Simulated Quantization, Real Power Savings

Mart van Baalen Brian Kahne Eric Mahurin Andrey Kuzmin Andrii Skliar
Markus Nagel Tijmen Blankevoort
Qualcomm AI Research*

{mart,bkahne,emahurin,akuzmin,askliar,markusn,tijmen}@qti.qualcomm.com

Abstract

Reduced precision hardware-based matrix multiplication accelerators are commonly employed to reduce power consumption of neural network inference. Multiplier designs used in such accelerators possess an interesting property: When the same bit is 0 for two consecutive compute cycles, the multiplier consumes less power. In this paper we show that this effect can be used to reduce power consumption of neural networks by simulating low bit-width quantization on higher bit-width hardware. We show that simulating 4 bit quantization on 8 bit hardware can yield up to 17% relative reduction in power consumption on commonly used networks. Furthermore, we show that in this context, bit operations (BOPs) are a good proxy for power efficiency, and that learning mixed-precision configurations that target lower BOPs can achieve better trade-offs between accuracy and power efficiency.

1. Introduction

Neural networks have achieved impressive breakthroughs in various fields, but are notoriously power hungry. In recent years, neural network quantization, i.e. running neural networks on specialized low bit-width integer hardware, has proven successful in reducing neural network power requirements [3], with lower bit-widths generally yielding proportionally reduced power consumption. However, designing and producing dedicated low bit-width hardware requires significant R&D investment.

To further increase efficiency, most matrix multiplication hardware accelerators split up large matrix multiplications into chunks of weights and activations. These chunks are calculated in dedicated multiply-accumulate (MAC) arrays, reducing power consumption as a result of data transfer through increased locality. If a bit in an individual MAC array multiplier unit is 0 for two or more consecutive cycles, a phenomenon we will from here on refer to as 'consecutive 0 bits', this multiplier consumes less power.

In this paper we show that this effect can be exploited to achieve power benefits from low bit-width quantization even on higher bit-width hardware, e.g. by running 4 bit quantized networks on 8 bit hardware. Furthermore, we show that power efficiency gains from simulated low bit-width quantization can be vastly increased by bit-shifting weights appropriately. Lastly, we show that model Bit Operation (BOP) count [15] correlates well with power consumption, and show that mixed precision configurations targeting low BOP count yield better trade-offs between accuracy and power consumption than fixed bit-width configurations.

2. Background

2.1. Quantization

To use efficient low bit integer hardware, a network must be quantized to an appropriate bit-width [4,7,10]. As an example, consider 8 bit quantization of a floating point tensor $\mathbf{X}$:

$$\mathbf{X} \approx \mathbf{X}_q \tag{1}$$

$$= s_x \cdot \mathbf{X}_{int} \tag{2}$$

$$\mathbf{X}_{int} = \texttt{clip}\left(\left\lfloor \frac{\mathbf{X}}{s_x} \right\rfloor, \texttt{int\_min}, \texttt{int\_max}\right), \tag{3}$$

where $(\texttt{int\_min}, \texttt{int\_max})$ is $(0, 255)$ for unsigned and $(-128, 127)$ for signed tensors, and s_w is a quantization scale, set heuristically or through gradient descent on some target loss function. The matrix-matrix product $\mathbf{WX}$ can then be approximated as follows [4, 10]:

$$\mathbf{WX} \approx \mathbf{W}_q\mathbf{X}_q \tag{4}$$

$$= s_w s_x \mathbf{W}_{int}\mathbf{X}_{int}. \tag{5}$$

Since only integer multiplications are required to compute the product $\mathbf{W}_{int}\mathbf{X}_{int}$, this can be run on efficient low-precision hardware. Note that we use matrix-matrix multiplications for illustrative purposes; the same principles hold

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

for convolutions and certain extensions such as per-channel quantization.

To improve quantized performance, quantization-aware training (QAT) is often employed [2, 4, 11]. In these approaches, the (non-differentiable) quantization operation is used in the forward pass, but ignored in the backward pass.

2.1.1 Mixed Precision quantization

The quantization procedure described in the previous section introduces noise to weight and activation tensors. Some neural network layers are more sensitive to this noise than other layers. It can therefore be beneficial to use *mixed precision quantization* (MPQ), i.e. using tensor specific bit-widths instead of a fixed bit-width for each tensor in the network. Since the resulting space of MPQ configurations grows exponentially with the number of layers in a network, precluding exhaustive search, several methods have been developed to efficiently find MPQ configurations with good efficiency vs accuracy trade-offs [1, 14–16].

2.2. Hardware accelerators

Computations for a neural network layer can be seen as a matrix-matrix multiplication $\mathbf{Y} = \mathbf{W}\mathbf{X}$ between input activations $\mathbf{X}$ and weights $\mathbf{W}$. The product $\mathbf{W}\mathbf{X}$ entails the multiplication of individual elements in $\mathbf{W}$ with elements in $\mathbf{X}$ and the addition (accumulation) of the resulting scalar products. Neural network accelerators are implemented in hardware as multiply-accumulate (MAC) arrays, in which a subset of weights is multiplied with a subset of activations in parallel each cycle, and subsequently accumulated [3].

For example, a MAC array taking 16 weights $\mathbf{W}_{1:4,1:4}$ and 4 activations $\mathbf{X}_{1:4,k}$ as input, would have 16 multipliers and 4 accumulators. In each cycle the scalar products $\mathbf{W}_{nm}\mathbf{X}_{mk}$ are computed in parallel, in multipliers $M_{nm}; n, m \in [1, 4]$. The (partial) results for $\mathbf{Y}_{1:4,k}$ are stored in the accumulators. In this paper we focus on 8 bit integer MAC arrays, in which all inputs are 8 bit integers.

2.2.1 Power savings with zeros

If one of the inputs to a multiplier M_{mn} is 0 for two or more consecutive cycles, this multiplier does not consume power. This phenomenon occurs at the bit level: if the same input bit in a multiplier is 0 for two consecutive cycles, this multiplier bit consumes no power.

This is effect caused by a multitude of factors, of which the most important are: reduced dynamic power which is associated with bit switching in the input; reduced output bit switching due to more multiplication by 0; and fewer active bits (i.e. those bits that were 1 in the current or previous cycle), yielding proportionally lower multiplier power consumption. We refer readers to, e.g., [6] for details.

3. Method

To reduce power consumption, we aim to increase consecutive 0 bits. In this section we show how this can be achieved by simulating low bit-width quantization in higher bit-width hardware. Since activation tensors in ReLU networks are more sparse than the corresponding weights, we assume that the biggest gains can be achieved by increasing the number of consecutive 0 bits in the weights. For this reason, we restrict our focus to the weights of networks with ReLU activations.

3.1. Simulating low bit-widths

Low bit-width weights can be simulated on higher bit-width hardware by restricting the range of (integer) values weights can take. For example, simulating 4 bit integer quantization on 8 bit hardware is commonly done by restricting the range of integer weight values to $[-8, 7]$. However, due to the sign extension as a result of two's complement encoding of negative numbers, negative values will have 1s in their most significant bits (MSBs). Since our aim is to maximize the number of consecutive 0 bits this is an undesirable property.

To avoid the sign extension in 2s complement encoding, we bit-shift the low bit integer weights by an appropriate amount. Note that bit-shifting an integer by b bits is equivalent to multiplying by 2^b. Generally, when simulating signed b bit quantization on B bit hardware, we can multiply each value by 2^{B-b}, effectively bit-shifting each value by $B - b$ bits. To offset the effect of bit-shifting, we adjust the quantization scale s accordingly: $\hat{s} = \frac{s}{2^{B-b}}$.

As an example, consider simulating signed 4 bit weights on 8 bit hardware. Assume we have a signed weight tensor $\mathbf{W}_{int}$ with values in the range $[-8, 7]$ and an associated scale s_w. The binary representation of the range $[-8, 7]$ is $[11111000, 00000111]$. In this case we only have consecutive 0 bits if consecutive values are either both positive or both negative numbers. To circumvent this we can define $\widehat{\mathbf{W}_q} = 2^4 \cdot \mathbf{W}_{int}$, which effectively shifts each value in $\mathbf{W}_{int}$ by 4 bits. This yields a range from $[-128, 112]$ in decimal; $[10000000, 01110000]$ in binary. Note that sinc resulting values are multiples of 16, the least significant 4 bits will now always be zero, independent of the represented values. Lastly, we can cancel out the effect of multiplication by 16 by using quantization scale $\widehat{s_w} = \frac{s_w}{16}$. The resulting integer matrix product remains mathematically equivalent:

$$\begin{aligned}\widehat{s_w} s_x \widehat{\mathbf{W}_{int}} \mathbf{X}_{int} &= \frac{s_w}{16} s_x 16 \mathbf{W}_{int} \mathbf{X}_{int} \\ &= s_w s_x \mathbf{W}_{int} \mathbf{X}_{int}.\end{aligned}$$

3.2. Power-aware Mixed Precision Quantization

As explained in section 2.1.1, different tensors may exhibit different levels of quantization noise sensitivity. For

this reason, using a fixed bit-width for all weight tensors may not give an optimal trade-off between accuracy and power efficiency. To find better trade-offs, we adapt Differentiable Quantization [14] (DQ), to use low Bit Operations [15] (BOPs) as regularization target instead of model size.

DQ training jointly learns model parameters θ and quantization parameters ϕ. The quantization parameters ϕ consist of the maximum quantization range m_i and quantization scale s_i, for each quantizer q_i in the network. From the combination of m_i and s_i the bit-width of quantizer q_i can be inferred as $\text{bits}(\phi_i) = \left\lceil \log_2 \left(\frac{m_i}{s_i} + 1 \right) \right\rceil + \text{signed}(q_i)$.

We then define $\text{BOPs}(\phi)$ as:

$$\text{BOPs}(\phi) = \sum_{op_i \in network} \text{bits}(\phi_i)\text{MAC}(op_i) \tag{6}$$

where op_i denotes operations in a network (e.g. convolutions and linear layers), $\text{MAC}(op_i)$ denotes the number of Multiply-Accumulate (MAC) operations for op_i, and $\text{bits}(\phi_i)$ the bit-width of the weight quantizer q_i associated with op_i. We ignore activation quantizers in BOP regularization since their bit-width is fixed to 8. Using this regularization target will put more emphasis on lower bit-widths for high-MAC count layers. Since weights of higher MAC-count layers are used more in computations, and lower (simulated) bit-widths imply more 0 bits, we expect the BOP target to correlate well with power consumption.

Our adapted DQ training objective is then:

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{CE}(\theta, \phi) + \lambda \cdot \text{BOPs}(\phi) \tag{7}$$

where $\mathcal{L}_{CE}(\theta, \phi)$ denotes the cross-entropy loss on (a batch of) training data as a function of model and quantization parameters, and λ controls regularization strength, with higher regularization yielding objectives that favor low BOP count over accuracy.

4. Experiments and results

To validate whether our approach reduces power consumption we perform on-device power measurements with native 8 bit and simulated low bit models. To this end we have access to a device with a Snapdragon® 888 Mobile Platform. This device is equipped with Embedded Power Measurement capabilities, which sample instantaneous power consumption on various power rails, at sub-millisecond intervals. We run all models in 8 bits with a Qualcomm® Hexagon™ Fused AI Accelerator and measure power on the rail which powers this accelerator[1].

[1]Snapdragon and Qualcomm Hexagon are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

Model	Weight bit-width				
	8	7	6	5	4
ResNet18	70.34	70.31	70.28	70.12	69.73
MobNetV2	71.76	71.64	71.46	71.11	70.17
ResNet50	76.62	-	-	-	75.89
EffNetLite0	75.26	-	-	-	70.64
RN18 (PTQ)	69.70	-	-	-	68.55

Table 1. Post-QAT ImageNet validation top-1 accuracies for the models under consideration, and PTQ accuracies for ResNet18. Activations are kept in 8 bit.

4.1. Power consumption experiments

For our low bit-width experiments we use ResNet18, ResNet50, MobileNetV2 [12] and EfficientNet-Lite-0 [13] for ImageNet classification. For all models we measure power on a model with 8 bit weights and 8 bit activations (W8A8) as baseline power measurement. We then measure power on the same model, with a W4A8 configuration. For this configuration we consider both default simulated 4 bit quantization (i.e. with weights in range [-8, 7]) as a baseline and our proposed bit-shifted simulated 4 bit quantization (weights in range [-128, 112] with step size 16). For ResNet18 and MobileNetV2 we consider all intermediate weight bit-widths (7, 6, 5 bits) as well. For all models and weight bit-width configurations we keep activations in 8 bits. We report power reductions relative to a model with 8 bit weights.

4.1.1 Model Preparation

Using naively quantized models could result in overestimating the potential power reductions of simulated low bit-width quantization. For example, in a W8A8 ResNet18 model quantized after FP32 training using min-max range estimation, 7% of weights are 0 and accuracy is 69.7%. If we quantize the same FP32 model to W4A8 using the same techniques, 84% of weights are 0, yielding lower power consumption as a result, while accuracy drops to 0.10%.

To ensure weight distributions are realistic, we perform 20 epochs of quantization-aware training (QAT) for each model and bit-width configuration [2, 5]. We start from models pre-trained in FP32 on ImageNet, and use learned step size quantization (LSQ) to learn quantization parameters [2]. We do not fold batch-normalization layers into the corresponding weight layers [4, 7], and thus implicitly use per-channel quantization. Following [11] we re-estimate the batch normalization running statistics using 100 batches prior to final evaluation. We run experiments on a range of learning rates and report results on the model that gave best validation accuracy. For Post-training quantization (PTQ) experiments on ResNet18 we use Data Free Quantization [9] for the 8 bit weight model and AdaRound [8] for the 4 bit weight model. Table 1 shows validation scores

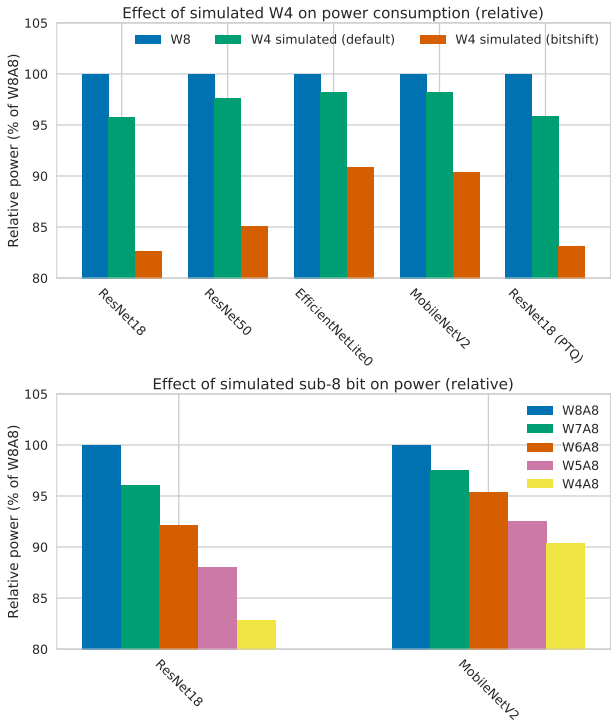

Figure 1. **Top:** Power consumption reduction relative to a W8A8 model for simulated low bit-widths on 8 bit hardware. The blue bar shows the W8A8 model, the green bar shows a W4A8 model simulated using the default method, the red bar shows the same W4A8 model simulated using our bit-shifted approach.
Bottom: Relative power consumption reduction for intermediate bit-widths on ResNet18 and MobileNetV2.

for each model. Note that these results are not novel; similar results have been reported in literature before (e.g. [8–10]).

4.1.2 Power consumption experiment results

Figure 1 (top) shows the power efficiency results obtained from simulated W4A8 models run 8 bit hardware, relative to a W8A8 model. In this plot we see that our bit-shifting approach provides significantly more reduction in relative power consumption compared to the default approach. There is no appreciable difference between the results for the QAT and PTQ ResNet18 models.

Figure 1 (bottom) shows that each (simulated) bit that is removed yields proportional reduction in power consumption. This implies that we can use non power-of-two bit-widths to find a wider range of trade-offs between power consumption reduction and accuracy.

4.2. Mixed precision experiments

We investigate whether mixed precision configurations can achieve better trade-offs between accuracy and power efficiency than the fixed weight bit-width models from the previous section. We use MobileNetV2 in our experiments

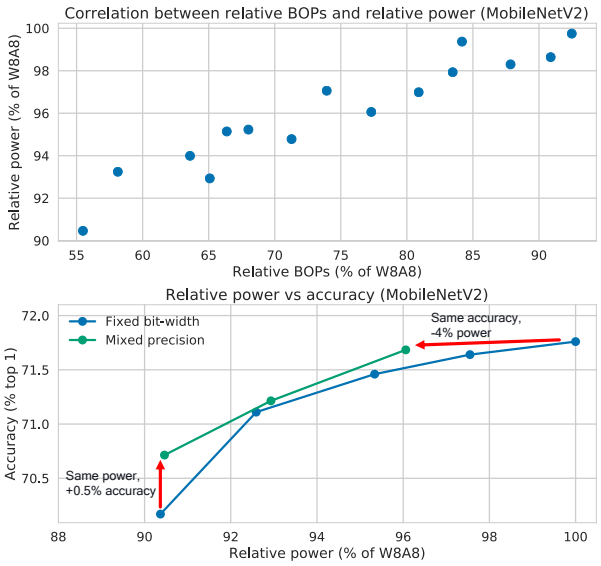

Figure 2. **Top:** Correlation between BOPs relative to W8A8 and power consumption relative to W8A8.
Bottom: Relative power consumption reduction vs accuracy for MobileNetV2 with fixed bit-width configurations (W4-8A8) and mixed precision configurations learned using BOP targeted DQ.

as it shows relatively strong accuracy degradation for low bit-widths. We run optimization with several values of λ for each network, and report results for the best learning rate for each λ.

4.2.1 Results for mixed precision experiments

Figure 2 (top) shows the correlation between relative BOPs and power compared to a W8A8 model, while Figure 2 (bottom) shows a comparison between mixed precision configurations and fixed bit-width configuration models. We see that relative BOPs correlate strongly with relative power consumption, and as a result differentiable quantization with a BOP regularizer can indeed find better trade-offs between accuracy and power consumption than fixed configuration counterparts, yielding 4% power consumption reduction at no accuracy loss, or a 0.5% increase in accuracy at no extra expense in power.

5. Conclusion

In this paper we provided reduced power consumption as a motivation for simulating low bit-width quantization on higher bit-width hardware. We introduced a novel bit-shifting approach to simulating low bit-widths that increases the number of consecutive 0 bits, and showed that this approach indeed shows greater reduction in power consumption than the default approach. Lastly, we showed that we can improve power efficiency further by selecting bit-width configurations that minimize bit operations while maintaining good accuracy.

References

[1] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019. 2

[2] 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

[3] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014. 1, 2

[4] 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, 2, 3

[5] Sambhav R Jain, Albert Gural, Michael Wu, and Chris Dick. Trained uniform quantization for accurate and efficient neural network inference on fixed-point hardware. *arXiv preprint arXiv:1903.08066*, 6:6, 2019. 3

[6] Shweta S Khobragade and Swapnili P Karmore. Low power vlsi design of modified booth multiplier. *Int. J. on Recent Trends in Engineering and Technology*, 9(1), 2013. 2

[7] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 1, 3

[8] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020. 3, 4

[9] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019. 3, 4

[10] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. 1, 4

[11] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. *arXiv preprint arXiv:2203.11086*, 2022. 2, 3

[12] 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. 3

[13] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. 3

[14] Stefan Uhlich, Lukas Mauch, Kazuki Yoshiyama, Fabien Cardinaux, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Differentiable quantization of deep neural networks. *arXiv preprint arXiv:1905.11452*, 2(8), 2019. 2, 3

[15] Mart van Baalen, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling. Bayesian bits: Unifying quantization and pruning. *arXiv preprint arXiv:2005.07093*, 2020. 1, 2, 3

[16] 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. 2