ABSTRACT

Data quantization is an effective method to accelerate neural network training and reduce power consumption. However, it is challenging to perform low-bit quantized training: the conventional equal-precision quantization will lead to either high accuracy loss or limited bit-width reduction, while existing mixed-precision methods offer high compression potential but failed to perform accurate and efficient bit-width assignment. In this work, we propose DYNASTY, a block-wise dynamic-precision neural network training framework. DYNASTY provides accurate data sensitivity information through fast online analytics, and maintains stable training convergence with an adaptive bit-width map generator. Network training experiments on CIFAR-100 and ImageNet dataset are carried out, and compared to 8-bit quantization baseline, DYNASTY brings up to 5.1× speedup and 4.7× energy consumption reduction with no accuracy drop and negligible hardware overhead.

CCS CONCEPTS

• Computing methodologies → Neural networks; Philosophical/theoretical foundations of artificial intelligence.

KEYWORDS

fully-quantized network training, mixed-precision quantization, neural network training acceleration

1 INTRODUCTION

Neural networks (NN) have been widely adopted in many fields, including computer vision[14], speech recognition[1], and natural language processing [18]. However, training NN usually needs large computing power, which requires high-cost cloud servers and makes it infeasible to train on low-power edge devices. Researchers have been studying various methods for NN training acceleration, such as exploiting data pruning algorithm[6] and developing more efficient hardware accelerators[21].

As NN training is usually carried out with 32-bit float point data, low-bit network quantization can be an effective way for NN training acceleration. There are 3 ways for NN quantization: post-training quantization[3], quantization aware training[12], and fully-quantized training[22, 23]. The first 2 methods only quantize network inference data and do not accelerate training process, and they usually need to train auxiliary NN or use methods like evolutionary search, leading to even longer training time[19]. Meanwhile, fully-quantized training aims at online quantization of all training data, which can significantly reduce the computation cost needed for training.

Early works on fully-quantized training adopt an equal bit-width for the entire network[23], which leads to either high accuracy loss or limited bit-width reduction. Later works utilize mixed-precision training to provide better acceleration while maintaining network accuracy. Some works like [9] assign different bit-widths to each network layer, and state-of-the-art quantization results are acquired through block-wise mixed-precision training[4]. As shown in Fig. 1, block-wise mixed-precision training divides training data into small blocks, and quantizes them into different bit-widths. Whether layer-wise or block-wise, mixed-precision training is challenging in how to perform online precision assignment for each layer/block of data. Some of the existing works use data quantization sensitivity acquiring algorithms such as calculating Hessian eigenvalue and trace[8, 9], which are too complicated for hardware implementation; other works utilize heuristic methods like greedy search[4], which does not generate accurate bit-width distribution.

To address the aforementioned weakness, we propose DYNASTY (DYnamic Analytics of Sensitivity TY), a block-wise dynamic-precision NN training framework. By using analytics methods, DYNASTY performs fast dynamic data bit-width assignment, and achieves...
average 2-bit quantization with minimum accuracy drop. Key contributions of the work are concluded as follows:

- A software-hardware co-designed block-wise fully-quantized mixed-precision NN training framework named DYNASTY. It provides a block-float-point quantized training algorithm with 0-8 bits dynamic block precision, and extends an online bit-width assignment module to the NN acceleration hardware architecture.
- A Relative Quantization Sensitivity Analytics algorithm. By applying relaxed Lagrange duality, it transforms the NP-hard bit-width searching problem into $O(N)$ complexity, providing fast but still accurate online data quantization sensitivity analysis.
- An Adaptive Bit-Width Map Generator that maps data sensitivity to bit-width. It performs online tuning of algorithm hyperparameter to maintain stable average bit-width, and employs bit-width map temporal smoothing to furtherly enhance network accuracy.
- Experiments on our algorithm show 5.1× training speedup with no accuracy drop on CIFAR-100, and 1.9× speedup with 0.39% accuracy drop on ImageNet, compared to 8-bit quantization baseline. Our hardware architecture shows minimum overhead on die area (+1.4%) and brings at most 78.8% energy consumption reduction.

2 PRELIMINARY & RELATED WORKS

2.1 Quantized NN Training Basics

While NN inference requires only one forward pass computation, training involves three different stages: forward, backward and weight update. Each stage includes multiply-accumulate (MAC) operations on different data: for the forward stage, they are activations and weights; for the backward stage, they are activation gradients (also called errors) and weights; and for the backward stage, they are activations and activation gradients.

Researchers have found training quantization to be more challenging than inference. Xiao Sun et al. [16] point out that quantization saturation can be severe in training due to the large dynamic range of gradient data in the backward stage. Later work [17] furtherly discover that different network layers show widely different ranges of gradients across training epochs.

To better suit these different data distributions, either hand-crafted quantization formats for different layers and stages need to be carefully designed as having been done in [16, 17], or better we employ mixed-precision quantization that adaptively adjusts the quantization method for different data in networks.

2.2 Mixed-Precision NN Training

Early works of mixed-precision fully-quantized training such as HAWQ [9] and AdaQS [10] only calculate a relative ordering of the importance of each data block. They require empirically assigning bit-width to each block, making their method design include trial and error and hard to be accurate.

A better way to get bit-width distribution is through analytics methods, where the bit-width distribution solving is formulated as an optimization problem to minimize impacts brought by quantization noise. ActNN [4] proposes to define the mean square of weight gradients as the optimization target, but this optimization is hard to solve and they have to use greedy search for a heuristic result. HAWQ-V3 [20] needs to solve an integer linear programming problem, and MPQC0 [5] formulates it into a multiple-choice knapsack problem, both of which are NP-hard to solve.

3 BLOCK-WISE DYNAMIC-PRECISION NN TRAINING FRAMEWORK

3.1 Quantization Training Framework

Figure 2 depicts the overall framework of our system, where the DYNASTY module extends an NN accelerator for block-wise dynamic-precision training. Input data of all training stages are quantized by Block-Float-Point (BFP) format, which divides data into 4 × 4 blocks. Data in each block have mantissa quantized into a same bit-width, and they also share a single 8-bit exponent. Two strategies are used for mantissa-bit-width assignment: in each training stage, one of the input data is statically quantized to 8-bit mantissa; sensitivity analytics is performed for the other input data, whose mantissa is dynamically quantized into 0-8 bits according to the bit-width map generated by the DYNASTY module.

The DYNASTY module is in charge of the dynamic bit-width assignment, and consists of two sub-modules. The first sub-module, named Relative Quantization Sensitivity Analytics, receives the current quantization range together with data gradients, and generates a relative quantization sensitivity value for each block. This analytics happens at the end of the training of each mini-batch. This sub-module will be detailed in Section 3.2. The second sub-module, the Adaptive Bit-Width Map Generator, works at the end of each training epoch. It maps the sensitivities into a bit-width distribution and maintains stable training. This mapping happens at the end of each training epoch. Details of this sub-module will be provided in Section 3.3.

1 Block division is along $C_{in}$ and $C_{out}$ dimensions for weight and weight gradients, $C_{in}$ and $Batch$ dimensions for activations, and $C_{out}$ and $Batch$ dimensions for activation gradients.
Figure 3: An example of per-layer relative sensitivity distribution. Red markers represent the mean sensitivity per layer, and different colored squares represent the proportion of different sensitivity ranges in each layer.

3.2 Relative Quantization Sensitivity Analytics

To perform fast online data analysis and dynamically identify which data is more important and need larger quantization bit-width than others, the Relative Quantization Sensitivity Analytics module is designed to calculate data quantization sensitivity based on their quantization range and corresponding gradients. We first establish an optimization problem based on the minimization of first-order mean square loss perturbation. Then we propose to solve the optimization by relaxed Lagrange duality method and get the sensitivity values.

3.2.1 Optimization Problem Based On First-Order Mean Square Loss Noise. Quantization brings noise to network training process, leading to less stable training and lower network accuracy. An ideal bit-width assignment should minimize this noise and bring minimum perturbation to the training loss. We propose to use the first-order mean square loss noise($\Delta L^2$) to quantify this perturbation. The math formula of $\Delta L^2$ caused by weight quantization is shown as Eq. (1). It is calculated as the mean square value of first-order loss noise caused by each quantized data. And we use $S_i$ to denote the ratio of loss noise to quantization levels in each data block. In Eq. (1), $\Delta W_{i,j}$ is the gradient of the $j$th weight of the $i$th block; $\sigma_{i,j}$ is the quantization noise of corresponding data; $N$ is the total number of data blocks; $B$ is the number of values inside each block; $b_i$ is the quantization bit-width of the mantissa of the $i$th block of weights; and $e_j$ is the exponent of each block so that $2^{e_j}$ is the corresponding quantization range. Note that, $\Delta L^2$ brought by activation quantization can also be calculated in the same way, and we omit its formula here for simplicity. With our formulation, $\Delta L^2$ is calculated from the quantization range of each block and their corresponding gradient values, both of which are already known during NN training. We do not choose higher-order representations for loss noise, as that would require much more complex computation which is not suitable for hardware implementation, and our experiments show that first-order estimation is sufficient for training large networks.

$$\Delta L^2 = \frac{1}{NB} \sum_{i=1}^{N} \sum_{j=1}^{B} (\Delta W_{i,j} \cdot \sigma_{i,j})^2 = \frac{2}{3NB} \sum_{i=1}^{N} S_i (2^{-b_i})^2$$  
(1)

$$S_i = \frac{(2^{e_i})^2}{16} \sum_{j=1}^{B} \Delta W_{i,j}^2$$

The full optimization is shown as Eq. (2). Aside from $\Delta L^2$ as our optimization target function, we also specify the optimization constraints. The $\alpha$ denotes the desired average computation bit-width,

$$\arg \min_{b} \ f(b) = \Delta L^2$$  
(2a)

subject to

$$\sum_{i=1}^{N} T_i b_i \leq \alpha$$  
(2b)

$$b_i \in \mathbb{Z}$$  
(2c)

$$0 \leq b_i \leq \beta$$  
(2d)

3.2.2 Problem Solving for Relative Quantization Sensitivity. Optimization in Eq. (2) is a mixed integer convex programming problem. It is NP-hard to solve and too complicated for hardware implementation. Thus we propose to relax the restriction in Eq. (2c) to the $\mathbb{R}$ set, and then apply Lagrange duality to this problem. This leads to the solution shown in Eq. (3). The bit-width is calculated from value $r$ and a hyperparameter $\lambda$. $r$ is easy to compute based on Eq. (3b). It is positively correlated with $S_i$ and negatively correlated with $T_i$, and only requires $O(N)$ time complexity for weights and activations of the entire network. $r$ tells the relative bit-width between data blocks, and we call it the relative quantization sensitivity. Though we still do not know the value of $\lambda$, we successfully transform the searching of bit-width of $N$ different blocks into the searching of only a single hyperparameter. We call $\lambda$ the global quantization coefficient.

$$b_i = \max(0, \min(r_i - \lambda, \beta))$$  
(3a)

$$r_i = \frac{1}{2} \log_2 \frac{S_i}{T_i}$$  
(3b)

An example of $r$ distribution during training of ResNet-18 is shown as Fig. 3. We can see that the desired bit-width distribution should be different across network layers. Data in several layers requires much larger bit-widths than others, including weights in the first, the last, and the 3 shortcut layers, and activation in the last layer. Bit-width distribution inside each layer is also different, and activations’ bit-widths tend to be more spread out and reside in larger ranges than weights’. These observations suggest that manually setting bit-width can hardly suit the data requirement, and online bit-width assignment based on data sensitivity is needed.

However, though Eq. (3b) uncovers the relative bit-widths between data blocks, it still does not give us the absolute bit-width of each block of data. Two problems remain to be solved. The first

374
The Adaptive Bit-Width Map Generator aims to solve the remaining quantization problem, we propose a bit-width map temporal smoothing algorithm to dynamically determine the coefficient during training. In theory, if we ignore these upper and lower bounds constraints in Eq. (3a), an approximate of $\lambda$ can be easily deduced as $\lambda = \sum_i b_i - Na$, where $N$ is the number of blocks. With experiments, we discover that in the initial training of a network, differences in the importance of data in the network are not much and seldomly make $b_i$ reach the lower or upper bound, thus $\lambda$ is a good estimation of the initial $\lambda$ value.

However, as shown in Fig. 5, using only $\lambda$ leads to the average bit-width becoming larger than desired as training continues, especially for activation data. To prevent $\lambda$ drifting away from the desired value, we design Algorithm 1, which tunes $\lambda$ with linear interpolation using the last $\lambda$ value as the start point of the interpolation. The number of interpolation iterations $L$ is empirically set to 3. By using this method, average bit-width can be maintained stably at the target bit-width, as the blue lines show in Fig. 5.

3.3.3 Prevent Bit-Width Fluctuation. To solve the bit-width fluctuation problem, we propose a bit-width map temporal smoothing method: at every epoch, an exponential moving average is performed on the bit-width map, as shown in Eq. (4). This prevents sudden changes in data bit-width caused by any calculation error. The smooth factor $\gamma$ is empirically set to 0.5.

$$b_{\text{L,smoothed}} = \gamma \cdot b_{\text{L last}} + (1 - \gamma) \cdot b_{\text{L new}}$$

(4)

Note that, during the first training epoch when we have not got the value of $\lambda$, we assign 4 bits to every block as the initial bit-width map. And for hardware simplicity, we perform rounding on the bit-width map to only use 0, 2, 4, 6, 8 as possible quantized bit-widths. Specially, when a block is quantized to 0 bit, we skip the computation of this block. The smoothed bit-width map is then sent to the accelerator computing cores for dynamic data quantization.

![Figure 5: Real network bit-width when targeting 3-bit quantization. Without linear interpolation-based tuning of $\lambda$, the average bit-width tends to become larger than desired.](image)

**Algorithm 1** Global Quantization Coefficient Tuning

**Input:** $\lambda_{in}$: the last global quantization coefficient $r$: relative sensitivity $a$: target averaged quantization bit-width

**Output:** $\lambda_{out}$: the tuned global quantization coefficient

**Parameter:** $L$: number of interpolation iterations

1. **begin**
2. $\lambda_{out} = \lambda_{in}$
3. $b_0, \lambda_1 \leftarrow \min(r) - a, \max(r) - a$
4. **for** $l = 1$ to $L$ **do**
5. $b_0 \leftarrow \frac{1}{N} \sum_i \max(r_i - \lambda_0, \beta)$, 0
6. $b_1 \leftarrow \frac{1}{N} \sum_i \max(r_i - \lambda_1, \beta)$, 0
7. $\tilde{b} \leftarrow \frac{1}{N} \sum_i \max(r_i - \lambda, \beta)$, 0
8. **if** $\tilde{b} > a$ **then**
9. $t \leftarrow \frac{(\tilde{b} - a)}{(\tilde{b} - b_1)}$
10. $\lambda_0, \lambda_{out} \leftarrow \lambda_{out}, (1 - t) \lambda_{out} + t \cdot \lambda_1$
11. **else**
12. $t \leftarrow \frac{(\tilde{b} - a)}{(\tilde{b} - b_0)}$
13. $\lambda_{out}, \lambda_1 \leftarrow (1 - t) \lambda_{out} + t \cdot \lambda_0, \lambda_{out}$
14. **end if**
15. **end for**
16. **end**

![Figure 6: DYNASTY module hardware architecture](image)

4 HARDWARE ARCHITECTURE

Support of variable bit-width operation has been realised through NN accelerators with bit-serial-based computation array[15, 21]. These accelerators get linear speedup as the operation bit-width shrinks. The DYNASTY hardware architecture extends such accelerators with online bit-width assignment ability.

Fig. 6 shows the hardware architecture of the DYNASTY module. The use of 16 parallel sensitivity analytics units makes it capable of calculating the sensitivities and generating bit-width map for 16 blocks, i.e. 256 weights or activations in one cycle, enabling the per batch sensitivity analytics to be completed in only 16% network training time for 2-bit ResNet-18 on ImageNet dataset. Note that, the DYNASTY module is running in parallel with the main array.
of the accelerator. A recovery remainder-based divider is used in the Bit-Width Map Generator, but since there is only one divider, and the divisions only happen at the end of each training epoch, it does not bring much area and power overhead.

## 5 EXPERIMENTAL RESULTS

### 5.1 Experiment Setup

In our experiments, we train ResNet-18 network[11] on the CIFAR-100[13] and ImageNet[7] dataset. For both datasets, we adopt a batch size of 128, and use SGD as the optimizer with an initial learning rate of 0.1. For CIFAR-100, we train the network for 200 epochs; learning rate linear warming up is used in the 1st epoch; and we perform learning rate decay at the 60th, 120th, and 160th epochs with a decay factor of 0.2. For ImageNet, we train the network for 95 epochs; learning rate linear warming up is used in the first 5 epochs; and we perform learning rate decay at the 30th, 60th, and 80th epoch with a decay factor of 0.1.

We also perform training with 2 other methods aside from DYNASTY as our baseline. The first method is to train with float-32 format without quantization; the second is 8-bit quantized training.

### 5.2 Algorithm Evaluation

#### 5.2.1 Network Accuracy

Table 1 shows the validation accuracy that DYNASTY gets with different quantization bit-width. On CIFAR-100, we achieve 4-bit quantization with no accuracy drop compared to float-32 training; even with 2-bit quantization, we only get an accuracy drop of 1.40% compared to float-32 training and get better accuracy than the 8-bit baseline. On ImageNet, while the accuracy drop with 2-bit quantization is not small (6.25%), the 4-bit training experiment still reveals good accuracy with only 0.88% accuracy drop.

#### 5.2.2 Training Convergence Speed Improvement

Figure 7 presents the accuracy-time curve in the experiments. Note that, the training speed for unquantized float-32 training is not compared here, as the cores of the UNPU accelerator cannot handle float-32 operations.

Because the DYNASTY module runs in parallel with the accelerator cores, we enjoy a linear speedup as the quantization bit-width decreases. Compared to 8-bit training baseline, for CIFAR-100, DYNASTY reaches the same validation accuracy (74%) in 58.3%, 38.9%, 29.2%, and 19.6% training time with 6, 4, 3, 2-bit quantization, respectively; because the better convergence than baseline, the speedup (5.1× with 2-bit) is even higher than expectation (4×).

For ImageNet, DYNASTY needs 75.0% and 52.5% time to get 67% accuracy with 6 and 4-bit quantization, respectively.

#### 5.3 Architecture Evaluation

Table 2 shows the area and power of the DYNASTY module compared to other modules in the accelerator. We can see that DYNASTY brings no more than 1.4% area overhead.

The average power consumption of DYNASTY varies with the network scale and quantization bit-width as these factors affect the duty cycle of the DYNASTY module, and we only show the value.
in a typical situation in Table 2. Details on the energy consumption overhead can be seen in Fig. 8. DYNASTY brings at most 79.7% power consumption overhead to the accelerator, but this overhead is overcome by the energy saving due to shorter training time with lower-bit quantization. When targeting 74% accuracy on CIFAR-100, we get 40.0%, 59.5%, 69.2%, and 78.8% energy consumption reduction when quantizing to average 6, 4, 3, and 2 bits. For ImageNet, we also achieve 23.2% and 45.7% energy reduction with 6 and 4-bit quantization when targeting 67% accuracy.

6 CONCLUSION

In this paper, we propose DYNASTY, a software-hardware co-designed block-wise dynamic-precision neural network training framework. We utilize an analytics method based on network first-order mean square loss noise to efficiently generate data quantization sensitivity online, and develop an Adaptive Bit-Width Map Generator to accurately map sensitivity to bit-width distribution that maintains desired computation reduction and high network accuracy at the same time. Experiments on CIFAR-100 and ImageNet dataset are carried out, and compared to 8-bit quantization baseline, we achieve up to 5.1x speedup and 4.7x energy consumption reduction with no accuracy drop and negligible hardware overhead.

ACKNOWLEDGMENTS

This work was partly supported by the National Key R&D Program under Grant 2021YFB3200903, in part by the National Natural Science Foundation of China under Grant 61934005, the NSF of Beijing Municipality under Grant L211005, Beijing National Research Center for Information Science and Technology, and Beijing Innovation Center for Future Chip.

REFERENCES

[1] Osama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. 2014. Convolutional Neural Networks for Speech Recognition. IEEE/ACM Transactions on Audio, Speech, and Language Processing 22, 10 (Oct 2014). 1533–1545. https://doi.org/10.1109/TASLP.2014.2339736
[2] Ron Banner, Itay Hubara, Eldad Hoffer, and Daniel Soudry. 2018. Scalable methods for 8-bit training of neural networks. In Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc., 3693–3703. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/e82ebc8fb8e1516dcd254d4839f7ba9f08e-Paper.pdf
[3] Ron Banner, Yury Nahtman, and Daniel Soudry. 2019. Post training 4-bit quantization of convolutional networks for rapid-deployment. In Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/5df813389fd4e435bca5d5ff0b0d-Paper.pdf
[4] Jianfei Chen, Lianmin Zheng, Zhenwei Yao, Dequan Wang, Ion Stoica, Michael Mahoney, and Joseph Gonzalez. 2021. ActNN: Reducing Training Memory Footprint via 2-Bit Activation Compressed Training. In Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139), Marina Meila and Tong Zhang (Eds.). PMLR, 1803–1813. https://proceedings.mlr.press/v139/chen21z.html
[5] Weihan Chen, Peisong Wang, and Jian Cheng. 2021. Towards Mixed-Precision Quantization of Neural Networks via Constrained Optimization. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 5350–5359.
[6] Pengcheng Dai, Jianfei Yang, Xucheng Ye, Zonghao Luo, Bing Hu, Zhenguo Tang, and Shunli Zeng. 2022. Exploring the Potential of Low-bit Training for Deep Neural Networks. In Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2022/file/1bf89413eb2e8c485b874757746-Paper.pdf
[7] Zhen Dong, Zhewei Yao, Dayiann Arfenn, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. HAWQ-V2: Hessian Aware trace-Weighted Quantization of Neural Networks. 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., 18518–18529. https://proceedings.neurips.cc/paper/2020/file/d777c035f67bbf95388150b92f5e9ee-Paper.pdf
[8] Zhen Dong, Zhewei Yao, Amin Gholami, Michael W. Mahoney, and Kurt Keutzer. 2019. HAWQ: Hessian AWare Quantization of Neural Networks With Mixed-Precision. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).
[9] Junrong Guo, Wannao Liu, Wang Wang, Jundong Han, Ruxian Li, Yijun Lu, and Songlin Hu. 2020. Accelerating Distributed Deep Learning By Adaptive Quantization. In ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 1603–1607. https://doi.org/10.1109/ICASSP40776.2020.9054164
[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
[11] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical Report.
[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/c3996e2d3bb7d6878c4363e92a466c45b-Paper.pdf
[13] Jianwen Lee, Changhyun Kim, Sanghun Kang, Dongjoon Shin, Sangyoub Kim, and Hoj-Jun Yoo. 2019. UNPU: An Energy-Efficient Deep Neural Network Accelerator With Fully Variable Weight Bit Precision. IEEE Journal of Solid-State Circuits 54, 1 (Jan 2019). 173–185. https://doi.org/10.1109/JSSC.2018.2865499
[14] Xiaowei Sun, Jungwok Choi, Cha-Yu Chen, Naigang Wang, Swagath Venkataramani, Vijayalakshmi (Vij) Srinivasan, Xudong Cui, Wei Zhang, and Kaiish Gopalakrishnan. 2019. Hybrid 8-bit floating Point (HFP) Training and Inference for Deep Neural Networks. In Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/65fe9b4a978a89f352e2112cd2b9f8e-Paper.pdf
[15] Xiao Sun, Naigang Wang, Cha-Yu Chen, Jianmin Ni, Ankur Agrawal, Xiaodong Cai, Swagath Venkataramani, Kaustuv El Maghraoui, Vijayalakshmi (Vij) Srinivasan, and Kaiash Gopalakrishnan. 2020. Ultra-Low Precision 4-bit Training of Deep Neural Networks. 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., 1796–1807. https://proceedings.neurips.cc/paper/2020/file/3f5ee243547de911b035c4a84f5a-Paper.pdf
[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., 5978–6000. https://proceedings.neurips.cc/paper/2017/file/3ee2f423547de911b035c4a84f5a-Paper.pdf
[17] Tianzhe Wang, Kuan Wang, Han Cai, J.I. Lin, Zhijian Liu, Hanrui Wang, Yujun Liu, and Song Han. 2020. APQ: Joint Search for Network Architecture, Pruning and Quantization Policy. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).