

Towards Bit-Shareable Inference on Microcontrollers

Charalampos Bezaits, Yaman Umuroglu, Di Liu, and Magnus Sjalander
 Norwegian University of Science and Technology
 FirstName.LastName@NTNU.no

Abstract—Many embedded systems are now being deployed in energy-constrained environments, with some systems utilizing energy-harvesting technologies. Consequently, the energy available to these systems is dynamic. For example, energy harvesting from the sun can provide excess energy during the daytime, but energy levels run low at night. In such energy-harvesting environments, low-power microcontroller (MCU) platforms are used to run machine learning inference, but their software is not adaptive to the energy fluctuations. **BitSIM** is the first to provide a clear methodology to train and deploy switchable-precision networks (SP-nets) that tackle the challenges of an MCU platform.

BitSIM employs a novel quantizer, **PolyQAT**, which not only enables weight-sharing but also bit-shareable weights. In bit-shareable weights, the narrower-precision weight can be directly extracted from the wider weight. With **PolyQAT**, SP-nets can be trained with low precision (i.e., with weights of four bits or less), which enables the deployment of large networks with respect to the memory size of MCUs. For the deployment of the SP-nets, **BitSIM** considers one minimalistic MCU hardware extension that enables efficient execution of sub-byte quantized neural networks.

I. INTRODUCTION

Machine learning at the edge requires efficient and compact models, which are commonly achieved through quantization [1]. Many edge devices are deployed in ever-changing environments with varying demands for latency and energy usage. An ideal machine learning algorithm should be adaptable to deliver the best possible accuracy given current environmental conditions. Several methods [2], [3] can reduce the computational demand at runtime but still require storing the entire model at full precision, which limits their applications on memory-constrained edge devices. Recent works [4]–[9] have investigated switchable-precision networks (SP-nets) where the precision of fully quantized weights can be adjusted at runtime, enabling a single set of compact weights to be stored.

During the execution of SP-nets, the same number of mathematical operations is performed, but at different precisions. Depending on the underlying hardware, narrower precision can provide faster execution and reduce memory-bandwidth usage. Low-power MCUs typically lack a floating-point unit (FPU), making it inefficient to store weights in the floating-point format, where the weights get quantized to different lower precisions on inference, see Figure 1 (*Store floats*).

Existing SP-nets [5]–[10] are not applicable for an MCU deployment. An MCU deployment needs to be *bit-shareable* and not only weight shareable. With *bit-shareable*, we mean that the narrower precision weight can be directly extracted from the stored weight by simply selecting a subset of the stored bits, as seen in Figure 1.

This work was funded in part by Institutional Research Gifts from Advanced Micro Devices (AMD), Inc.

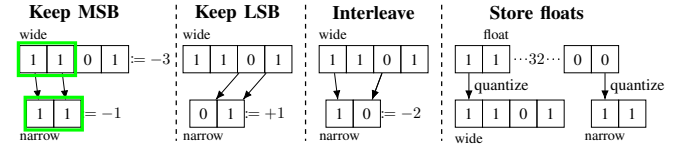


Fig. 1. Different ways of reducing the precision of two's-complement values and their decimal counterparts. *Bit-shareable* are: Keep MSB, Keep LSB, and Interleave. Storing the float is *weight-sharing*

We present a novel quantizer for *bit-shareable* weights that is not limited to a power-of-two scaling factor, and that can be trained with weights as small as two bits. Our quantizer can be applied to any quantization algorithm by introducing the additional steps shown in Figure 2, and aligns with state-of-the-art training methodologies for quantized neural networks [11]. We present results when deploying switchable-precision networks (SP-nets) with bit-shareable weights on a low-cost microcontroller (MCU).

II. BIT-SHAREABILITY TEST

Several previous works [5]–[9] present quantization strategies with switchable-precision. For such strategies to be beneficial for MCU deployment, the weights need to be bit-shareable. For bit-shareability, the narrow weights should be solely defined by a subset of the bits from the original weights. Each narrow weight should be a partial overlap of the original weight. There are several ways to achieve a partial overlap as seen in Figure 1. Keeping the most significant bits (MSBs), as seen with green in Figure 1, is the *bit-shareable* partial overlap that we want to test a quantizer for.

We start from the binary representation of natural numbers \mathbb{N} . Let n represent the bit width needed to express the number. The representation can be defined by Eq. 1. Here, we use natural numbers to simplify the proof, but the same holds for integers.

$$N = b_n \times 2^n + b_{n-1} \times 2^{n-1} + \dots + b_1 \times 2^1 + b_0 \times 2^0 \quad (1)$$

Each bit can, then, be calculated with Eq. 2, where $\lfloor \cdot \rfloor$ represents the floor function.

$$b_i = (\lfloor \frac{N}{2^i} \rfloor) \bmod 2 = \lfloor \frac{N}{2^i} \rfloor - 2 \times \lfloor \frac{N}{2^{i-1}} \rfloor \quad (2)$$

We use an apostrophe (e.g., b') to refer to a narrower bit-width weight. The narrower bit width is related to the original bit width based on the number of bit shifts sif , i.e., $n' = n - sif$. In Figure 1, 2-bit weights are extracted from 4-bit weights. So, $sif = 2$, $n = 4$, and $n' = 4 - 2 = 2$. The same relation can be applied to the bit position $i' = i - sif$ as the narrow bits are calculated by shifting the wide bits.

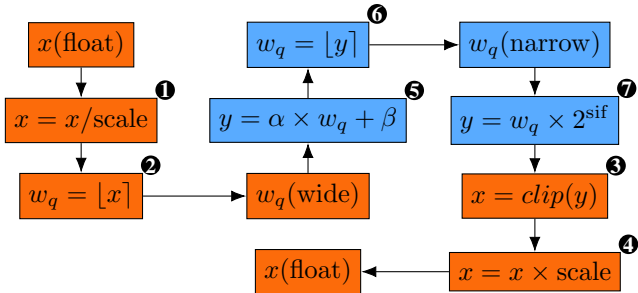


Fig. 2. Computational graph for `PolyQAT`. The blue color shows our `PolyQAT` additions, which enable bit-shareable aware training, while the orange depicts the original learned step size quantization (LSQ).

For *bit-shareability*, the bits after shifting must be equal for all the positions of the narrow weight. In Figure 1 with green, the first and second bit positions of the 2-bit narrow weight correspond to the third and fourth bit positions of the wide weight. This equality can, in general, be expressed as in Eq. 3.

$$\forall i \in \{sif, sif + 1, \dots, n\}, \quad b_i = b'_{i-sif} \quad (3)$$

If a quantizer produces wide and narrow weights that satisfy Eq. 3, then the quantizer fulfills the *bit-shareability* test.

`AdaBits` employs two quantizers, *original* and *modified* [5]. The *original* quantizer needs to store the floating-point versions to recover the quantized weights. The *modified* quantizer can generate weights with *bit-shareability*, but it is unable to converge for four bits and below. `InstantNet` [6], `Any-Precision` [7], `Bit-mixer` [9], and `CoQuant` [8] do not satisfy Eq. 3, which is crucial for achieving bit-shareability.

III. POLYQAT QUANTIZER

Quantization-aware training (QAT) methods typically use a quantizer based on learned stepsize quantization (LSQ) [11], especially for low bit widths [12]. LSQ introduces the concept of the learnable scale to quantize weights, as shown in orange in Figure 2 with steps ①, ②, ③, and ④. `PolyQAT` extends LSQ with the use of a polynomial function and a second round. The polynomial is fitted between the original wide quantized weight values and the narrow weights ⑤. After the division by the scale is applied and the first rounding is performed, the polynomial is used, bringing the weights close to the narrow, quantized weights. The second round aligns the result exactly to the narrow, quantized weights ⑥. Next, the result is shifted back towards their original value by multiplying with the power-of-two shift ⑦. The computational graph of `PolyQAT` can be seen in Figure 2. In total `PolyQAT` adds three *operations*:

$$y = \alpha \times w_q + \beta \quad \text{⑤} \quad w_q = [y] \quad \text{⑥} \quad y = w_q \times 2^{sif} \quad \text{⑦}$$

The polynomial ⑤ was chosen because it is differentiable, making backpropagation feasible. It also enables bit-shareable aware training of any scale as it is applied on the quantized weights. The second round ⑥, while it introduces another straight-through estimator (STE), it enables strong performance. Shifting the weights back towards the original values reduces the error between them ⑦ and improves the training convergence for low bit widths.

TABLE I
ACHIEVED ACCURACY (IN %) OF CIFAR10 AND CIFAR100

| Model | Switchable | CIFAR10 | | CIFAR100 | |
|----------|------------|---------|-------|----------|-------|
| | | 4A4W | 2A2W | 4A4W | 2A2W |
| ResNet8* | No | 92.31 | 90.90 | 67.46 | 55.87 |
| ResNet8* | Yes | 91.96 | 90.98 | 66.02 | 61.06 |

* First and last layer at 8-bit precision

TABLE II
CYCLES ON A SIMD-ALU MCU

| Quant. | First | Non-MAC | Conv | Final | Sum |
|--------|-------|---------|--------|-------|--------|
| 4A4W | 1.5M | 8.9M | 292.2M | 0.82M | 320M |
| 2A2W | 1.5M | 16.3M | 152.1M | 0.82M | 170.6M |

A first-degree polynomial is sufficient to align the original quantized weights for the rounding to correctly *move* them into place. For example, to *move* from a 4-bit signed wide precision to a 2-bit signed narrow precision, the values of α and β are 0.24 and -0.4 , respectively. `PolyQAT` can train concurrently two (or more) different precisions by introducing a loss function over the different precisions as $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{wide}} + \mathcal{L}_{\text{narrow}}$.

TABLE III
RELATED WORK COMPARISON ON CIFAR10

| Name | Bit-Shareable Weights | Model Size ResNet | First/Last Layer Bit width | 4A4W | 2A2W |
|-----------------------|-----------------------|-------------------|----------------------------|-------|-------|
| CoQuant* [8] | NO | 18 | 32 | 95.1 | 94.1 |
| Any-Precision* [7] | NO | 20 | 32 | 93.95 | 93.97 |
| MultiQuant* [13] | NO | 20 | 8 | 91.87 | 89.83 |
| AdaBits-original+ [5] | NO | 18 | 8 | 90.5 | 88.2 |
| AdaBits-modified+ [5] | YES | 18 | 8 | 43.2 | 10 |
| BitSIM | YES | 18 | 8 | 93.47 | 90.65 |

*indicates our experiments from released code

*indicates accuracy results as presented in the original work

IV. RESULTS

`ResNet-8` (RN-8) deployed with `PolyQAT` achieves the accuracies on the CIFAR datasets shown in Table I. The RN-8 parameters need 0.73 MiB for 4-bit and 0.43 MiB for 2-bit quantization. Both sets of weights do not fit within a typical 1 MiB MCU Flash, especially when accounting for the operating system and other system configurations, unless switchable-precision with *bit-shareability* is applied.

For the deployment of SP-nets, we consider a variable-precision SIMD unit presented by `XpulpNN` [14]. `XpulpNN` can execute efficient inference at 4A4W and 2A2W, and offer less latency on the narrower bit width, while only increasing the core power consumption by 24% [15]. The results in cycles from executing RN-8 are shown in Table II. Transposing the weights from the 4-bit configuration to the 2-bit is also needed and counted in the Non-MAC operations.

Finally, we compare `BitSIM` with four alternative SP-net implementations in Table III. The results show that `BitSIM` achieves comparable results to the state-of-the-art while also being bit-shareable.

REFERENCES

- [1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *JMLR*, vol. 18, pp. 1–30, 2018. [Online]. Available: <https://dl.acm.org/doi/10.5555/3122009.3242044>
- [2] H. Rahmath P, V. Srivastava, K. Chaurasia, R. G. Pacheco, and R. S. Couto, "Early-exit deep neural network - a comprehensive survey," *ACM Computing Surveys*, vol. 57, p. 75:1–75:37, Nov. 2024. [Online]. Available: <https://doi.org/10.1145/3698767>
- [3] J. Yu and T. S. Huang, "Universally slimmable networks and improved training techniques," in *Proceedings of the IEEE/CFV International Conference on Computer Vision*, 2019, pp. 1803–1811. [Online]. Available: <https://doi.org/10.1109/ICCV.2019.00189>
- [4] L. Guerra, B. Zhuang, I. Reid, and T. Drummond, "Switchable precision neural networks," Feb. 2020. [Online]. Available: <http://arxiv.org/abs/2002.02815>
- [5] Q. Jin, L. Yang, and Z. Liao, "AdaBits: Neural network quantization with adaptive bit-widths," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2146–2156. [Online]. Available: <https://doi.org/10.1109/CVPR42600.2020.00222>
- [6] Y. Fu, Z. Yu, Y. Zhang, Y. Jiang, C. Li, Y. Liang, M. Jiang, Z. Wang, and Y. Lin, "InstantNet: Automated generation and deployment of instantaneously switchable-precision networks," in *Proceedings of the ACM/IEEE Design Automation Conference*, Dec. 2021, pp. 757–762. [Online]. Available: <https://doi.org/10.1109/DAC18074.2021.9586254>
- [7] H. Yu, H. Li, H. Shi, T. S. Huang, and G. Hua, "Any-precision deep neural networks," in *Proceedings of the AAI Conference on Artificial Intelligence*, May 2021, pp. 10763–10771. [Online]. Available: <https://doi.org/10.1609/aaai.v35i12.17286>
- [8] X. Sun, R. Panda, C.-F. R. Chen, N. Wang, B. Pan, A. Oliva, R. Feris, and K. Saenko, "Improved techniques for quantizing deep networks with adaptive bit-widths," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 957–967. [Online]. Available: <https://doi.org/10.1109/WACV57701.2024.00100>
- [9] A. Bulat and G. Tzimiropoulos, "Bit-Mixer: Mixed-precision networks with runtime bit-width selection," in *Proceedings of the IEEE/CFV International Conference on Computer Vision*, Oct. 2021, pp. 5168–5177. [Online]. Available: <https://doi.org/10.1109/ICCV48922.2021.00514>
- [10] C. Tang, H. Zhai, K. Ouyang, Z. Wang, Y. Zhu, and W. Zhu, "Arbitrary bit-width network: A joint layer-wise quantization and adaptive inference approach," in *Proceedings of the ACM International Conference on Multimedia*, Oct. 2022, p. 2899–2908. [Online]. Available: <https://doi.org/10.1145/3503161.3548001>
- [11] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," May 2020. [Online]. Available: <http://arxiv.org/abs/1902.08153>
- [12] I. Colbert, A. Pappalardo, and J. Petri-Koenig, "A2Q: Accumulator-aware quantization with guaranteed overflow avoidance," in *Proceedings of the IEEE/CFV International Conference on Computer Vision*, 2023, pp. 16943–16952. [Online]. Available: <https://doi.org/10.1109/ICCV51070.2023.01558>
- [13] K. Xu, Q. Feng, X. Zhang, and D. Wang, "MultiQuant: Training once for multi-bit quantization of neural networks," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Jul. 2022, pp. 3629–3635. [Online]. Available: <https://doi.org/10.24963/ijcai.2022/504>
- [14] A. Garofalo, G. Tagliavini, F. Conti, D. Rossi, and L. Benini, "XpulpNN: Accelerating quantized neural networks on RISC-V processors through ISA extensions," in *Proceedings of the International Conference on Design, Automation and Test in Europe*, Mar. 2020, pp. 186–191. [Online]. Available: <https://doi.org/10.23919/DAT48585.2020.9116529>
- [15] D. Metz, V. Kumar, and M. Sjalander, "BISDU: A bit-serial dot-product unit for microcontrollers," *ACM Transactions on Embedded Computing Systems*, vol. 22, p. 79:1–79:22, Sep. 2023. [Online]. Available: <https://doi.org/10.1145/3608447>