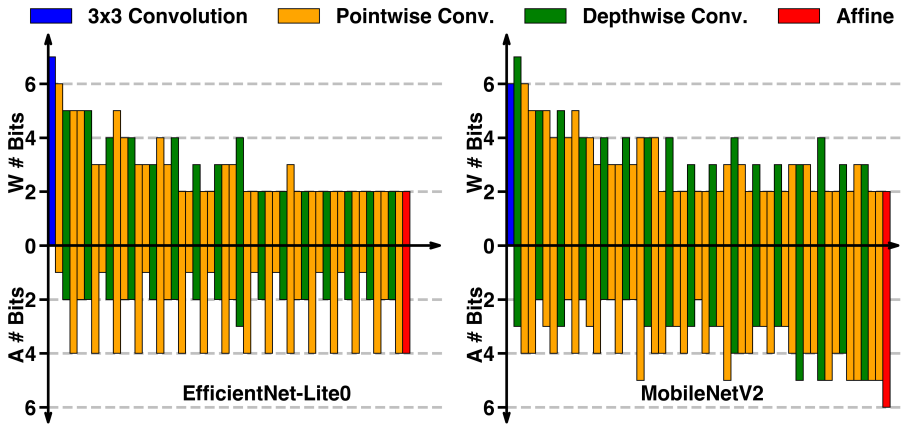


Figure 5. Internal bit allocation across layers of weights (up) and activations (down) for EfficientNet-Lite0 and MobileNetV2. Weights in the first layers have higher bit-widths for both models. Activations bitwidths for EfficientNet-Lite0 form a high precision path, e.g. activations which are residuals have higher precision. For both models the last affine layer has high precision.

ison to the other layers of MBConv blocks. Our results suggest that pointwise convolutional layers that have both residual and direct inputs require much higher precision to prevent quantization-induced information loss. This high precision bit-allocation indicates a critical information flow pathway. The MobileNetV2 bit allocation is similar to that of EfficientNet-Lite0, with higher precision weights in the initial layers which reduce with network depth. The critical path for the activation is not as pronounced as it is for the EfficientNet architecture and additionally high activation bit-width are allocated to layers closer to the final affine layer whose activation bit-width is the highest.

## 4.4. Additional Consideration

### 4.4.1 Bias Quantization

While our reported models quantize biases, we also examined the effect of keeping biases at higher bfloat16 precision. Typically, accelerators can pre-load biases into the hardware accumulator, minimizing the energy impact. We summarize the impact of bias quantization in Table 1. Crucially, we note that MobileNetV2 at larger memory budgets do not see accuracy benefits ($\leq 0.01\%$ higher accuracy). However, EfficientNet-Lite0 models with tight memory budgets do see an increase their accuracy by approx. 1.56%.

### 4.4.2 Knowledge Distillation

Recently proposed quantization techniques have shown that applying knowledge distillation (KD) to their existing quantization techniques can improve results. We examine how KD impacts the EfficientNet-Lite0 and MobileNetV2 models on our Pareto frontier 1. We use the KD process in [21], using soft labels created by a B16 vision transformer [6] with an accuracy of 85.49% (soft-label KD). We also examined the

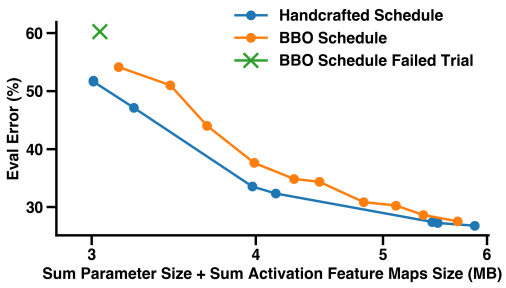


Figure 6. Comparison between QAT employing our schedule and the scheduled derived through a distributed black-box optimization (BBO) method evaluated for quantizing an Efficient-Lite0. The BBO-derived schedule performs strictly worse than ours and fails to train a network on the strictest budget (2 bits on average).

knowledge distillation technique employed by PROFIT [32], here the knowledge of the teacher model is induced into the student model through a penalty term in the loss function (penalty KD). The results summarized in Table 1 show that knowledge distillation can have a positive effect on the accuracy of up to 0.36% but can also have negative effects. Neither of the KD techniques examined dramatically altered the model accuracy across the Pareto frontier.

### 4.4.3 Exploring Training Schedule and Quantization Approaches

We evaluate our proposed QAT schedule against both automatically searched schedules and those derived using convex optimization approaches.

**Automated Schedule Search** We compare our handcrafted QAT schedule (see Fig. 3) to a schedule derived

Table 1. Effect of knowledge distillation (KD) on heterogeneously quantized networks and unquantized biases for the last affine layer. The first column shows the effect of KD on floating point networks the following columns are networks from the efficient frontier. Soft-label KD refers to a KD technique where the targets of the student network are the predictions of the teacher network [21]. Meanwhile penalty KD uses one-hot encoding as the target and adds a penalty term to force the student model prediction to align to the teacher [32].

| EfficientNet-Lite0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Size (MB) | 22.66 | 3.01 | 3.23 | 3.98 | 4.14 | 5.45 | 5.50 | 5.87 |
| Base Accuracy (%) | 75.53 | 48.37 | 52.87 | 66.46 | 67.66 | 72.56 | 72.75 | 73.21 |
| Soft-Label KD | -0.31 | 0.21 | 0.24 | 0.11 | 0.10 | 0.17 | 0.10 | -0.11 |
| Penalty KD | -0.22 | 0.16 | 0.27 | 0.08 | 0.16 | 0.17 | 0.04 | -0.11 |
| No Bias Quant | - | 1.56 | 1.16 | 0.20 | 0.26 | 0.14 | 0.15 | 0.00 |

| MobileNetV2 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Size (MB) | 20.25 | 2.89 | 3.21 | 3.48 | 3.51 | 4.82 | 5.05 | 5.62 | 5.76 |
| Base Accuracy (%) | 71.46 | 60.72 | 63.18 | 65.20 | 65.39 | 68.50 | 68.93 | 69.54 | 69.68 |
| Soft-Label KD | 0.15 | 0.36 | 0.12 | 0.13 | 0.23 | 0.12 | -0.26 | -0.24 | 0.02 |
| Penalty KD | 0.03 | 0.16 | 0.17 | 0.17 | 0.28 | 0.11 | -0.19 | -0.23 | -0.03 |
| No Bias Quant | - | 0.38 | -0.01 | 0.24 | 0.08 | 0.21 | 0.05 | 0.16 | 0.13 |

through distributed black-box optimization (BBO) [16, 37], to determine the performance of our approach. We set up the BBO search space to include: (i) frequency of bit-width update, (ii) weight & activation penalty ($\beta$ in eq (1)), (iii) homogeneous bit-width for pre-training, (iv) ramp-up length of the quantization training, and (v) ramp-up mode (linear, cosine, or exponential). The BBO was directed to maximize accuracy for the EfficientNet model on a randomly sampled subset of the training data while minimizing the discrepancy between model budget and achieved size. We chose an average of 3 bits for the model budget (1731 kB for weights and 2505 kB for the sum of activations). The BBO conducted 766 evaluations within our compute budget (approx. 14,000 accelerator hours) to converge to a recipe.

Figure 6 compares our schedule against the schedule determined by the BBO, showing that our schedule consistently outperforms the automated search. For the strict budget (average of 2 bits for each tensor), the BBO-derived schedule exceeds the allocated budget by 135.15 kB. We hypothesize that our hand-crafted schedule outperforms the BBO schedule due to the large search space, with varying levels of sensitivity to the scheduling parameters.

**Optimization-Based Quantization** Our problem requires simultaneously optimizing for accuracy and model size, constrained by a memory budget (see eq. (2)). Alternating direction method of multipliers (ADMM) [4], combines the decomposability of dual ascent with the convergence guarantees of the method of multipliers making it an attractive solution with theoretical grounding. We reformulate our quantization to be compatible with ADMM by separating the problem objectives into optimizing for accuracy and model size, constrained by equality between model parameters between the two optimization steps. ADMM, then operates on two sets of model weights, updating weights optimizing each objective and until the two sets of parameters converge.

Our experimental findings demonstrate the superiority of our gradient-based approach over ADMM. When quantizing an EfficientNet-Lite0 to an average bit-width of 4 bits, our method achieved 72.46% accuracy, surpassing the 69.21% achieved by ADMM. Details on the ADMM formulation and parameters (including hyperparameter search) are in A.6.

## 5. Conclusions

We introduce a recipe for quantization-aware training for heterogeneously quantized neural networks where bit-widths are trained alongside model parameters. We employ a novel gradient scaling function to account for discretization due to quantization in the backward pass. Combined with careful scheduling in penalizing the accuracy loss and model size allows us to achieve exceed state-of-the-art model quantization. Models quantized by our technique occupy the Pareto optimal frontier of model size (including weights and activations) against performance (evaluation error) on the ImageNet dataset. To the best of our knowldege, our methods delivers best in class multi-bit neural networks with total memory footprint below 4.3 MB. Extensive evaluation and sensitivty analysis verifies our quantization performance.

## References


[1] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019. 4

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 2, 4, 13

[3] 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. 2, 3, 5, 11

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011. 8

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 5

[6] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations. *arXiv preprint arXiv:2106.01548*, 2021. 1, 7

[7] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019. 1, 6

[8] 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. 2, 3, 5, 11

[9] Steve Dai, Rangha Venkatesan, Mark Ren, Brian Zimmer, William Dally, and Brucek Khailany. Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference. *Proceedings of Machine Learning and Systems*, 3:873–884, 2021. 2, 11

[10] 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*, pages 248–255. Ieee, 2009. 5

[11] 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*, 33:18518–18529, 2020. 2, 3, 5, 11

[12] Ahmed T Elthakeb, Prannoy Pilligundla, FatemehSadat Mireshghallah, Amir Yazdanbakhsh, and Hadi Esmaeilzadeh. Releq: A reinforcement learning approach for deep quantization of neural networks. *arXiv preprint arXiv:1811.01704*, 2018. 2

[13] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 2, 3, 5, 11

[14] Angelo Garofalo, Yvan Tortorella, Matteo Perotti, Luca Valente, Alessandro Nadalini, Luca Benini, Davide Rossi, and Francesco Conti. Darkside: A heterogeneous risc-v compute cluster for extreme-edge on-chip dnn inference and training. *IEEE Open Journal of the Solid-State Circuits Society*, 2:231–243, 2022. 1

[15] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. Squeezenext: Hardware-aware neural network design. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1638–1647, 2018. 5

[16] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1487–1495. ACM, 2017. 8, 12

[17] 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. 3, 5

[18] 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. 5

[19] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. 5

[20] Matteo Hessel, David Budden, Fabio Viola, Mihaela Rosca, Eren Sezener, and Tom Hennigan. Optax: composable gradient transformation and optimisation, in jax!, 2020. 5

[21] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 7, 8

[22] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248*, 2019. 14

[23] Ehab M Ibrahim, Linyan Mei, and Marian Verhelst. Survey and benchmarking of precision-scalable mac arrays for embedded dnn processing. *arXiv preprint arXiv:2108.04773*, 2021. 4

[24] 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*, pages 2704–2713, 2018. 1, 13

[25] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh

Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017. 1

[26] 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. 2, 3, 5, 11

[27] Jangho Kim, KiYoon Yoo, and Nojun Kwak. Position-based scaled gradient for model quantization and pruning. *Advances in Neural Information Processing Systems*, 33:20415–20426, 2020. 3, 4, 5, 13

[28] Junghyup Lee, Dohyung Kim, and Bumsub Ham. Network quantization with element-wise gradient scaling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6448–6457, 2021. 3, 4, 5, 11, 13

[29] Zewei Mo, Zejia Lin, Xianwei Zhang, and Yutong Lu. motuner: a compiler-based auto-tuning approach for mixed-precision operators. In *Proceedings of the 19th ACM International Conference on Computing Frontiers*, pages 94–102, 2022. 1

[30] Maarten Molendijk, Floran de Putter, Manil Gomony, Pekka Jääskeläinen, and Henk Corporaal. Braintta: A 35 fj/op compiler programmable mixed-precision transport-triggered nn soc. *arXiv preprint arXiv:2211.11331*, 2022. 1

[31] Hieu Duy Nguyen, Anastasios Alexandridis, and Athanasios Mouchtaris. Quantization aware training with absolute-cosine regularization for automatic speech recognition. In *Interspeech*, pages 3366–3370, 2020. 3, 4, 5, 13

[32] Eunhyeok Park and Sungjoo Yoo. Profit: A novel training method for sub-4-bit mobilenet models. In *European Conference on Computer Vision*, pages 430–446. Springer, 2020. 2, 3, 5, 7, 8

[33] Matteo Risso, Alessio Burrello, Giuseppe Maria Sarda, Luca Benini, Enrico Macii, Massimo Poncino, Marian Verhelst, and Daniele Jahier Pagliari. Precision-aware latency and energy balancing on multi-accelerator platforms for dnn inference. *arXiv preprint arXiv:2306.05060*, 2023. 1

[34] Georg Rutishauser, Francesco Conti, and Luca Benini. Free bits: Latency optimization of mixed-precision quantized neural networks on the edge. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 1–5. IEEE, 2023. 1

[35] 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. 1, 5

[36] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019. 1

[37] Xingyou Song, Sagi Perel, Chansoo Lee, Greg Kochanski, and Daniel Golovin. Open source vizier: Distributed infrastructure and api for reliable and flexible black-box optimization. In *Automated Machine Learning Conference, Systems Track (AutoML-Conf Systems)*, 2022. 8, 12

[38] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1, 5

[39] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed precision dnns: All you need is a good parametrization. *arXiv preprint arXiv:1905.11452*, 2019. 1, 2, 3, 5

[40] Weier Wan, Rajkumar Kubendran, Clemens Schaefer, S Burc Eryilmaz, Wenqiang Zhang, Dabin Wu, Stephen Deiss, Priyanka Raina, He Qian, Bin Gao, et al. Edge ai without compromise: Efficient, versatile and accurate neurocomputing in resistive random-access memory. *arXiv preprint arXiv:2108.07879*, 2021. 1

[41] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019. 1, 2, 3, 5

[42] Yannan Nellie Wu, Joel S Emer, and Vivienne Sze. Accelergy: An architecture-level energy estimation methodology for accelerator designs. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019. 4

[43] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021. 1, 6

[44] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pages 581–590. IEEE, 2020. 2

[45] Amir Yazdanbakhsh, Kiran Seshadri, Berkin Akin, James Laudon, and Ravi Narayanaswami. An evaluation of edge tpu accelerators for convolutional neural networks. *arXiv preprint arXiv:2102.10423*, 2021. 1, 6

[46] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *arXiv preprint arXiv:2106.04560*, 2021. 1

[47] Yichi Zhang, Zhiru Zhang, and Lukasz Lew. Pokebnn: A binary pursuit of lightweight accuracy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2022. 4