Value-Driven Mixed-Precision Quantization for Patch-Based Inference on Microcontrollers

Wei Tao[1], Shenglin He[1], Kai Lu[1], Xiaoyang Qu[2], Guokuan Li[1*], Jiguang Wan[1], Jianzong Wang[2*], Jing Xiao[2]
[1]Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China
[2]Ping An Technology (Shenzhen) Co., Ltd., Shenzhen, China

***Abstract*—Deploying neural networks on microcontroller units (MCUs) presents substantial challenges due to their constrained computation and memory resources. Previous researches have explored patch-based inference as a strategy to conserve memory without sacrificing model accuracy. However, this technique suffers from severe redundant computation overhead, leading to a substantial increase in execution latency. A feasible solution to address this issue is mixed-precision quantization, but it faces the challenges of accuracy degradation and a time-consuming search time. In this paper, we propose QuantMCU, a novel patch-based inference method that utilizes value-driven mixed-precision quantization to reduce redundant computation. We first utilize value-driven patch classification (VDPC) to maintain the model accuracy. VDPC classifies patches into two classes based on whether they contain outlier values. For patches containing outlier values, we apply 8-bit quantization to the feature maps on the dataflow branches that follow. In addition, for patches without outlier values, we utilize value-driven quantization search (VDQS) on the feature maps of their following dataflow branches to reduce search time. Specifically, VDQS introduces a novel quantization search metric that takes into account both computation and accuracy, and it employs entropy as an accuracy representation to avoid additional training. VDQS also adopts an iterative approach to determine the bitwidth of each feature map to further accelerate the search process. Experimental results on real-world MCU devices show that QuantMCU can reduce computation by 2.2x on average while maintaining comparable model accuracy compared to the state-of-the-art patch-based inference methods.**

***Index Terms*—microcontroller, deep learning, patch-based inference, outlier value, mixed-precision quantization search**

I. Introduction

Neural networks have been widely used in various fields. Microcontroller units (MCUs) are a cost-effective and energy-efficient solution for neural network inference. Nonetheless, deploying neural network inference on MCUs poses substantial challenges owing to their restricted memory and computational resources. A typical MCU has a static random-access memory (SRAM) of less than 512 KB and a processing core with a frequency below 400MHz, which is inadequate for general neural networks.

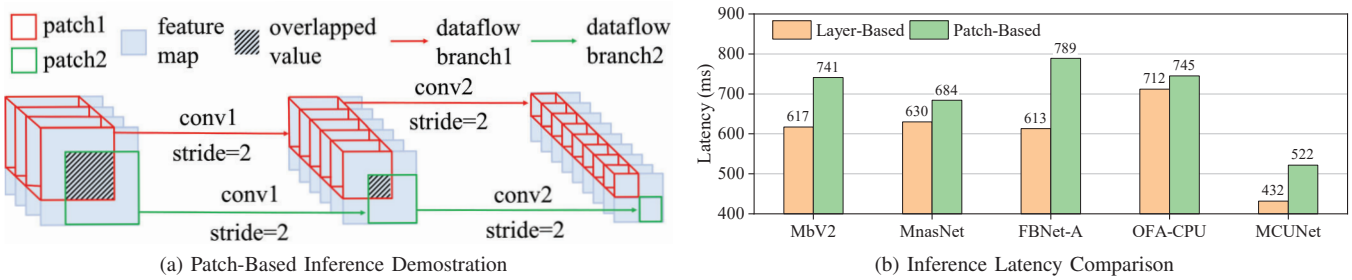
Prior researches have employed various strategies to tackle this problem, including mixed-precision quantization [1]–[5], efficient network architecture design [6]–[8], and dataflow scheduling [8]–[12]. However, the mixed-precision quantization search process and the design of network architecture are time-consuming. Furthermore, mixed-precision quantization leads to accuracy loss. While dataflow scheduling does not result in accuracy loss, it is only suitable for a few neural networks.

Recently, a new dataflow scheduling method known as patch-based inference [8]–[10] has emerged. Figure 1a demonstrates the patch-based inference computation process. Patch-based inference splits the input feature map into multiple patches, causing the neural network's dataflow to be divided into branches, each following a patch. Neural network inference involves sequential execution of dataflow branches. Each branch has a lower peak memory use compared to the overall dataflow. Patch-based inference successfully reduces the neural network's memory footprint. Patch-based inference is a common dataflow scheduling technique that may be used in nearly all neural networks. However, patch-based inference presents a severe issue of redundant computation. As is illustrated in Figure 1a, there exist overlapped values in the dataflow branches. Overlapped values are computed twice, leading to increased computation and inference latency. Layer-based inference is the traditional inference method, where the neural network is executed layer-by-layer. We compared the latency of layer-based and patch-based inference on different models. Figure 1b demonstrates that patch-based inference leads to an 8%-17% increase in inference latency. Previous patch-based inference approaches used heuristic methods [8], [9] to address this issue, resulting in suboptimal results. Mixed-precision quantization can reduce redundant computation due to the fact that values with lower bitwidth require less computation. However, it can lead to accuracy loss and time-consuming search processes.

To address these issues, we propose QuantMCU, a novel patch-based inference approach that uses value-driven mixed-precision quantization to reduce redundant computation. First, QuantMCU performs value-driven patch classification (VDPC) to ensure model accuracy. VDPC divides patches into two categories: outliers and non-outliers. We apply 8-bit quantization for outlier class patches and feature maps on the following dataflow branches. QuantMCU uses a value-driven quantization search (VDQS) strategy on non-outlier class patches and feature maps in the following dataflow branches to reduce search time. VDQS introduces a new search measure that incorporates both accuracy and computation. Model accuracy is represented by activation value entropy, eliminating the need to train the model at each search step. Furthermore, VDQS uses a lightweight iterative technique to determine the bitwidth for each feature map. By applying VDPC and VDQS, we can effectively reduce the redundant computation of the model in patch-based inference

*Guokuan Li (email: liguokuan@hust.edu.cn) and Jianzong Wang (email: jzwang@188.com) are the corresponding authors.

979-8-3503-4859-0/DATE24/© 2024 EDAA



(a) Patch-Based Inference Demostration

(b) Inference Latency Comparison

Fig. 1: **(a):** A simple demonstration of patch-based inference process (only two patches are drawn). **(b):** A comparison experiment result of the inference latency of patch-based and layer-based inference.

and further reduce its peak memory usage, with comparable model accuracy.

The main contributions of our work are summarized below.

- We propose value-driven patch classification (VDPC) for the feature map patches in patch-based inference (**Section III-A**) to maintain model accuracy during our quantization.
- We apply value-driven quantization search (VDQS) on the outlier class patches and the feature maps on the dataflow branches that follow (**Section III-B**). VDQS can effectively shorten the search time of mixed-precision quantization.
- We implement QuantMCU on real-world MCU devices with different resource constraints. We test QuantMCU on various neural networks and two standard datasets. Extensive experiments show that QuantMCU can, on average, reduce the BitOPs and inference latency of state-of-the-art patch-based inference methods by 2.2x and 1.5x, respectively.

II. Related Works

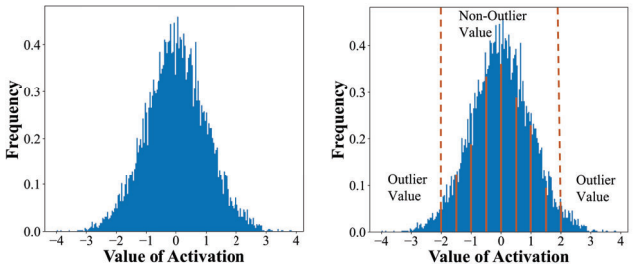
Neural network inference on MCUs. The application of neural network inference on MCUs has shown rapid growth in recent years. Many inference frameworks have been designed, including Tensorflow Lite Micro [13], CMSIS-NN [14], CMix-NN [15], TinyEngine [8], etc. However, none of these frameworks comprehensively considers the optimization for memory usage and computation. Due to these frameworks, neural networks are either unable to deploy because of excessive memory utilization, or if they do, the inference latency will be intolerable.

Dataflow scheduling. The memory footprint of a neural network during inference is closely related to its dataflow. Recent attempts have been made to reduce the peak memory usage of neural networks by scheduling their dataflow. Some researchers [11] try to reorder the operator inferences. They search for an optimal topology of the dataflow graph manually, which represents the smallest peak memory usage. Miao et al. [12] propose to swap data between SRAM and external storage. However, the above dataflow scheduling methods cannot be applied to all the neural networks. Other works [8]–[10] focus on patch-based inference. For example, Cipolletta et al. [9] design a restructuring algorithm to find the optimal patch split layer and dataflow branch length. Saha et al. [10] design a new pooling operator to compute partial feature maps across multiple layers. Nonetheless, there exists a severe redundant computation issue in patch-based inference methods, and their heuristic solutions cannot address it well. In this work, we integrate patch-based inference with value-driven mixed-precision quantization, resulting in a significant reduction in the model's redundant computation and further decreasing its peak memory usage.

Mixed-precision quantization. Quantization, which has long been a popular method of compressing models, is the process of converting floating point 32-bit (FP32) type data in neural networks into lower-bit type [16]. Mixed-precision quantization methods [1]–[5], [7] aim to assign different bitwidth (not higher than 8-bit) for the data of different layers based on the observation that different layers in a neural network have different sensitivity to accuracy. For example, Wang et al. use reinforcement learning (RL) [2] to search for the quantization configuration, which is effective but requires a lot of time and computation resources. The goal of HAWQ-V3 [3] is to allocate bandwidth for each layer based on specific and easily derived metrics, e.g. Hessian spectrum. However, this method fails to consider the change of sensitivity when the values are being quantized or updated in the quantization-aware training process. While Rusci et al. [4] have proposed efficient per-channel quantization, their quantization does not take model accuracy optimization into account. Some researchers [5], [7] have tried to use entropy as the agent of model accuracy to achieve better gradient approximation and lower computation cost. Nonetheless, they usually rely on complicated search techniques such as neural architecture search (NAS). In summary, traditional mixed-precision quantization methods suffer from accuracy loss and usually require time-consuming search processes. In this work, We propose a value-driven mixed-precision quantization approach that utilizes VDPC to maintain excellent model accuracy and employs VDQS to significantly shorten search time.

III. Proposed Method

In this section, the value-driven patch classification and the value-driven quantization search in QuantMCU will be described in detail. We will first introduce how we classify patches according to the outlier value. Then, the design of



(a) (b)

Fig. 2: **(a):** The distribution of the output activation value of the first layer in ResNet18. **(b):** The separation of outlier value and non-outlier value.

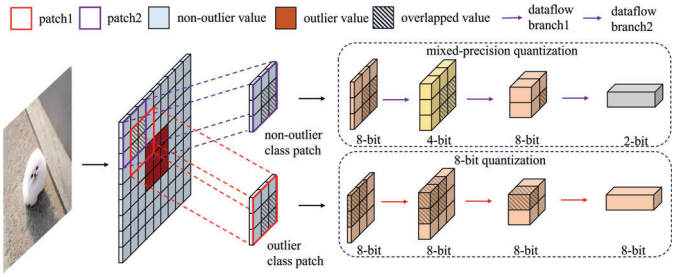


Fig. 3: A demonstration of VDPC. Patch1 is classified as an outlier class patch since it contains an outlier value at its bottom right corner. Patch2 is classified as a non-outlier class patch since it does not contain any outlier value. For patch1 and dataflow branch1, we apply 8-bit quantization, while for patch2 and dataflow branch2, we apply mixed-precision quantization.

a new quantization search metric and a lightweight iterative search algorithm will be described.

A. Value-driven patch classification

In general, the activation value distribution of neural networks exhibits a bell-shaped or Gaussian-like pattern, as is illustrated in Figure 2a. One notable characteristic of such distribution is that while a small portion of values deviates significantly from 0 and often plays a crucial role, the majority of values cluster around 0 and have minimal impact on model accuracy.

Based on the idea presented in [17], we designate the value around 0 as the non-outlier value, and the value far away from 0 as the outlier value, as is illustrated in Figure 2b. Since activation values are composed of outlier values and non-outlier values, each split patch may consist entirely of outlier values, entirely non-outlier values, or maybe both. As a result, the patches are divided into **outlier class** and **non-outlier class** using VDPC. The patches with outlier values are known as the outlier class patches. Applying mixed-precision quantization to the outlier class patches will significantly affect model accuracy because outlier values play a major role in accuracy. Since 8-bit quantization can preserve more model information than mixed-precision quantization, we only apply it to outlier class patches. Feature maps on the dataflow branches that follow should also only use 8-bit quantization. The VDPC method will avoid a significant accuracy decrease in the quantization. On the other hand, the non-outlier class patches are those patches full of non-outlier values. Since non-outlier values are less important in terms of accuracy, we apply mixed-precision quantization to these patches and the feature maps on the dataflow branches following them. Figure 3 shows a VDPC demonstration.

Assuming that the activation value distribution follows a Gaussian distribution, we compute its Probability Density Function (PDF) to find the best outlier/non-outlier separation. For each activation value x, there is:

$$F(x) = \begin{cases} 0, & \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \leq \phi \\ 1, & \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} > \phi \end{cases} \tag{1}$$

where $F(x) = 1$ indicates an outlier value and $F(x) = 0$ indicates a non-outlier value, μ is the mean, σ is the variation, and ϕ is a predefined threshold. ϕ should be properly set. An excessively large ϕ will eliminate some important information conveyed by the outlier value and result in a sharp decline in accuracy, while an extremely small ϕ can not fully reduce the redundant computation. We test different configurations of ϕ in the practical experiment (**Section IV-D**).

B. Value-driven quantization search

Quantization score. It should be noted that the accuracy and computation of the model will change each time quantization is applied to a feature map. To accurately measure the impact, VDQS specifies a new search metric called quantization score.

First, we use Bit Operations (BitOPs) as the representation of computation. If the i_{th} feature map is quantized to b bitwidth, we define the impact on the model computation as follows:

$$\Phi(i, b) = \frac{\Delta B(i, b)}{B} \tag{2}$$

where $\Delta B(i, b)$ is the BitOPs reduction of the i_{th} feature map after b-bit quantization, and B is the sum of the BitOPs of all the feature maps in the full-precision model.

Subsequently, we employ activation value entropy as the representation of accuracy so as to avoid training the model at each step of the quantization. Our insight is that a quantized feature map with higher entropy can preserve more representative capabilities of the original model [5]. Furthermore, the entropy of the last output feature map determines the expressiveness of the system [7]. In order to calculate the entropy of the i_{th} feature map with bitwidth b, we use the empirical distribution to approximate the actual distribution of the activation values. In other words, we divide the activation value range uniformly into k bins, where k is a predefined hyperparameter. We count the number of activation values that fall into the j_{th} bin as x_j. Then we assume that each activation value in the j_{th} bin has the probability:

$$\hat{p}_j = \frac{x_j}{n_i} \tag{3}$$

where n_i is the total number of activation values in the i_{th} feature map. The entropy of the i_{th} feature map with bitwidth b is calculated as follows:

$$H(i,b) = -\sum_j \hat{p}_j log(\hat{p}_j) \tag{4}$$

We define the impact on the model accuracy as follows:

$$\Omega(i,b) = \frac{\Delta H(i,b)}{H(N, b_{last})} \tag{5}$$

where $\Delta H(i,b)$ is the entropy reduction of the i_{th} feature map after b bitwidth quantization, and $H(N, b_{last})$ is the entropy of the last output feature map with bitwidth b_{last}.

Finally, we define quantization score as the weighted sum of $\Phi(i,b)$ and $\Omega(i,b)$, which is calculated as follows:

$$S(i,b) = -\lambda\Omega(i,b) + (1-\lambda)\Phi(i,b) \tag{6}$$

where λ is the weight parameter that balances the importance of computation and accuracy. λ is chosen based on how current applications appreciate the neural network's computation and accuracy. $S(i,b)$ represents the benefits of b bitwidth quantization to the i_{th} feature map.

Lightweight iterative search process. VDQS utilizes a lightweight iterative search algorithm based on quantization score to determine the bitwidth of each feature map. The detailed algorithm is shown in Algorithm 1. For every feature map, there are m kinds of candidate bitwidths(In a real implementation, due to the constraint of the software library, the feature map is only able to be quantized to 8, 4, and 2 bits. Therefore m is set to be 3 in practice.). First, we calculate the quantization score for every feature map with every possible bitwidth. Then we select each feature map with the highest quantization score as the initial bitwidth. It should be noted that every two adjacent feature maps should satisfy the following constraint:

$$\text{Mem}(i,b_i) + \text{Mem}(i+1, b_{i+1}) \leq M \quad i = 0, 1, ..., N-1 \tag{7}$$

where Mem(i,b_i) denotes the memory usage of the i_{th} feature map with bitwidth b_i, and M denotes the memory constraint of the MCU. Finally, we iterate the dataflow branch until Equation 7 is satisfied. During the two iterations, we consider the pair that consists of two adjacent feature maps. We adjust the latter feature map in the first iteration and the former feature map in the second iteration. To make the necessary change, the bitwidth of the feature map with the suboptimal quantization score in comparison to the present one is set.

Algorithm 1 Bitwidth determination

Require: A dataflow branch of N layers, a memory constraint M, available bitwidth kinds m, candidate bitwidths sets for each feature map: $\{s_1^i, s_2^i, ..., s_m^i\}_{i=0}^N$
Ensure: The bitwidth for each feature map: $b_0, b_1, ..., b_N$

```
for i = 0 to N do
    for j = 1 to m do
        calculate S(i, s_j^i) according to Equation 6
    end for
    sort s_1^i, s_2^i, ...s_m^i according to the descending order of quantization score and derive a new set t_1^i, t_2^i, ..., t_m^i
    b_i ← t_1^i
end for
while Equation 7 is not True do
    TRAVERSE(0, N-1, 1)
    TRAVERSE(1, N, -1)
end while
function TRAVERSE(a, b, r)
    for i = a to b do
        Denote the index of b_{i+r} in t_1^{i+r}, t_2^{i+r}, ..., t_m^{i+r} as k
        while NEEDCHANGE(i, r, k) do
            b_{i+r} ← t_{k+1}^{i+r}
        end while
    end for
end function
function NEEDCHANGE(i, r, k)
    if Mem(i, b_i) + Mem(i+1, b_{i+1}) > M then
        if k < m and Mem(i, b_i) ≤ Mem(i+r, b_{i+r}) then
            return True
        end if
    end if
    return False
end function
```

IV. Evaluation Results

In this section, we first present the experiment results of QuantMCU with layer-based and patch-based inference methods on image classification tasks and object detection tasks on two MCU platforms. Then we conduct ablation studies to validate the impact of VDPC and VDQS. Finally, we analyze the influence of hyperparameters and visualize the quantization results of QuantMCU.

A. Experiment setup

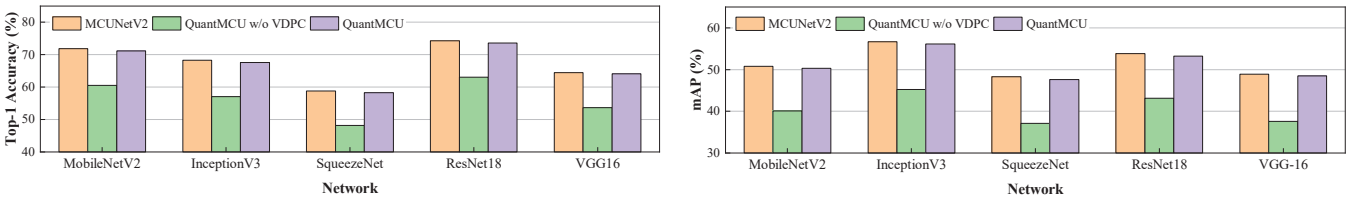
Datasets. We conduct the experiments on two classic neural network applications: image classification and object detection to evaluate QuantMCU in various fields. We use two standard benchmarks in this work: ImageNet [18] for the image classification task and Pascal VOC [19] for the object detection task. All the images are resized to a resolution of 224*224.

Implementation. We have tested QuantMCU on two different real-world MCU devices: Arduino Nano 33 BLE Sense (ARM Cortex-M4, 256kB SRAM/1MB Flash) and STM32H743 (ARM Cortex-M7, 512KB SRAM/2MB Flash). We use Tensorflow Lite [13] to execute 8-bit quantization and the CMix-NN library [15] for sub-byte quantization on MCUs.

Metrics. We use BitOPs to evaluate the model computation. For the image classification task, we employ the Top-1 correct rate as the accuracy measure. For the object detection task, mAP (mean Average Precision) is used to evaluate accuracy. We also measure the inference latency of neural networks using QuantMCU and patch-based inference approaches on MCU devices.

TABLE I: The comparison of QuantMCU with state-of-the-art patch-based inference methods and layer-based inference methods on MobileNetV2 network on different datasets and different platforms. The width multiplier and resolution of the model are adjusted to fit MCU memory.

Platform	Dataset	Metric	Layer-Based	MCUNetV2 [8]	Cipolletta et al. [9]	RNNPool [10]	QuantMCU
Arduino Nano 33 BLE Sense (256KB SRAM, 1MB Flash)	ImageNet [18]	Peak Memory (KB)	244	196	122	226	**78**
		BitOPs (M)	1536	1690	1721	1582	**719**
		inference Lat. (ms)	617	741	784	640	**486**
	PascalVOC [19]	Peak Memory (KB)	252	207	146	242	**99**
		BitOPs (M)	2176	2459	2524	2389	**1171**
		inference Lat. (ms)	656	793	848	717	**502**
STM32H743 (512KB SRAM, 2MB Flash)	ImageNet [18]	Peak Memory (KB)	505	465	380	477	**298**
		BitOPs (M)	4057	4283	4405	4124	**1987**
		inference Lat. (ms)	1684	1799	1945	1736	**1208**
	PascalVOC [19]	Peak Memory (KB)	509	438	382	477	**303**
		BitOPs (M)	5842	6162	6347	5938	**2933**
		inference Lat. (ms)	1792	1912	2089	1836	**1323**



(a) Top-1 Accuracy on ImageNet

(b) mAP on Pascal VOC

Fig. 4: The accuracy comparison of QuantMCU with patch-based inference on different networks on two different datasets.

B. Comparison with state-of-the-art patch-based inference methods

We evaluate the performance of QuantMCU against three state-of-the-art patch-based inference methods and a layer-based inference method. The results are shown in Table I. The table shows that QuantMCU may further reduce the peak memory usage of patch-based inference methods by 1.26x-2.90x. Furthermore, all the patch-based inference methods have higher BitOPs and inference latency than the layer-based inference method. However, QuantMCU can reduce the BitOPs and inference latency of patch-based inference methods, even smaller than those of the layer-based inference method. QuantMCU decreases the BitOPs and the inference latency by 2.2x and 1.5x respectively on average compared to the patch-based inference methods. This is mainly due to our combination of quantization techniques.

C. Ablation study

Impact of VDPC. We compare the accuracy of MCUNetV2 [8] (a classic patch-based inference method), "QuantMCU without VDPC", and QuantMCU on two datasets. In the "QuantMCU without VDPC" group we apply VDQS to all the patches. The results are shown in Figure 4. "QuantMCU without VDPC" experiences 10%-15% accuracy loss compared with MCUNetV2. In contrast, QuantMCU achieves less than 1% accuracy loss on both datasets, demonstrating how VDPC preserves the model accuracy.

Impact of VDQS. We compare VDQS with some state-of-the-art quantization methods, including uniform-precision and mixed-precision. The results are shown in Table II. We can observe that VDQS reduces the peak memory usage of other quantization methods by 1.32x-2.62x. Besides, VDQS increases the accuracy by 0.7%-7.8% compared with other quantization methods. In addition, VDQS can finish the quantization process in 0.5 minutes, which is drastically faster than other methods. This is because VDQS uses entropy as the representation of accuracy to avoid extra training and applies an iterative search process instead of relying on time-consuming RL or NAS.

TABLE II: Comparison of different quantization methods on MobileNetV2 network on ImageNet dataset. "W/A-Bits" means the bitwidth of weight/activation value. "MP" means mixed-precision. "Time" means the time cost of quantization process.

Method	W/A-Bits	Top-1	BitOPs	Memory	Time
Baseline	8/8	71.9%	19.2G	1372kB	-
Pact [20]	4/4	61.4%	7.42G	692kB	45min
Rusci et al. [4]	MP/MP	61.8%	7.42G	690kB	33min
HAQ [2]	MP/MP	68.5%	42.8G	950kB	90min
HAWQ-V3 [3]	MP/MP	63.4%	13.6G	787kB	30min
QuantMCU	8/MP	**69.2%**	10.9G	**523kB**	**0.5min**

D. Analysis

We measure the Top-1 and Top-5 accuracy on ImageNet with QuantMCU under different ϕ. As is illustrated in Figure 5, the Top-1 and Top-5 accuracy stays stable when ϕ is smaller than 0.96. However, when ϕ exceeds 0.96, the accuracy decreases rapidly. Therefore, we choose 0.96 as the optimal value of ϕ. In addition, we test the impact of hyperparameter λ on QuantMCU. As is shown in Table III, when λ increases, the BitOPs rise along with the Top-1 Accuracy. We choose 0.6 as the value of λ to achieve the best comprehensive benefit.

We visualize the bitwidth assignment results after quantization in QuantMCU for MobileNetV2 and MCUNet in Figure 6. From the figure, we can see that more than half of the feature maps are assigned sub-byte precision. The bitwidths of the layers at the end of a branch are mainly 8-bit, while the feature maps at the start of a branch are usually assigned low

TABLE III: The impact of different values of λ on QuantMCU.

λ	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Top-1 Acc. (%)	65.6	67.1	67.9	68.7	69.2	70.1	71.2
BitOPs (G)	7.6	8.4	9.2	10.1	10.9	14.3	18.7

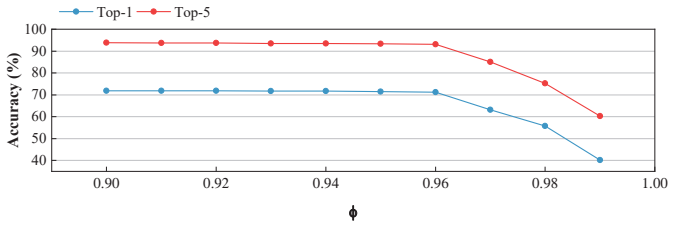


Fig. 5: Top-1 and Top-5 accuracy of QuantMCU under different ϕ values on MobileNetV2 network on the ImageNet dataset.

bitwidths. This is because the first few feature maps usually have a large size and need to be quantized to lower bitwidths in order to reduce computation, whereas the last few feature maps typically contribute significantly to model accuracy and require quantization to higher bitwidths to maintain accuracy.

V. Conclusion

We propose a novel value-driven mixed-precision quantization called QuantMCU for patch-based inference on MCUs. We utilize VDPC to maintain accuracy in the quantization, which classifies patches according to whether they contain outlier values. Besides, we employ VDQS to decrease the quantization search time. VDQS defines a new search measure based on activation value entropy and BitOPs to avoid additional training, and it uses a lightweight iterative search method to speed up the search process. Experimental results on real-world MCU devices prove that QuantMCU can reduce the computation of previous patch-based inference methods by 2.2x on average.

VI. Acknowledgements

This work was sponsored by the Key Research and Development Program of Guangdong Province under Grant No. 2021B0101400003, the National Natural Science Foundation of China under Grant No.62072196, and the Creative Research Group Project of NSFC No.61821003. The corresponding authors are Guokuan Li from Huazhong University of Science and Technology (liguokuan@hust.edu.cn) and Jianzong Wang from Ping An Technology (Shenzhen) Co., Ltd. (jzwang@188.com).

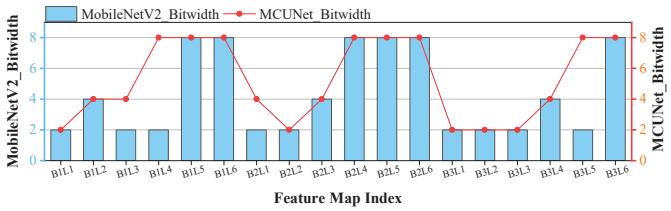


Fig. 6: The visualization of bitwidth assignment after quantization for MobileNetV2 and MCUNet network. The "BxLy" means the y_{th} feature map on the x_{th} branch.

References

[1] S. Kundu, S. Wang, Q. Sun, P. A. Beerel, and M. Pedram, "Bmpq: bit-gradient sensitivity-driven mixed-precision quantization of dnns from scratch," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 588–591.

[2] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8612–8620.

[3] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. Mahoney *et al.*, "Hawq-v3: Dyadic neural network quantization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 875–11 886.

[4] M. Rusci, A. Capotondi, and L. Benini, "Memory-driven mixed low precision quantization for enabling deep network inference on microcontrollers," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 326–335, 2020.

[5] X. Zhu, W. Zhou, and H. Li, "Adaptive layerwise quantization for deep neural network compression," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[7] Z. Sun, C. Ge, J. Wang, M. Lin, H. Chen, H. Li, and X. Sun, "Entropy-driven mixed-precision quantization for deep network design," *Advances in Neural Information Processing Systems*, vol. 35, pp. 21 508–21 520, 2022.

[8] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. Han, "Mcunetv2: Memory-efficient patch-based inference for tiny deep learning," *arXiv preprint arXiv:2110.15352*, 2021.

[9] A. Cipolletta and A. Calimera, "Dataflow restructuring for active memory reduction in deep neural networks," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 114–119.

[10] O. Saha, A. Kusupati, H. V. Simhadri, M. Varma, and P. Jain, "Rnnpool: Efficient non-linear pooling for ram constrained inference," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 473–20 484, 2020.

[11] E. Liberis and N. D. Lane, "Neural networks on microcontrollers: saving memory at inference via operator reordering," *arXiv preprint arXiv:1910.05110*, 2019.

[12] H. Miao and F. X. Lin, "Towards out-of-core neural networks on microcontrollers," in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*. IEEE, 2022, pp. 1–13.

[13] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[14] L. Lai, N. Suda, and V. Chandra, "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus," *arXiv preprint arXiv:1801.06601*, 2018.

[15] A. Capotondi, M. Rusci, M. Fariselli, and L. Benini, "Cmix-nn: Mixed low-precision cnn library for memory-constrained edge devices," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 871–875, 2020.

[16] X. Qu, J. Wang, and J. Xiao, "Quantization and knowledge distillation for efficient federated learning on edge devices," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2020, pp. 967–972.

[17] C. Guo, C. Zhang, J. Leng, Z. Liu, F. Yang, Y. Liu, M. Guo, and Y. Zhu, "Ant: Exploiting adaptive numerical data type for low-bit deep neural network quantization," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022, pp. 1414–1433.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[19] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, pp. 303–338, 2010.

[20] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.