LETTER

Sample-wise dynamic precision quantization for neural network acceleration

Bowen Li[1], Dongliang Xiong[1, a)], Kai Huang[1], Xiaowen Jiang[1], Hao Yao[2], Junjian Chen[2], and Luc Claesen[3]

Abstract Quantization is a well-known method for deep neural networks (DNNs) compression and acceleration. In this work, we propose the Sample-Wise Dynamic Precision (SWDP) quantization scheme, which can switch the bit-width of weights and activations in the model according to the task difficulty of input samples at runtime. Using low-precision networks for easy input images brings advantages in terms of computational and energy efficiency. We also propose an adaptive hardware design for the efficient implementation of our SWDP networks. The experimental results on various networks and datasets demonstrate that our SWDP achieves an average of 3.3× speedup and 3.0× energy saving over the bit-level dynamically composable architecture BitFusion.

Keywords: convolutional neural networks, dynamic quantization, hardware accelerators
Classification: Integrated circuits (logic)

1. Introduction

Deep neural networks have achieved state-of-the-art performance in various computer vision tasks such as image classification [1] and object detection [2]. With the development of DNNs, their application scenarios become more extensive, from GPUs to domain-specific accelerators [3, 4]. Static networks with fixed computational graphs and parameters are difficult to adapt to diverse inference resource budgets. In contrast, networks with dynamic depth [5], width [6] and kernel size [7] tackle this problem and become an emerging research topic. Many accelerators with execution prediction are also proposed to reduce the computation of DNNs [8, 9].

Quantization is a common step in DNNs deployment [10, 11, 12]. However, dynamic bit-width quantization has not been well studied. On the one hand, previous algorithm research focused on training a set of model weight to support quantization with a variety of bit-widths [13, 14, 15]. The network can directly switch precision without retraining when the deployment scenario changes. On the other hand, previous hardware accelerators provide flexible architectural support for variable bit-width across DNN layers and models [16, 17, 18, 19]. But none of them take the finer-grained differences between the input samples into consideration. Since the classification difficulty of different inputs is different for the neural network, using a low-precision model to infer easy images can achieve the effect of acceleration and energy-saving without a significant network accuracy drop. However, it is hard to predict the classification difficulty and allocate appropriate precision for each input picture. It is a common issue for all networks with input-dependent dynamic architecture. Dynamic pruning tackled this by introducing an additional gate module for dynamic filter selection or width decision [20, 21]. Multi-exit networks commonly consider the classifier confidence to decide whether to early-exit the architecture [5].

To address this, we propose an algorithm-architecture co-design named sample-wise dynamic precision (SWDP). Specifically, we adopt the quantization scheme proposed in our previous work (Dynamic Precision Onion Quantization) DPOQ [22]. The key idea of DPOQ is to train a network that supports multiple-precision quantization, and the high-precision network reuses the intermediate results from the low-precision network during forward propagation. Therefore, we regard the low-precision output as an early exit and try to find the optimal exit for each sample in this work. The automatic precision adaption is illustrated in Fig. 1. Input images are routed to whether to use high precision regarding the confidence of low-precision output. The cost of obtaining the n-bit high-precision network results after the $n/2$-bit low-precision network has been run is running another $n/2$-bit network instead of the whole n-bit network.

We also design the hardware implementation for SWDP algorithm based on the variable precision architecture BitFusion [16]. Extra element-wise addition for intermediate results and the input sample difficulty comparator are introduced to support our sample-wise precision decision algorithm. The main contributions of this work are summarized as:

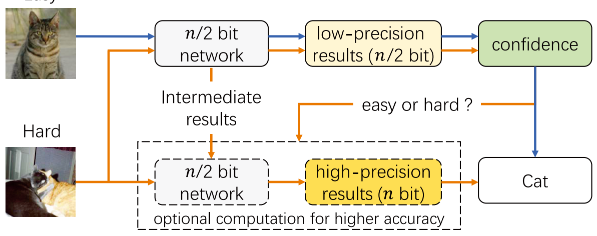
Easy
Hard
$n/2$ bit network
low-precision results ($n/2$ bit)
confidence
Intermediate results
easy or hard ?
$n/2$ bit network
high-precision results (n bit)
Cat
optional computation for higher accuracy

Fig. 1 Overview of sample-wise dynamic precision for inference.

[1] School of Micro-Nano Electronics, Zhejiang University, Hangzhou 310030, China
[2] Digital Grid Research Institute, China Southern Power Grid, Guangzhou 510670, China
[3] Engineering Technology-Electronics-ICT Department, University of Hasselt, 3590 Diepenbeek, Belgium
[a)] xiongdl@zju.edu.cn

DOI: 10.1587/elex.19.20220229
Received May 13, 2022
Accepted July 4, 2022
Publicized July 13, 2022
Copyedited August 25, 2022

1. We propose the dynamic quantization scheme named SWDP, whose precision is adaptive to the input difficulty.
2. We design the hardware architecture implementation for efficient SWDP inference acceleration.
3. We perform extensive experiments and demonstrate that our SWDP outperforms its static counterparts.

2. Sample-wise dynamic precision quantization

In this section, we introduce the proposed algorithm method for sample-wise dynamic quantization. The approach involves two steps: training a network supporting multi-precision and determining the inference precision.

2.1 Multi-precision quantization

Multi-precision quantization can satisfy various hardware resource budgets, and has a wide range of application scenarios. For example, the DNNs deployment platform switches from cloud hardware to edge hardware, or the system may enter low-power mode or high-speed mode if the battery level changes even in the same mobile device. We adopt the multi-precision quantization scheme from our previous work DPOQ. The main calculation of neural networks is concentrated in convolution or fully-connected layer, which can be divided into multiple multiplication and accumulation calculations. The quantized n-bit integer weight W is split into the corresponding $n/2$-bit high part W_h and low part W_l, and reformulates the output O of a convolution layer with input I as:

$$O = I * ((W_h << \frac{n}{2}) + W_l) = O_h + O_l \tag{1}$$

The whole computation is split into two parts, the calculation of O_l can be saved as long as the quantization error is acceptable. However, due to the nonlinear activation function, outputs after the first convolution have different inputs and can not be directly reformulated. Therefore, W_h and W_l are treated as two independent $n/2$-bit weights rather than the above high and low part of the n-bit integer weight. Thanks to the learnable characteristic of neural network parameters, the weight W_l can be trained to approximate the output difference caused by weight W_h quantization error with $\min ||O - O_h - I_l * W_l||_2$. When training a network, it is to minimize the final loss.

Most networks are built from basic blocks, and each basic block contains several convolutional, batch normalization and ReLU layers. Different form DPOQ, the frequency of such error compensation is reduced from layer-wise to block-wise to further save data transmission, as shown in Fig. 2. The upper half network and the lower half network are termed as the backbone network and the compensation network, separately. The low-precision network is wrapped in the high-precision network as a part and thus the whole network looks like an onion. The benefit of this scheme is that it allows simple precision switch by only increasing or decreasing part of the network.

Since the data distributions of two precision contradict each other, precision shift batch normalization (PSBN) is adopted. The PSBN provides an individual set of learnable scale γ and shift β for each precision. Those are channel-wise parameters, and the cost is neglected compared with the convolutional layer. The interaction of two precision networks is an element-wise addition, whose cost is almost the same as the shortcut connections in ResNet [1]. Note that the dual-precision network can be easily extended to a multi-precision network by adding a network with difference bit-width and the corresponding connections. In this paper, we train the network with 4 kinds of precision (bit list = {2, 4, 8, 16}) by default, and the weight bit-widths of 4 subnetworks are 2, 2, 4 and 8. We train networks of all precision jointly:

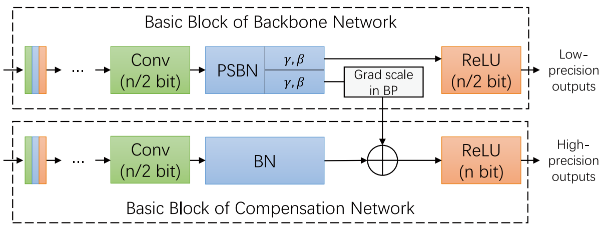

Fig. 2 Detailed basic block with multi-precision quantization.

$$Loss = \sum_{k \in bitlist} \alpha_k Loss_k \tag{2}$$

Where each loss is assigned with a parameter α to weigh its contribution. The default value of α is 1. Moreover, a scale can be multiplied to the gradient from high precision to low precision during backward propagation. This gradient scale parameter is used to adjust the training preference, which balances the influence of high precision output loss and low precision output loss on low precision weights. The value of gradient scale depends on the bit-widths of high and low precision networks.

2.2 Precision decision

After the network is trained, it can be quantized to flexible bit-width. Same as shallow-deep network [23], assigning simple samples with low precision not only provides significant improvements in inference time, but also prevents overfitting. Next step is to find the optimal precision for every input. With the help of our unique data reuse mechanism, the low precision output is an early-exit. The low precision inference and the high-precision inference can be carried out successively without repeated calculations, and their predictions can be combined to obtain the ensemble result.

We use the output confidence as the condition of criterion for exit decision in classification problems. Confidence is the maximum output after softmax, and it is $\in$ [0,1]. By comparing the confidence with a user-selected threshold, we can determine the degree of precision between current precision and higher precision for inference. The selection of threshold is a performance-complexity trade-off depending on the actual application requirements. Our work is plotting the accuracy-operation diagram.

The trained network supports 4 kinds of precision, which means we have 3 thresholds to be set, namely $T_{2\rightarrow4}$, $T_{4\rightarrow8}$ and $T_{8\rightarrow16}$. Our goal is to achieve maximum accuracy under any computation budget. In this section, we discuss

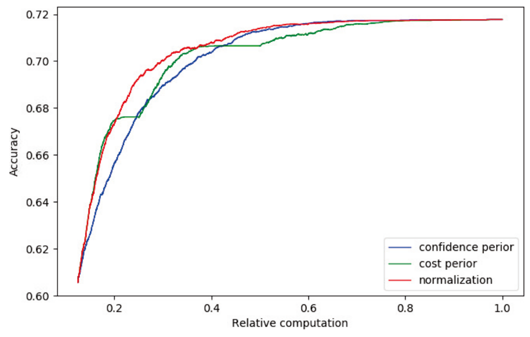

Fig. 3 Computation VS. accuracy (ResNet20 in CIFAR100).

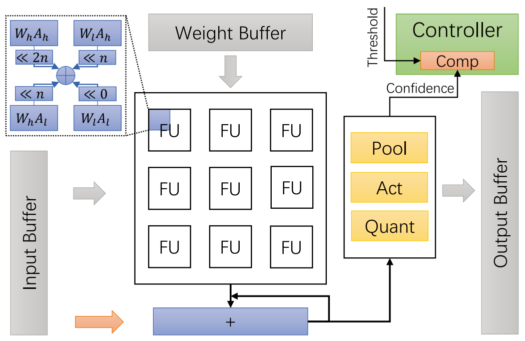

Fig. 4 Hardware architecture overview.

3 relationships of the threshold values. The first one is confidence-prior, 3 thresholds are set with the same value. The problem of the confidence-prior way is that it never considers the cost. For example, the computational cost of raising the precision of an input from 2 bits to 4 bits is much less than that from 8 bits to 16 bits, but they have the same priority when their confidences are the same. So the second one is cost-prior, 3 thresholds are simplified to one and the other two thresholds are either 1 or 0. The problem of the cost-prior way is that the growth of accuracy becomes slower with complexity increases. The third one is normalization, the relationship of 3 thresholds is described by the following formula:

$$T_{2\to 4}^{1/4} = T_{4\to 8}^{1/2} = T_{8\to 16} \tag{3}$$

Where threshold is normalized to $T_{8\to 16}$ with power function, because it will not change the value range of [0,1]. The exponent for $T_{2\to 4}$ is calculated as $(4-2)/(16-8) = 1/4$ considering the cost of different bit-widths.

The accuracy-operation diagram of the three relationships is shown in Fig. 3. The number of computations is the relative value to the 16-bit quantized network. The accuracy is obtained from the ensemble results. From the figure, it can be seen that the problems of the confidence-prior way and the cost-prior way mentioned in the above analysis are reflected in the figure, and the normalization way achieves the highest cost performance almost everywhere.

3. Hardware architecture design

In order to give full play to the superiority of our variable bit-width quantization scheme, our hardware implementation needs to support various numerical precision. Many state-of-the-art accelerators support variable precision [16, 17, 18, 19].

Stripes [17] uses bit-serial compute units, whose execution time scales linearly with the length of the numerical representation used. But it only supports the neuron (weight) precision to be variable. BitFusion [16] is another bit-flexible accelerator. The fusion unit in it can offer multiply operation between operands with 2-bit, 4-bit, 8-bit and 16-bit, which adapts to our algorithm well. We believe that {2,4,8,16} quantization is a reasonable bit selection. Firstly, it covers almost all scenarios. Secondly, too fine-grained bit width adjustment has little significance. Especially in the range of 8-16 bits, increasing or reducing 1 bit has little impact on network accuracy and inference cost. Therefore, our hardware implementation adopts the design of the fusion unit (FU) in BitFusion.

3.1 Overall architecture

We illustrate the overview of our hardware implementation in Fig. 4. The major components in the design are listed as follows: on-chip buffers, a systolic array, an accumulator, a controller and post-process unit.

The on-chip buffer is composed of weight buffer, input buffer and output buffer. Network parameters and input images are stored in the off-chip DRAM. Double buffering is applied to overlap computation and off-chip data access. To provide data with different bit widths, a series of registers and multiplexers are placed at the buffer data port. We also add an extra data path from the input buffer to the accumulator to ensure that the intermediate data of the backbone network can be sent to the compensation network during high precision inference.

The systolic array of FU is the main operation execution module, and it has advantages of high data reuse rate and simple control. Input data is shared across columns, different columns of FUs compute different output channels. Partial sums of output data are accumulated across rows, different columns of FUs compute different input channels. The construction of FU is detailed in the figure, the decomposed multiplication results of 4 combinations of most significant bit (MSB) and the least significant bit (LSB) of weights and activations are left-shifted and added together.

The accumulator accumulates partial sums until all input channels and kernels are traversed. It can support the element-wise addition in increasing precision. The post-process unit is used to accomplish the operations after multiply-accumulation including pooling, activation and quantization. And it can also perform Softmax function and output confidence to the controller.

The controller records states and generates control signals for the whole system. We add an additional comparator in the controller to judge the difficulty of input samples by comparing the calculated confidence with the user-defined threshold. If the confidence is larger than the threshold or the highest precision inference is performed, the done interrupt will be set. Otherwise, higher precision is required to be applied in the network for more accurate predictions.

3.2 Dynamic precision cost

The precision of the backbone network and three com-

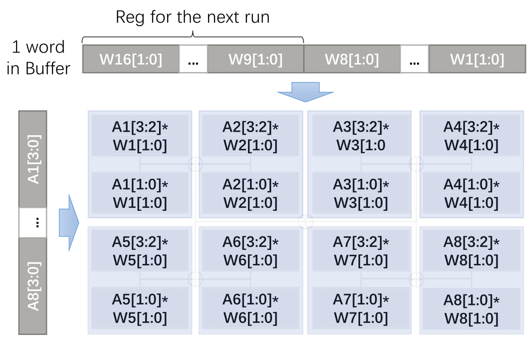

Fig. 5 Data mapping for W2A4 compensation network.

pensation networks for $\{2,4,8,16\}$ quantization are W2A2, W2A4, W4A8 and W8A16, separately. The number of bit operations for our 4-bit dynamic precision quantization F_{dp4} is the sum of the bit operations of W2A2 network F_{w2a2} and W2A4 network F_{w2a4} as follows:

$$F_{dp4} = F_{w2a2} + F_{w2a4} = (2*2+2*4)F_{net} = 12F_{net} \quad (4)$$

where F_{net} is the number of operations of the network, it is calculated as:

$$F_{net} = \sum_{l=1}^{L} F_{conv}(l) = \sum_{l=1}^{L} M_l * N_l * X_l * Y_l * K_l^2 \quad (5)$$

where $F_{conv}(l)$ is the number of multiplications in the lth convolutional layer, M and N are the output and input channels, X and Y are output feature size, K is the filter kernel size. Reducing the bit-width of the network almost reduces the bit-level computations quadratically. For the above we can see that our F_{dp4} cost less than the static W4A4 quantization $F_{w4a4} = 16F_{net}$. Similarly, $F_{dp8} = 44F_{net} < F_{w8a8} = 64F_{net}$, $F_{dp16} = 172F_{net} < F_{w8a8} = 256F_{net}$. The drawback of our dynamic precision quantization is that it introduces extra memory data transmission because of the sequential execution mechanism of each precision.

3.3 Data mapping

The dynamic precision network has several subnetworks with different bit-widths. The weight parameters of the subnetworks are stored separately. Fig. 5 takes the W2A4 compensation network as an example, and shows the data storage format in the buffer and the computation mapping in the FU.

Eight 4-bit input activations or sixteen 2-bit weights are organized into one word in the buffer. In this logically fusing mode, the FU can only process 8 weights at a time. Therefore, $W9$ to $W16$ are registered for the next run to avoid repetitive access to the weight buffer. The FU is configured to construct 8 Fused-PEs and each PE supports W2A4 operands for multiply operations. The 8 multiply results from Fused-PEs are accumulated together to generate one single outgoing partial sum.

4. Experimental results

4.1 Dynamic precision results

We evaluate our SWDP on benchmarks of diverse networks (LeNet-5 [24], VGG16 [25], ResNet20 and ResNet18 [1]) and datasets (MNIST [26], CIFAR10, CIFAR100 [27] and ImageNet [28]). We adopt the LSQ [29] to quantize weights and activations. The network training and evaluating are implemented under the Pytorch framework [30] with 8 NVIDIA RTX 2080Ti GPUs. For training models, we use the SGD optimizer with a momentum of 0.9 and a weight decay of 5e-4. The initial learning rate is 0.1 and decays with cosine scheduler. The batch size is 256. All the networks are trained from scratch, and for MNIST, CIFAR and ImageNet datasets, networks are trained for 10, 160 and 90 epochs, respectively.

The quantization results of individual training baseline and our SWDP are listed in Table I. From the results we can see that our SWDP achieves higher accuracy than the baseline in most cases. With the increase of bit width, the advantages of our method are more obvious. For example, Our SWDP significantly boosts accuracy of 3.3% on 16-bit ResNet20 of CIFAR100 dataset over the full-precision baseline. The results prove that multi-precision quantization is feasible, and it is able to further improve trade-off between accuracy and efficiency with dynamic precision quantization.

Based on the above trained network, we try to find the optimal inference precision for the input samples by comparing confidence against the thresholds. Fig. 6 shows the proportion of samples in the dataset of each precision, “ResNet” and “CIFAR” are abbreviated to “RN” and “C” in the figure. And we guarantee that the accuracy of SWDP is no lower than that of static 8-bit quantization under this confidence threshold setting. It can be seen from the figure that the bit widths vary across networks and datasets to guarantee no degradation of prediction accuracy. Most of the samples in the dataset are quite easy to classify. More than half of the images can be applied with 2-bit quantization, and only

Table I Multi-precision quantization results.

Network (Dataset)	Method	Top-1 Accuracy @ Precision			
		2	4	8	16/fp
LeNet-5 (MNIST)	baseline	98.45	98.85	98.99	98.90
	SWDP	98.55	98.98	99.02	99.19
ResNet20 (CIFAR10)	baseline	88.16	91.56	92.26	92.79
	SWDP	88.23	91.16	92.47	92.86
ResNet20 (CIFAR100)	baseline	60.95	67.80	68.13	68.45
	SWDP	60.56	67.62	70.64	71.77
VGG16 (CIFAR10)	baseline	91.10	92.33	92.76	92.89
	SWDP	91.27	93.00	93.32	93.44
VGG16 (CIFAR100)	baseline	67.86	70.55	70.98	71.35
	SWDP	67.95	71.82	72.62	73.04
ResNet18 (ImageNet)	baseline	65.22	70.25	70.72	69.76
	SWDP	64.92	68.50	72.71	73.68

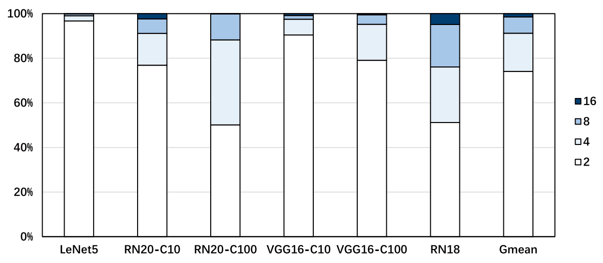

Fig. 6 Sample proportion of each bit widths.

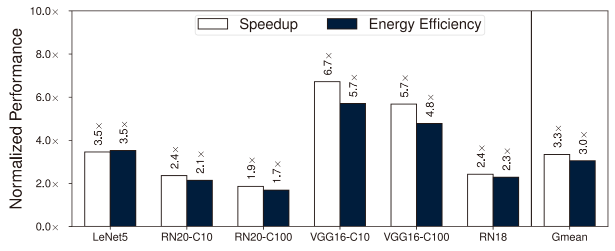

Fig. 7 Normalized performance of SWDP over BitFusion.

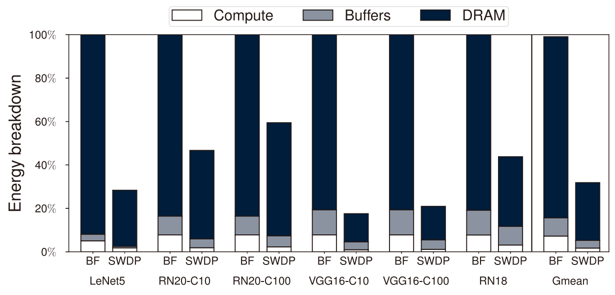

Fig. 8 Energy breakdown.

less than 5% images require 16-bit quantization. This offers great potential for acceleration.

4.2 Hardware results

To evaluate the performance of our hardware architecture, we develop a simulator as the behavior model and implement the accurate RTL model with Verilog HDL. The design is synthesised by Synopsys Design Compiler under 45 nm technology with the commercial standard-cell library. The on-chip buffers are modeled by the CACTI [31]. The system clock is run at the frequency of 500 MHz.

To evaluate the performance and energy of our implementation, we make a comparison with BitFusion using their open-source simulator[1]. For the same area budget, we use the same systolic array size of 16 × 32 and buffer size of 112 KB for both designs. The BitFusion operates on the 8-bit quantization network for all input samples. Our SWDP supports assigning flexible bit width to each sample, and the proportion of each bit width is summarized in Fig. 6. The normalized performance is presented in Fig. 7, and it shows that our SWDP achieves 3.3× speedup and 3.0× energy efficiency on average. The highest improvements of 6.7× speedup are achieved on VGG16-C10 because of its low average bit width and wide network channel. The worst effect is on RN20-C100, but still saves 1.9× relative processing latency and 1.7× energy over BitFusion.

To make a further insight into the energy consumption reduction, we show the energy breakdown for each component in Fig. 8. Most of the energy (over 80%) is consumed in data movement from off-chip DRAM to on-chip SRAM. Quantization reduces the number of bits used for data representation, so it significantly saves the memory access energy of DRAM and Buffers. Moreover, the multiply-accumulate computation is simplified with the decrease of operands precision and its energy consumption is reduced almost quadratically. However, computation energy accounts for a small share in the whole system, the overall energy saving mainly depends on a reduction in data transmission.

In conclusion, our adaptive sample-wise dynamic precision leads to improvements in performance and energy since it compresses network models and simplifies calculation processes.

[1] https://github.com/hsharma35/bitfusion

5. Conclusion

In this paper, we propose the sample-wise dynamic precision quantization and design the corresponding hardware. Our SWDP not only saves redundant computation for easy samples with low-precision inference, but also preserves representation power for hard samples with high-precision inference. The experimental results show that our SWDP achieves remarkable advantages in computational and energy efficiency compared to the static networks.

Acknowledgments

This work is supported by the National Key R&D Program of China (2020YFB0906000, 2020YFB0906001).

References

[1] K. He, *et al.*: "Deep residual learning for image recognition," IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2016) 770 (DOI: 10.1109/CVPR.2016.90).
[2] R.B. Girshick, *et al.*: "Rich feature hierarchies for accurate object detection and semantic segmentation," CVPR (2014) 580 (DOI: 10.1109/CVPR.2014.81).
[3] Y. Chen, *et al.*: "Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE J. Solid State Circuits **52** (2017) 127 (DOI: 10.1109/JSSC.2016.2616357).
[4] T. Luo, *et al.*: "DaDianNao: a neural network supercomputer," IEEE Trans. Comput. **66** (2017) 73 (DOI: 10.1109/TC.2016.2574353).
[5] S. Scardapane, *et al.*: "Why should we add early exits to neural networks?," Cogn. Comput. **12** (2020) 954 (DOI: 10.1007/s12559-020-09734-4).
[6] J. Yu, *et al.*: "Slimmable neural networks," ICLR (2019).
[7] H. Cai, *et al.*: "Once-for-all: train one network and specialize it for efficient deployment," ICLR (2020).
[8] V. Akhlaghi, *et al.*: "SnaPEA: predictive early activation for reducing computation in deep convolutional neural networks," ISCA (2018) 662 (DOI: 10.1109/ISCA.2018.00061).
[9] M. Song, *et al.*: "Prediction based execution on deep neural networks," ISCA (2018) 752 (DOI: 10.1109/ISCA.2018.00068).
[10] N.P. Jouppi, *et al.*: "In-datacenter performance analysis of a tensor processing unit," ISCA (2017) 1 (DOI: 10.1145/3079856.3080246).
[11] J. Wu, *et al.*: "Quantized convolutional neural networks for mobile devices," CVPR (2016) 4820 (DOI: 10.1109/CVPR.2016.521).
[12] F. Zhang, *et al.*: "HFOD: a hardware-friendly quantization method for object detection on embedded FPGAs," IEICE Electron. Express **19** (2022) 20220067 (DOI: 10.1587/elex.19.20220067).
[13] Q. Jin, *et al.*: "AdaBits: neural network quantization with adaptive bit-widths," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 2143 (DOI: 10.1109/CVPR42600.2020.00222).
[14] A. Bulat and G. Tzimiropoulos: "Bit-mixer: mixed-precision networks with runtime bit-width selection," ICCV (2021) 5168 (DOI: 10.1109/ICCV48922.2021.00514).
[15] H. Yu, *et al.*: "Any-Precision Deep Neural Networks," AAAI (2021) 10763.
[16] H. Sharma, *et al.*: "Bit fusion: bit-level dynamically composable architecture for accelerating deep neural network," 45th ACM/IEEE Annual International Symposium on Computer Architecture (ISCA) (2018) 764 (DOI: 10.1109/ISCA.2018.00069).

[17] P. Judd, *et al.*: "Stripes: bit-serial deep neural network computing," 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO) (2016) 19:1 (DOI: 10.1109/MICRO.2016.7783722).
[18] J. Lee, *et al.*: "UNPU: a 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," ISSCC (2018) 218 (DOI: 10.1109/ISSCC.2018.8310262).
[19] S. Zhang, *et al.*: "Thread: towards fine-grained precision reconfiguration in variable-precision neural network accelerator," IEICE Electron. Express **16** (2019) 20190145 (DOI: 10.1587/elex.16.20190145).
[20] Z. Chen, *et al.*: "You look twice: GaterNet for dynamic filter selection in CNNs," IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 9172 (DOI: 10.1109/CVPR.2019.00939).
[21] C. Li, *et al.*: "Dynamic slimmable network," IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021) 8607 (DOI: 10.1109/cvpr46437.2021.00850).
[22] B. Li, *et al.*: "DPOQ: dynamic precision onion quantization," Asian Conference on Machine Learning (ACML) (2021) 502.
[23] Y. Kaya, *et al.*: "Shallow-deep networks: understanding and mitigating network overthinking," Proceedings of the 36th International Conference on Machine Learning (ICML) (2019) 3301.
[24] Y. LeCun, *et al.*: "Gradient-based learning applied to document recognition," Proc. IEEE **86** (1998) 2278 (DOI: 10.1109/5.726791).
[25] K. Simonyan and A. Zisserman: "Very deep convolutional networks for large-scale image recognition," ICLR (2015).
[26] Y. LeCun, *et al.*: "Gradient-based learning applied to document recognition," Proc. IEEE **86** (1998) 2278 (DOI: 10.1109/5.726791).
[27] A. Krizhevsky: "Learning multiple layers of features from tiny images," Techinical Report (2009).
[28] J. Deng, *et al.*: "ImageNet: a large-scale hierarchical image database," CVPR (2009) 248 (DOI: 10.1109/CVPR.2009.5206848).
[29] S.K. Esser, *et al.*: "Learned step size quantization," ICLR (2020).
[30] A. Paszke, *et al.*: "PyTorch: an imperative style, high-performance deep learning library," NeurIPS (2019) 8024.
[31] S. Li, *et al.*: "CACTI-P: architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques," ICCAD (2011) 694 (DOI: 10.1109/ICCAD.2011.6105405).