Minimizing Energy of DNN Training with Adaptive Bit-width and Voltage Scaling

TaiYu Cheng
Osaka University
t-cheng@ist.osaka-u.ac.jp

Masanori Hashimoto
Osaka University
hasimoto@ist.osaka-u.ac.jp

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401556

***Abstract*—Training DNN mostly relies on GPUs with FP32 format. While FP16 is acknowledged for its advantage of high computation and memory efficiencies for training DNN, the training must be accompanied with techniques dedicated for a particular dataset. Therefore, a hardware engine with a configurable bit-width feature is desirable for covering any datasets and applications. This work proposes an adaptive bit-width and voltage scaling (ABVS) scheme for DNN training. The key idea is to increase fraction bit-width (FB) gradually from a small value according to current training quality (e.g., accuracy, mAP). Since less FB achieves shorter hardware latency, this training scheme concurrently adapts bit-width and voltage scaling and intensify energy reduction. Experimental results show that the ABVS scheme achieves the comparable quality to FP32 with at most 0.5% accuracy drop, but up to 63% energy reduction.**

***Index Terms*—deep learning training, floating-point, configurable bit-width, bit-width scaling, voltage scaling**

I. Introduction

Deep neural network (DNN) has thrived in many applications [1], [2], where there is a trend that deeper models require more computation would attain higher accuracy [3]. Training demands more complicated operations and larger amount of data than inference through tens of iterations, resulting in overwhelming computations. Let us exemplify the CPU time difference between inference and training for ImageNet dataset [4], showing that training with 50 epochs demands 1,300 times CPU time than inference. Although training is basically a one-time effort, the recent increases in the size of training datasets and model complexity strongly motivate us to improve the training efficiency while sustaining the training quality.

DNN is inherently error-tolerant, and then approximate computing (AC) is compatible with DNN. Several representative AC techniques like approximate arithmetic unit [5]–[11], voltage-over-scaling [12], [13], bit-width scaling (quantization) [13]–[17] can be either solely or hybridly applied for gaining efficiency. However, most of them [5], [11]–[14] focus on inference, whereas [8]–[10], [15]–[17] aim at training. Besides, the AC techniques are conventionally applied to forward propagation only, and back propagation rely on exact computation [5], [11]–[14] or additional training stages are required [5], [11]. Although the inference and forward propagation enjoy AC benefits, the efficiency improvement of back propagation in training has less explored despite its importance.

DNN training with GPU often adopts 32-bit floating-point (FP32) since a large dynamic range is necessary for gradient computation in back propagation. Meanwhile, training in which a shorter format is applied to back propagation as well is recently studied supposing that training with FP32 is more than

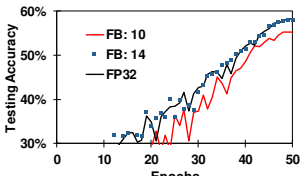

Fig. 1: FP16 (FB:10) cannot attain FP32 training quality (ImageNet).

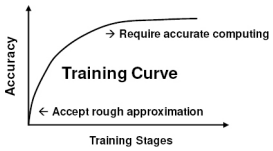

Fig. 2: Gradual training approximation (GTA).

necessary. Especially, FP16 is drawing attention since it improves computation throughput, mitigates memory bandwidth, and reduces power consumption [15]–[17]. However, there is a concern that the FP16 format may not have representation capability enough for modern DNN training. Fig. 1 shows an example of the training curves for the image classification of ImageNet. FB stands for fraction bit-width, where FB = 23 in FP32 format and 10 in FP16. Here, the exponent bit-width is fixed to 8 to sustain the dynamic range, and thus, the training quality entirely depends on FB. We observe that 10-bit FB has a significant gap compared with FP32, and 14-bit FB is necessary to reach the same accuracy with the FP32 case. For certain public datasets, state-of-the-art researches are dedicated to achieving training with FP16 while sustaining the quality. Such sophisticated techniques like mixed-precision training [15], [16], chunk-based accumulation [17], or stochastic rounding [17] enable those datasets to be trained with FP16. However, it is not sure whether FP16 can always guarantee the training quality comparable with FP32 one for a new real-world dataset. Therefore, to pursue both the quality and efficiency in modern DNN training, we consider that a hardware engine possessing FB adjustability, or configurable FB, could be a solution.

This work proposes an adaptive bit-width and voltage scaling (ABVS) scheme to achieve energy minimization for DNN training. The key idea is to adopt less FB in the early training stage while the FB is gradually increased depending on the present training quality. Since the computation with less FB reduces the complexity of the hardware engine and shortens the latency, the extra timing margin can be used for voltage lowering. Note that, in this work, at a training stage (epoch), all the computations in both forward and back propagation share the same data format with one type of FB to guarantee that scaled voltage applies to whole circuit once the FB is determined. Several datasets across different applications, such as CIFAR-10, CIFAR-100, and ImageNet in image classification, and Pascal VOC in object detection, are used for validation. The contributions are:

- ABVS can achieve comparable training quality (0%-0.5% loss) against FP32 for various datasets in size. We also provide a guideline to tackle with a new unknown dataset.
- ABVS can achieve energy reduction by 9% to 37% than

978-1-7281-9201-7/21/$31.00 ©2021 IEEE

training even with "least sufficient FB" without extra iterations (epochs). Compared to common FP32, ImageNet and Pascal VOC enjoy 57%-63% energy reduction.

II. Related work

Table I briefly categorizes the existing works. All the works in the list already involve BWS or approximate arithmetic unit. The AC techniques are applied to only forward propagation in [5], [11]–[14], and the back propagation still relies on FP32 to compensate the error induced in the forward propagation, or [5], [11] execute additional training stages. Besides, the main purpose of [5], [11]–[14] is to perform inference efficiently exploiting approximations for forward computation. Efficiency improvement of back propagation in training is beyond their interests. On the other hand, [8]–[10], [15]–[17] address efficiency improvement including back propagation. Reference [8] applies BWS and approximate multipliers hybridly to training, while [15], [16] adopt training with FP16 and [17] even conducts training with FP8 partially. To enable training with FP16 or even less, sophisticated efforts are required and [15], [16] even demand computations in FP32 partially.

Researches [9], [10] propose a gradual training approximation (GTA) scheme in Fig. 2, inspiring us to develop ABVS. They claim that the early stage in training accepts rough approximation while the late stage requires accurate computing. However, their works focus on the approximation for multipliers only and the accumulators remain FP32 accurate type, where FP32 accumulators would hinder VS efficiency. Besides, large-scale datasets and sophisticated applications are not included in their validations.

III. Proposed ABVS scheme for DNN training

The proposed ABVS scheme adopts less FB in the early training stage while the FB is gradually increased depending on the present training quality. The proposed scheme supposes a configurable FP hardware unit whose FB can be dynamically changed. Furthermore, the configurable FP unit operates at the minimum voltage at which the FP multiply-accumulate (MAC) result is correct for each FB configuration. In this case, the computation with small FB saves power thanks to fewer signal transitions in the FP unit and lower operating voltage. For minimizing training energy, we should keep the FB as small as possible throughout the training process.

Fig. 3 illustrates the procedure of the proposed ABVS scheme. We start the training with a pre-determined smallest FB denoted as FB_{min}, and set the initial A_{check} to 0. For every epoch, the training engine would provide a metric that

TABLE I: Existing works applying AC techniques. FP/BP: forward/back propagation.

Ref.	[5], [11], [14]	[12], [13]	[8]–[10], [15]–[17]	ABVS
FP-AC	✓	✓	✓	✓
BP-AC			✓	✓
VS		✓		✓

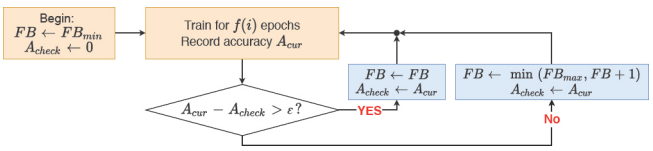

Fig. 3: ABVS flow. Training ends when i reaches T, but this process is omitted here.

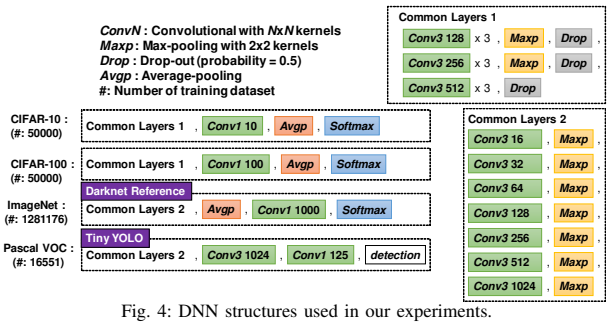

Fig. 4: DNN structures used in our experiments.

we can use for estimating the training quality, e.g., classification accuracy for validation dataset, and we assign it to A_{cur}. Then, at a certain epoch assigned for checking quality improvement, we compare the latest A_{cur} with A_{check}. If the difference is smaller than ε, we increase the FB by 1 for the next epoch, until reaching the pre-determined maximum FB, denoted as FB_{max}. $f(i)$ is a function that determines the schedule of quality checking and gives the checking interval, where i represents the current epoch number. In a simple case, $f(i)$ is constant. T is the total epoch count given to training. This scheme is independent of the training architecture while the amount of VS depends on the architectures and circuits.

When there is preliminary information on the training dataset, the least sufficient FB might be known, where "least sufficient FB" is the FB that can achieve the same quality as FP32. In this case, we can assign it to FB_{max}. For a new dataset, on the other hand, the least sufficient FB is unknown. A guideline for this case is to enlarge the range between FB_{min} and FB_{max}, e.g., $FB_{min} = 6$ and $FB_{max} = 23$. As for the checking schedule, when $FB_{max} - FB_{min}$ is large, frequent checking is necessary to make sure FB_{max} is reachable during the training. On the other hand, when $FB_{max} - FB_{min}$ is small, sparse checking is desirable since sparse checking prevents unnecessary FB elevation originating from the noisy metric trend. We consider this tendency and suggest

$$f(i) = T/(\alpha \times (FB_{max} - FB_{min})), \tag{1}$$

where α is a tuning coefficient. The appropriate α value is experimentally determined in Section IV-B.

IV. Experimental results

A. Evaluation strategy and experimental setup

We evaluate the energy reduction in two steps. The first step checks whether FB scaling keeps the training quality and how much FB can be reduced. The second step estimates the energy reduction supposing a FB configurable FP unit.

For experiments, we apply the proposed ABVS scheme to Darknet [18], one of the most popular frameworks for image classification and object detection. Darknet supports GPU acceleration and is easy to apply in-depth modifications since it is fully developed by C and CUDA based programming. Therefore, we can easily implement the BWS rounding algorithm in it and enjoy the GPU acceleration. We emulate adaptive FB scaling such that the rounding is applied after each basic floating-point computation, where this implementation is leveraged by QPyTorch in [19].

Fig. 4 shows the DNN structures and the datasets used in our experiments. CIFAR-10/100 [20], and ImageNet [4] are for image classification, and Pascal VOC [21] is for object detection. The DNN structure used in our experiments for CIFAR-10/100 is the one recommended by [18]. The DNN structures we selected for ImageNet and Pascal VOC are named Darknet reference [22] and Tiny YOLO [22]. Tiny YOLO is constructed based on Darknet reference, where most of the composition of the layers are identical, while only the last layers are replaced with detector. The developer of Tiny YOLO adopts transfer learning to improve the training quality of YOLO. Therefore, we initialize the weights with those pre-trained for ImageNet.

All training cases are performed with 50 epochs in total. Learning rate decay is applied with a poly-nominal formula, $r_{init} \times (1 - i/50)^P$, where r_{init} is the initial learning rate and i is the current epoch. P is set to 2 for image classification and 1 for object detection. The learning rate becomes 0 once it reaches $i = 50$. Therefore, with this setup of the learning rate, training beyond 50 epochs is meaningless. For applying the ABVS scheme, we use "accuracy" as the metric of training quality and $\varepsilon = 0.005$ for image classification, and we use "mAP" and $\varepsilon = 0.01$ for object detection. In most experiments, FB_{max} is set to the associated least sufficient FB. With this setup, we can demonstrate that the ABVS scheme reduces computation and energy further even compared with the baseline training with the least sufficient FB. For reproducing the situation that prior information is available for each dataset, we trained the NNs with various FBs as preliminary experiments. The obtained FB information is used to determine FB_{min} and FB_{max} for the experiments in the following sections.

B. FB reduction by ABVS

CIFAR-10 and CIFAR-100: Figs. 5 and 6 show the results of the proposed ABVS scheme for CIFAR-10/100, respectively, with two checking schedules described above the figures. Here, FB_{max} is set to 9 for CIFAR-10 and 10 for CIFAR-100, and the FB_{min} is 6 in both cases. The FB varies along the training stage, which corresponds to the right Y-axis. We can see that the duration where FB is lower than 9 in Fig. 5b is longer than in Fig. 5a. Then, the average FB across the entire training process in Fig. 5b is 7.66 and smaller than Fig. 5a despite the second schedule decreases the testing accuracy by 0.3%. Similar observations are found in the results of CIFAR-100 of Fig. 6. Smaller average FB involves a small penalty of the accuracy, but the FB drop is not significant. Even while the training in Fig. 6b ends with 9 bits, the accuracy difference is only 0.4%. The energy reduction obtained from a smaller average FB will be discussed in the next section.

Next, consider the situation that the dataset is new and unknown. We carried out the training with ABVS using $FB_{min} = 6$ and $FB_{max} = 23$. Fig. 7 shows the results for CIFAR-10/100 with different checking schedules. You can see that checking for every two or three epochs works well since the training quality is the same or almost comparable as FP32 one while the FB is smaller. These results indicate that α is

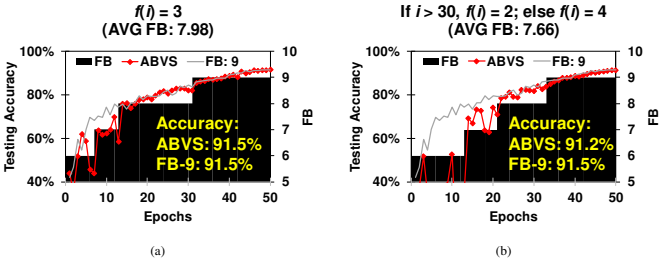

Fig. 5: ABVS results w/ two checking schedules (CIFAR-10).

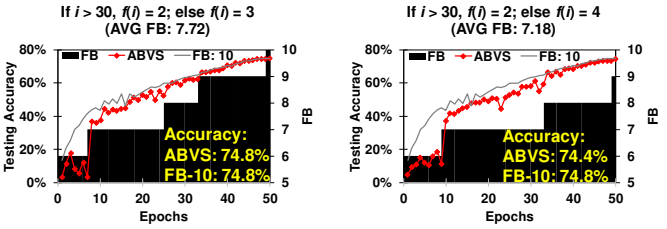

Fig. 6: ABVS results w/ two checking schedules (CIFAR-100).

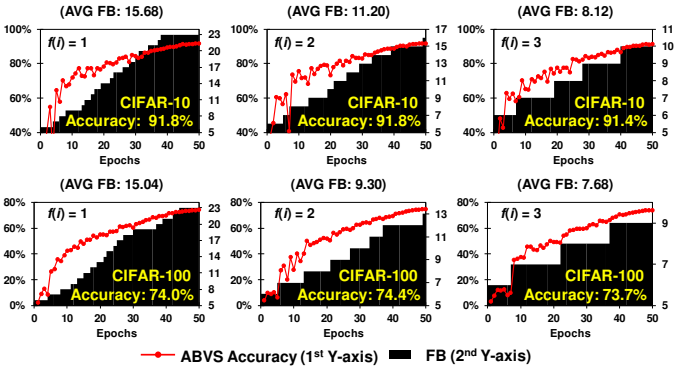

Fig. 7: Applying ABVS flow ($FB_{max} = 23$, CIFAR-10/100).

between 1 and 2. We would suggest such $f(i)$ in Eq. (1) to apply ABVS for the training with a new dataset.

ImageNet: Next, we evaluate ABVS on ImageNet [4]. Fig. 8a shows the results, where FB_{min} and FB_{max} are set to 11 and 14, respectively. The training curve of ABVS traces that of FB $= 14$, and 0.5% accuracy loss is considered acceptable.

Pascal VOC: Figs. 8b demonstrates the results of Pascal VOC, where FB_{min} and FB_{max} are 10 and 12, respectively. The mAP degradation of 0.5% compared with FB $= 12$ is still considered acceptable. The above results show the applicability of the ABVS scheme even for a large-scale dataset and more sophisticated networks. Note that, ImageNet and Pascal VOC cases all indicate training with FP16 (FB: 10) is not really convincing to meet FP32 quality. On the other hand, though detailed parameters tuning are required, the ABVS scheme is not confined by dataset and is applied to any new dataset.

Discussion: Let us discuss the reason why ABVS achieves a similar training quality with fewer FB. K. You, et al. point out that a large learning rate in the early training stage perturbates the training, which prevents the network from memorizing noisy data and results in better generality [23]. On the other hand, BWS injects noise originating from the FB truncation error, especially in the earlier stage with smaller FB. Meanwhile, the large learning rate at the beginning may tolerate

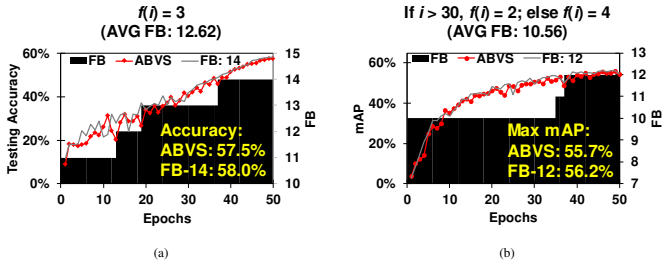

Fig. 8: ABVS results for (a) ImageNet and (b) Pascal VOC.

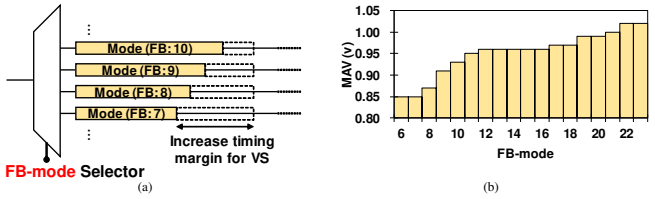

Fig. 9: (a) Assumed FP-MAC with FB configurability. (b) MAV results for each FP-MAC with different FBs.

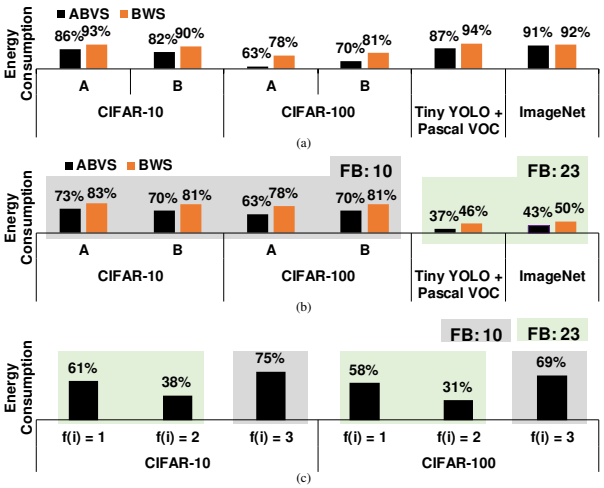

Fig. 10: Energy of ABVS training (a) normalized by that of least sufficient FB (b) normalized by that of FB: 10 or FB: 23 (c) normalized by that of FB: 10 or FB: 23 for CIFAR-10/CIFAR-100 with unknown dataset treatment.

larger noise, which provides high compatibility with ABVS.

C. Energy reduction by ABVS

This section evaluates the power reduction thanks to ABVS. We suppose, in DNN training, the majority of power consumption originates from MAC computation, especially in convolution layers. Thus, we prepare a hardware unit with configurable FB, which is shown in Fig. 9a. Note that developing a more sophisticated FB-configurable unit is our future work. The input of "FB mode selector" determines the FB for MAC operation. We implemented it with verilog and synthesized it with Nangate 45nm library at 1.0 GHz by Synopsys Design Compiler.

An important design consideration is that the FP-MAC should be able to operate at the lowest voltage for each FB configuration. For this purpose, the FP-MAC consists of separate circuits with different FBs that are synthesized individually at the lowest voltage achieving the same operating frequency. Fig. 9b shows the architecture of each FP-MAC, where $N = FB + 9$ since sign and exponent bits are also included. To include optimized floating-point multiplier and adder modules, we used Synopsys DesignWare IP in synthesis. The minimum acceptable voltage (MAV) for each N-bit FP-MAC is defined as the voltage at which the circuit can be synthesized with 0 worst negative slack (WNS). The MAV results are shown in Fig. 9b within the range 0.85V-1.02V.

For power estimation, we prepared a test input pattern representing convolution computation. The power of each N-bit FP-MAC is reported by Synopsys PrimeTime using the logic simulation result. Then, we estimate the training energy as

$$Energy = T_{epoch} \times (\sum_{i} (P^{(i)} \times N^{(i)}_{epoch})), \qquad (2)$$

where T_{epoch} is the computation duration for one epoch, $P^{(i)}$ is the power of i-bit FP-MAC and the number of epochs in which i-bit FP-MAC is applied (denoted as $N^{(i)}_{epoch}$). Here, we take Fig. 5b as an example to explain energy improvement calculation. T_{epoch} is assumed to be identical for all the epochs because of the same clock frequency. We count the number of epochs for each FB-mode (13, 8, 12, 17 epochs for 6, 7, 8, 9 FBs, respectively) and multiply them with the corresponding power values obtained by PrimeTime. Then, based on Eq. (2), we obtain energy. Finally, we calculate the improvement by comparing to the energy for FB: 9 (FB_{max} of CIFAR-10 and the least sufficient FB) times 50 (=13+8+12+17) epochs.

Fig. 10a shows the energy reductions, which correspond to Figs. 5, 6, 8a, and 8b. The values in Fig. 10a are normalized by the energy consumed by the training with the least sufficient FB. The results show that even comparing with the training with the least sufficient FB, the proposed ABVS can achieve 9% to 37% energy reduction.

Let us assume another case that only either of FP16 and FP32 is choosable. In this case, the proposed ABVS scheme provides larger values of energy saving. Fig. 10b shows the energy reduction, where the energies of CIFAR-10/100 are normalized by those of FB $= 10$ (the same precision as FP16), and the energies of ImageNet and YOLO are normalized by those of FP32. The ABVS achieves larger energy reduction. Especially for ImageNet and YOLO, 57%-63% energy reduction is achieved. On the other hand, if the VS is not applied, i.e., only BWS is applied, the energy reduction becomes less, as shown in Fig. 10a. There is a 15% difference in CIFAR-100. Thus, simultaneous FB and VS in the proposed scheme are effective. Finally, the energy reduction for CIFAR-10/100 with unknown dataset treatment are performed in Fig. 10c, where the ABVS can reduce energy by 25%-69%.

V. Conclusion

In this paper, we proposed the ABVS scheme for DNN training to minimize energy consumption. With a hardware unit with FB configurability, we can concurrently perform bit-width and voltage scaling during training, which enlarges energy saving. The proposed ABVS is proved to apply to various datasets across different applications with negligible quality loss (less than 0.5%) while saving up to 63% energy comparing to FP16 or FP32 cases. Also, up to 37% energy is reduced even comparing to the training with the least sufficient FB.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[2] S. Siddiqui, G. Rasool, R. P. Ramachandran, and N. C. Bouaynaya, "Using deep speech recognition to evaluate speech enhancement methods," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–7.

[3] B. Zhang, A. Davoodi, and Y. H. Hu, "Exploring energy and accuracy tradeoff in structure simplification of trained deep neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 4, pp. 836–848, 2018.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014. [Online]. Available: http://arxiv.org/abs/1409.0575

[5] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan, "Axnn: Energy-efficient neuromorphic systems using approximate computing," in *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2014, pp. 27–32.

[6] M. Imani, D. Peroni, and T. Rosing, "Cfpu: Configurable floating point multiplier for energy-efficient computing," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.

[7] M. Imani, R. Garcia, S. Gupta, and T. Rosing, "Rmac: Runtime configurable floating point multiplier for approximate computing," in *Proceedings of the International Symposium on Low Power Electronics and Design*, ser. ISLPED '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3218603.3218621

[8] T. Cheng, Y. Masuda, J. Chen, J. Yu, and M. Hashimoto, "Logarithm-approximate floating-point multiplier is applicable to power-efficient neural network training," *Integration*, vol. 74, pp. 19–31, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926019305826

[9] M. Imani, M. Masich, D. Peroni, P. Wang, and T. Rosing, "Canna: Neural network acceleration using configurable approximation on gpgpu," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 682–689.

[10] M. Imani, P. Wang, and T. Rosing, "Deep neural network acceleration framework under hardware uncertainty," in *2018 19th International Symposium on Quality Electronic Design (ISQED)*, 2018, pp. 389–394.

[11] J. Kung, D. Kim, and S. Mukhopadhyay, "A power-aware digital feedforward neural network platform with backpropagation driven approximate synapses," in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2015, pp. 85–90.

[12] S. Kim, P. Howe, T. Moreau, A. Alaghi, L. Ceze, and V. Sathe, "Matic: Learning around errors for efficient low-voltage neural network accelerators," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 1–6.

[13] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Dvafs: Trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 488–493.

[14] Y. Mishchenko, Y. Goren, M. Sun, C. Beauchene, S. Matsoukas, O. Rybakov, and S. N. P. Vitaladevuni, "Low-bit quantization and quantization-aware training for small-footprint keyword spotting," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 706–711.

[15] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," *CoRR*, vol. abs/1710.03740, 2017. [Online]. Available: http://arxiv.org/abs/1710.03740

[16] D. D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen, J. Yang, J. Park, A. Heinecke, E. Georganas, S. Srinivasan, A. Kundu, M. Smelyanskiy, B. Kaul, and P. Dubey, "A study of BFLOAT16 for deep learning training," *CoRR*, vol. abs/1905.12322, 2019. [Online]. Available: http://arxiv.org/abs/1905.12322

[17] N. Wang, J. Choi, D. Brand, C. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," *CoRR*, vol. abs/1812.08011, 2018. [Online]. Available: http://arxiv.org/abs/1812.08011

[18] J. Redmon, "Darknet: Open source neural networks in c," 2013-2016. [Online]. Available: http://pjreddie.com/darknet

[19] T. Zhang, Z. Lin, G. Yang, and C. D. Sa, "Qpytorch: A low-precision arithmetic simulation framework," *CoRR*, vol. abs/1910.04540, 2019. [Online]. Available: http://arxiv.org/abs/1910.04540

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 2012.

[21] M. Everingham, S. M. Eslami, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vision*, vol. 111, no. 1, p. 98–136, Jan. 2015. [Online]. Available: https://doi.org/10.1007/s11263-014-0733-5

[22] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[23] K. You, M. Long, M. I. Jordan, and J. Wang, "Learning stages: Phenomenon, root cause, mechanism hypothesis, and implications," *CoRR*, vol. abs/1908.01878, 2019. [Online]. Available: http://arxiv.org/abs/1908.01878