Free Bits: Latency Optimization of Mixed-Precision Quantized Neural Networks on the Edge

Free Bits: Latency Optimization of Mixed-Precision Quantized Neural Networks on the Edge

Georg Rutishauser*, Francesco Conti†, Luca Benini*†
*Departement Informationstechnologie und Elektrotechnik, ETH Zürich, Switzerland
†Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione, Università di Bologna, Bologna, Italy
*{georgr,lbenini}@iis.ee.ethz.ch †f.conti@unibo.it

***Abstract*—Mixed-precision quantization, where a deep neural network's layers are quantized to different precisions, offers the opportunity to optimize the trade-offs between model size, latency, and statistical accuracy beyond what can be achieved with homogeneous-bit-width quantization. To navigate the intractable search space of mixed-precision configurations for a given network, this paper proposes a hybrid search methodology. It consists of a hardware-agnostic differentiable search algorithm followed by a hardware-aware heuristic optimization to find mixed-precision configurations latency-optimized for a specific hardware target. We evaluate our algorithm on MobileNetV1 and MobileNetV2 and deploy the resulting networks on a family of multi-core RISC-V microcontroller platforms with different hardware characteristics. We achieve up to $28.6\,\%$ reduction of end-to-end latency compared to an 8-bit model at a negligible accuracy drop from a full-precision baseline on the 1000-class ImageNet dataset. We demonstrate speedups relative to an 8-bit baseline, even on systems with no hardware support for sub-byte arithmetic at negligible accuracy drop. Furthermore, we show the superiority of our approach with respect to differentiable search targeting reduced binary operation counts as a proxy for latency.**

***Index Terms*—Edge AI, Mixed-Precision Neural Networks**

I. Introduction

The number of internet of things (IoT) devices deployed is growing rapidly and is projected to reach 19.1 billion by 2025 [1]. To efficiently and accurately process the massive amounts of data collected by IoT sensor nodes under the strict latency and power constraints imposed by IoT applications, the emerging field of Edge AI aims to deploy deep learning (DL) algorithms directly on the edge devices that collect them. Microcontroller units (MCUs) have been a popular target for edge deployment of Deep Neural Networks (DNNs) due to their ubiquity and low cost, and extensive research has been conducted into designing efficient DNN models and techniques to enable inference on MCUs [2], [3]. As the active power consumption of edge nodes is generally dominated by components other than the arithmetic units, the most effective way to decrease the full-system inference energy on a given system is by reducing the inference latency while meeting their tight memory and storage constraints.

A key technique to reduce both memory footprint and inference latency of DNNs is *quantization*, where model parameters and intermediate activations are represented in low-precision formats. 8-bit quantized models generally exhibit equivalent accuracy to full-precision models. Thanks to Single Instruction Multiple Data (SIMD) instructions, these models can be executed on modern MCU-based system with lower latency and correspondingly reduced energy cost [4]. Quantization to even lower bit-widths has also seen widespread interest [5], [6], [7] and the hardware community has followed suit, proposing low-precision DNN execution engines as well as instruction set architecture (ISA) extensions to accelerate networks quantized to sub-byte precision [8].

However, homogeneous quantization to sub-byte precisions often incurs a non-negligible accuracy penalty. To find the best trade-off between execution latency and statistical accuracy, *mixed-precision quantization* proposes to quantize different parts (usually at the granularity of individual layers) of the network to different precisions. In order to efficiently navigate the intractable search space of precision configurations of a given model, multiple works have applied differentiable neural architecture search (DNAS) to mixed-precision search. These approaches generally rely on a proxy for latency, such as binary operation (BOP) count, to guide the search [9], [10], [11]. By modeling quantization to different precisions in a differentiable manner and adding a regularizer term to the loss to penalize high BOP counts, these algorithms jointly minimize networks' BOP count and task accuracy. The trade-off between operational complexity and accuracy is controlled by the regularization strength.

However, as Figure 1 shows, low BOP counts do not directly translate to reduced execution latency on real hardware platforms, which motivates our work. Depending on the target platform, certain layers even exhibit a higher latency when quantized to lower precisions. One reason for this counter-intuitive behavior is in the hardware implementation of low-precision operations. E.g., the XpulpNN ISA extension used by [8] only supports operands of equal precision - when weight and activation precisions differ, the lower-precision operands must be unpacked to the larger data format. In systems with hierarchically organized memory, tiling effects also shape the precision-latency landscape: larger tiles lead to more efficient execution, with the consequence that some layers exhibit lower latency in sub-8-bit precision on systems without hardware support for sub-8-bit arithmetic (XPulpV2 in Figure 1). Past works have targeted the search for mixed-precision networks for deployment to MCU-class platforms. [12] focuses on reducing the memory/storage footprints, but did not consider inference latency. [13] targets inference energy reduction with a DNAS-based approach to channel-wise

This work is funded in part by the Convolve project evaluated by the EU Horizon Europe research and innovation programme under grant agreement No. 101070374 and has been supported by the Swiss State Secretariat for Education Research and Innovation under contract number 22.00150.

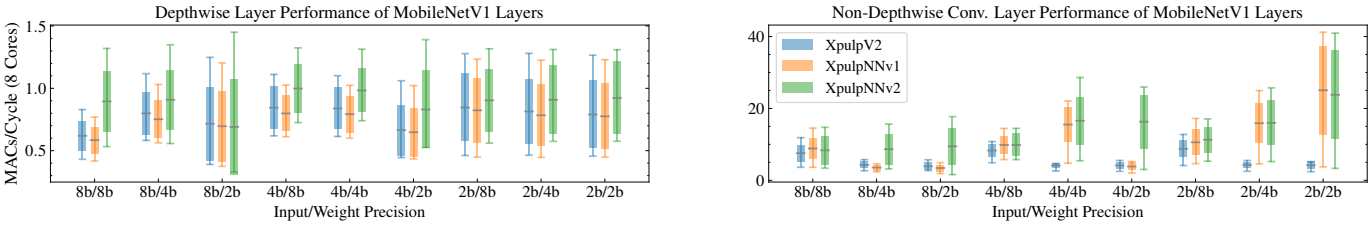

Fig. 1: Throughput vs. precision of MNv1 layers on systems implementing XpulpV2, XpulpNNV1 and XpulpNNV2 ISA extensions. Candlesticks' bodies are centered around the mean throughput and extend to one standard deviation above and below it. The ends of the wicks represent the minimum and maximum throughput of all layers.

quantization of convolutional layers. However, the channel-wise approach requires a more complex runtime to enable deployment, and we achieve equivalent results with layer-wise quantization on a much more challenging dataset than the MLPerf Tiny benchmark tasks used in [13].

In this paper, *we propose a mixed-precision latency optimization method consisting of a hardware-agnostic differentiable search step followed by a hardware-aware, profiling-based heuristic which both reduces execution latency and improves accuracy by increasing the precision in layers where higher precisions achieve lower latency.* We use Bayesian Bits [9] for the first step, but our method is not specific to Bayesian Bits and any mixed-precision search method could be used in its stead.

In evaluations on MobileNetV1 (MNv1) and MobileNetV2 (MNv2), deployed to a cycle-accurate RISC-V multi-core simulator, our approach results in an accuracy-latency trade-off curve that dominates those produced by differentiable search alone. To the best of our knowledge, we demonstrate for the first time end-to-end deployment of mixed-precision networks to an MCU-class platform that exhibit not only a reduced memory footprint but also reduced execution latency by up to $28.6\,\%$ at full-precision equivalent classification accuracy. Our key contributions are the following:

- We present a lightweight method to find latency-optimized mixed-precision quantization configurations for DNNs, consisting of a hardware-agnostic differentiable model search and hardware-aware heuristics, allowing efficient generation of optimized configurations for different platforms.
- We compose an end-to-end flow consisting of precision search, training, generation of integerized models and deploy the found configurations on a cycle-accurate simulator for high-performance RISC-V MCU systems.
- We analyze the resulting accuracy-latency trade-offs, demonstrating a reduction of end-to-end latency by up to $28.6\,\%$ vs. 8-bit quantization at full-precision equivalent classification accuracy and finds Pareto-dominant configurations with respect to homogeneous 4-bit quantization.

II. Free Bits

Free Bits is a multi-step method to find mixed-precision configurations of DNNs optimized for low latency on a given target hardware platform. In the first step, we employ two variants of the Bayesian Bits algorithm to find baseline reduced-precision configurations of the targeted network architecture.

In the second step, we use layer-wise profiling data collected on the target platform to update the initial configuration by increasing the precision of layers which exhibit lower latency in higher precisions. As these increases in precision are expected to improve both latency and statistical accuracy, we name this step the *free bits* heuristic.

From the configurations found in the second step, the one which best meets a given latency target is then selected and fine-tuned using a modified version of the Trained Quantization Thresholds (TQT) algorithm [14] and automatically converted to an integer-only model, which can be fed to a deployment backend for the target platform.

A. Differentiable Mixed-Precision Search

To find the initial mixed-precision configurations, we apply two variants of the Bayesian Bits algorithm. Bayesian Bits decomposes the quantization of each layer into the contributions from each of the allowed bit-widths and aims to reduce a network's total BOP count with a regularizer that penalizes each precision's contribution to the expected BOP count individually. Because the execution latency of a layer does not necessarily increase monotonously with precision and depends jointly on input and weight precisions, Bayesian Bits cannot target latency reduction directly. In addition to the original Bayesian Bits algorithm, we also employ a modified version enforcing equal input and weight precisions. This modification accounts for the fact that on our target platforms, the theoretical throughput for a layer with non-equal activation and weight precisions is bounded by the higher of the two precisions.

B. Free Bits Heuristic

As Figure 1 shows, there are many cases where a given layer does not profit from reduced precision, but in fact exhibits higher latency when executed in a lower precision. The *free bits* heuristic exploits this observation, relying on two core ideas: First, we assume our target platform executes networks layer-by-layer, which implies $L_{net} \approx \sum_{i=1}^{N} L_i$ for the total execution latency L_{net} of an N-layer network where the i-th layer is executed with latency L_i. Second, increasing a layer's input activation or weight precision never decreases the network's statistical accuracy.

Following the first assumption, we characterize each unique linear operator in the target network as a *layer type (LT)*,

Algorithm 1 Free Bits Heuristic

Input:

LD: Latency dictionary mapping LT t and precisions (b_{in}, b_{wt}) to a measured latency

C_{net}: Dictionary of LTs and precisions describing a MP network, of the form $\left\{i : \left(t^i, \left(b^i_{in}, b^i_{wt}\right)\right)\right\}_{i=1}^{N}$

P_{all}: Set of allowed combinations (b_{in}, b_{wt}) of input and weight precisions

Output:

C'_{net}: Latency-optimized MP configuration of input network

function HIGHER($(b_{in,1}, b_{wt,1}), (b_{in,2}, b_{wt,2})$)
return $(b_{in,1} \geq b_{in,2}) \wedge (b_{wt,1} \geq b_{wt,2})$
end function
$C' \leftarrow C$
for all $i, c^i = \left(t^i, \left(b^i_{in}, b^i_{wt}\right)\right) \in C_{net}$ **do**
$lat^i_0 \leftarrow LD\left[c^i\right]$ ▷ Initial latency
$cdts \leftarrow \{(b_{in}, b_{wt}) \mid (b_{in}, b_{wt}) \in P_{all}, LD\left[\left(t^i, (b_{in}, b_{wt})\right)\right] \leq lat^i_0, \text{HIGHER}((b_{in}, b_{wt}), (b^i_{in}, b^i_{wt}))\}$ ▷ Select lower-lat. configs with higher precisions
$best \leftarrow \underset{(b_{in}, b_{wt}) \in cdts}{\arg\min} LD[(t^i, (b_{in}, b_{wt}))]$ ▷ Select lowest-lat. candidate
$C'_{net}[i] \leftarrow (t^i, best)$ ▷ Update net configuration
end for

the tuple of all quantities that parametrize the invocation of a computational kernel, such as input dimensions, number of channels, or kernel size. For each LT occurring in the network, we profile the execution latency on the target platform for all supported precision configurations.We then update every layer in the network found by Bayesian Bits to the configuration of higher or equal precision that exhibits the lowest latency. By the two assumptions above, the resulting network's execution latency and statistical accuracy will be upper-bounded and lower-bounded, respectively, by those of the configuration found by Bayesian Bits. As it is expected to produce strictly superior configurations in terms of latency and statistical accuracy, we call this procedure the *free bits* heuristic. We show a pseudocode description of the procedure in Algorithm 1.

C. Quantization-Aware Fine-Tuning and Deployment

Having arrived at a latency-optimized mixed-precision configuration, we perform Quantization-Aware Training (QAT) with the QuantLabframework [1] to fine-tune the network's parameters using a generalized version of TQT [14], differing from the original algorithm in that we do not force clipping bounds to be exact powers of two.

III. Results

A. Experimental Setup

We performed experiments on the well-known and widely used MobileNetV1 and V2 architectures, applying the procedure proposed in Section II to MNv1 [15] and MNv2 [16]. We used width multipliers of 0.75 for MNv1 and 1.0 for MNv2. The input resolution was 224×224 for both networks. We trained our networks on the ILSVRC2012 [17] 1000-class dataset and report top-1 classification accuracies on the validation set.

a) Differentiable Mixed-Precision Search and QAT Fine-Tuning: We applied the two variants of Bayesian Bits described in Section II-A to the MNv1 and MNv2 network topologies. The configurations produced by our algorithm (as well as those produced by Bayesian Bits in the case of MNv1) were fine-tuned with TQT. In accordance with the capabilities of our hardware targets (see below), the precisions Bayesian Bits can select from are 2, 4, and 8 bits for both weights and activations.

b) Profiling, Deployment and Hardware Targets: QuantLab's automated integerization flow generates precision-annotated, integer-only ONNX models, which are consumed by the DORY [18] deployment backend. DORY generates C code leveraging a mixed-precision kernel library, which we run on GVSOC, a cycle-accurate, open-source simulator for multi-core RISC-V systems. The platforms we target are open-source RISC-V MCUs of the parallel ultra-low-power (PULP) family and are divided into two main domains. The System-on-Chip (SoC) domain contains a RISC-V core serving as the fabric controller, 512 KiB of L2 memory and a full set of peripherals. The cluster domain hosts 8 high-performance RISC-V cores operating on 64 KiB of high-bandwidth L1 scratchpad memory. This hierarchical memory structure necessitates *tiled* execution of a network's layers with each tile's inputs, outputs, and weights fitting into the L1 scratchpad. Tiling is automatically performed by DORY.

All cores in the target system implement the base RV32IMF ISA and the custom XpulpV2 extensions. We measure latency on three systems, with varying degrees of support for sub-byte arithmetic in the cluster cores. The *XpulpV2* system's cluster implements only XpulpV2, which supports only 8-bit SIMD arithmetic. The *XpulpNNv1* system implements the XpulpNN extension also used in [8], which provides support for packed-SIMD sub-byte arithmetic on 2- and 4-bit data. Because XpulpNN's arithmetic instructions require operands to have equal bit-width, mismatching activation and weight precisions require lower-precision data to be unpacked in software to the higher precision. Finally, the *XpulpNNv2* system eliminates this overhead by performing the unpacking transparently in Hardware. To generate the profiling data (shown in Figure 1 for MNv1) used by the free bits heuristic, we again use DORY to generate and export dummy networks for all layer types in all precision configurations.

B. Latency-Accuracy Trade-Offs for XpulpNNv1

a) MobileNetV1: Figure 2 shows the latency-accuracy trade-off for MNv1 deployed to a PULP system with the XpulpNNv1 ISA extensions, with the effect of the free bits heuristic indicated. We observe that the original Bayesian Bits algorithm generally does not produce low-latency configurations due to the reasons discussed in Section I.With two exceptions, applying the free bits heuristic improves the latency of all configurations substantially while increasing classification accuracy. For the 4 b/4 b baseline, the heuristic

[1] https://github.com/pulp-platform/quantlab/tree/georgr/bayesian_bits_gh

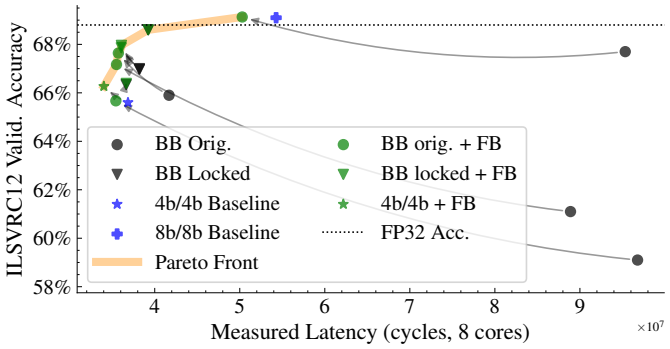

Fig. 2: Latency-Accuracy tradeoff of Bayesian Bits-trained MobileNetV1 configurations before and after applying the free bits heuristic. Grey arrows indicate the effect of the heuristic. **BB Orig./Locked**: Configurations found by the original Bayesian Bits algorithm and the modified version enforcing symmetric activation/weight precisions, respectively. **FB**: Free bits

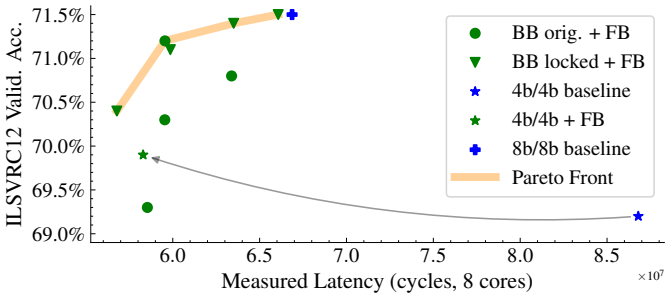

Fig. 3: Latency-Accuracy tradeoff of MobileNetV2 configurations optimized for XpulpNNv1. The grey arrow indicates the effect of the heuristic on the 4b/4b baseline.

increases the precision of 12 layers, improving latency and accuracy by 7% and 0.7 percentage points, respectively. As symmetric activation and weight precisions are theoretically optimal for XpulpNNv1's hardware implementation of sub-byte arithmetic, this is a non-trivial result. The free bits heuristic lifts the previously uncompetitive configurations found by the original Bayesian Bits algorithm to the Pareto front, yielding accuracy and latency gains of $1.4 - 6.6$ percentage points and $12.3\% - 61.6\%$, respectively. The most accurate configuration matches the 8b/8b baseline in statistical accuracy at 69.1% and reduces execution latency by 7.6%, and the configuration at the Pareto front's knee point improves execution latency by 27.9% at a classification accuracy within 0.2 percentage points of the full-precision baseline of 68.8%.

b) MobileNetV2: Figure 3 shows the latency-accuracy trade-off of MNv2 configurations produced by Bayesian Bits modified with the free bits heuristic running on the XpulpNNv1 system. The baseline 4b/4b configuration contains many asymmetric-precision convolutional layers due to adder node outputs being quantized to 8 bits. This leads to a latency higher than that of the 8b/8b baseline, which the free bits heuristic reduces by 46% while improving classification accuracy by 0.6 percentage points. Nevertheless, the resulting configuration is not Pareto-optimal with respect to those produced by our algorithm. In particular, the locked-precision version of Bayesian Bits, when combined with the free bits heuristic, produces configurations that dominate both baselines. The configuration at the Pareto front's knee point reduces execution latency by 10.9% at an accuracy penalty of only 0.3 percentage points from the 8b/8b baseline.

Acc. Margin	ISA	MobileNetV1		MobileNetV2	
		Lat. vs. 8b	Acc.	Lat. vs. 8b	Acc.
8b Baseline	*all*	+0%	69.1%	+0%	71.5%
0.5 pp.	XPv2	−5.5%	69.3%	−3.4%	71.0%
	XPNNv1	−27.9%	68.6%	−10.9%	71.2%
	XPNNv2	−28.6%	68.6%	−15.3%	71.0%
1.5 pp.	XPv2	−5.5%	69.3%	−6.3%	70.7%
	XPNNv1	−34.4%	67.6%	−15.1%	70.4%
	XPNNv2	−35.1%	67.6%	−15.3%	71.0%
4b + FB	XPv2	−3.5%	67.7%	−7.7%	70.9%
	XPNNv1	−37.1%	66.3%	−12.8%	69.9%
	XPNNv2	−39.8%	66.6%	−25.7%	69.6%
4b Baseline	XPv2	+49.9%	65.6%	+37.0%	69.3%
	XPNNv1	−32.3%	65.6%	+48.9%	69.3%
	XPNNv2	−38.3%	65.6%	−23.4%	69.3%

TABLE I: Configurations within margins of 0.5 and 1.5 percentage points (**pp.**) of 8b/8b classification accuracy for PULP systems implementing different ISA extensions: XpulpV2 (**XPv2**), XpulpNNv1 (**XPNNv1**) and XpulpNNv2 (**XPNNv2**). **4b+FB**: target-specific free bits heuristic applied to homogeneously quantized 4b/4b network.

C. Free Bits Across Different Target Platforms

To evaluate the portability of our algorithm, we optimized MNv1 and MNv2 configurations found with Bayesian Bits for the three different PULP systems described in Section III-A. Table I shows the lowest-latency configurations within 0.5 and 1.5 percentage points of classification accuracy of the 8b/8b baseline. Notably, our approach achieves latency reductions even on the XpulpV2 system without hardware support for sub-byte arithmetic, which can be attributed to a lower data movement overhead thanks to larger tile sizes.

IV. Conclusion

In this paper, we have presented *Free Bits*, an efficient method to find latency-optimized mixed-precision network configurations for inference on edge devices. Taking advantage of the fact that, depending on the target platform, increasing input or weight precision may lead to lower execution latency, the method optimizes mixed-precision configurations found by the hardware-agnostic Bayesian Bits differentiable search algorithm. Deploying the MNv1 and MNv2 configurations found with our algorithm on a family of high-performance MCU-class RISC-V platforms, we find that, i) with hardware support for sub-byte arithmetic, MNv1 end-to-end latency can be reduced by up to 30 % while retaining full-precision equivalent accuracy, ii) even without such hardware support, mixed-precision quantization enables a latency reduction of up to 7.7%, and iii) the found configurations offer a superior accuracy-latency trade-off with respect to homogeneous 4-bit and 8-bit quantization.

REFERENCES

[1] Statista, Inc., "Number of internet of things (iot) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030," https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/, May 2022.

[2] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. han, "Memory-efficient Patch-based Inference for Tiny Deep Learning," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[3] C. Banbury, C. Zhou, I. Fedorov, R. M. Navarro, U. Thakker, D. Gope, V. J. Reddi, M. Mattina, and P. Whatmough, "MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers," in *Proceedings of Machine Learning and Systems*, A. Smola, A. Dimakis, and I. Stoica, Eds., 2021, pp. 517–532.

[4] Arm Ltd., *Armv8-M Architecture Reference Manual*, Arm Ltd.

[5] J. Choi, S. Venkataramani, V. V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and Efficient 2-bit Quantized Neural Networks," in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 348–359.

[6] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights," in *International Conference on Learning Representations*, 2017.

[7] H. Alemdar, V. Leroy, A. Prost-Boucle, and F. Petrot, "Ternary neural networks for resource-efficient AI applications," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, may 2017, pp. 2547–2554.

[8] A. Garofalo, G. Ottavi, A. di Mauro, F. Conti, G. Tagliavini, L. Benini, and D. Rossi, "A 1.15 TOPS/W, 16-Cores Parallel Ultra-Low Power Cluster with 2b-to-32b Fully Flexible Bit-Precision and Vector Lockstep Execution Mode," in *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*. IEEE, sep 2021, pp. 267–270.

[9] M. van Baalen, C. Louizos, M. Nagel, R. A. Amjad, Y. Wang, T. Blankevoort, and M. Welling, "Bayesian bits: Unifying quantization and pruning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 5741–5752.

[10] Z. Cai and N. Vasconcelos, "Rethinking Differentiable Search for Mixed-Precision Neural Networks," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020, pp. 2346–2355.

[11] M. Nikolic, G. B. Hacene, C. Bannon, A. D. Lascorz, M. Courbariaux, Y. Bengio, V. Gripon, and A. Moshovos, "BitPruning: Learning Bitlengths for Aggressive and Accurate Quantization," *CoRR*, vol. abs/2002.0, 2020.

[12] M. Rusci, A. Capotondi, and L. Benini, "Memory-Driven Mixed Low Precision Quantization for Enabling Deep Network Inference on Microcontrollers," in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 326–335.

[13] M. Risso, A. Burrello, L. Benini, E. Macii, M. Poncino, and D. J. Pagliari, "Channel-wise Mixed-precision Assignment for DNN Inference on Constrained Edge Nodes," in *2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC)*, no. 101007321. IEEE, oct 2022, pp. 1–6.

[14] S. R. Jain, A. Gural, M. Wu, and C. H. Dick, "Trained Quantization Thresholds for Accurate and Efficient Fixed-Point Inference of Deep Neural Networks," in *Proceedings of Machine Learning and Systems*, 2020, pp. 112—-128.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *CoRR*, vol. abs/1704.0, apr 2017.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018, pp. 4510–4520.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[18] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, "DORY: Automatic End-to-End Deployment of Real-World DNNs on Low-Cost IoT MCUs," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1253–1268, aug 2021.