Accuracy Boosters: Epoch-Driven Mixed-Mantissa Block Floating Point for DNN Training

Simla Burcu Harma*, Ayan Chakraborty*, Babak Falsafi*, Martin Jaggi*, Yunho Oh†,
**EcoCloud*, EPFL *simla.harma@epfl.ch, ayan.chakraborty@epfl.ch babak.falsafi@epfl.ch martin.jaggi@epfl.ch*
†*ComSys*, Korea University *yunho_oh@korea.ac.kr*

***Abstract*—The unprecedented growth in DNN model complexity, size, and amount of training data has led to a commensurate increase in demand for computing and a search for minimal encoding. Recent research advocates Hybrid Block Floating Point (HBFP) to minimize silicon provisioning in accelerators by converting the majority of arithmetic operations in training to 8-bit fixed point. In this paper, we perform a full-scale exploration of the HBFP design space using mathematical tools to study the interplay among various parameters and identify opportunities for even smaller encodings across layers and epochs. Based on our findings, we propose *Accuracy Boosters*, an epoch-driven mixed-mantissa HBFP technique that uses 6-bit mantissas only in the last epoch and first/last layers, and 4-bit mantissas for 99.7% of all other arithmetic operations in training. Using analytic models, we show Accuracy Boosters enable increasing arithmetic density for an HBFP training accelerator by up to $21.3\times$ compared to FP32 and up to $4.4\times$ compared to another SOTA format BFloat16, while preserving or outperforming FP32 accuracy.**

I. Introduction

Over the past decade, improvements in Deep Neural Network (DNN) algorithms have led to unprecedented growth in model complexity and dataset size and, consequently, the required computational resources to train DNN models. One of the largest DNN models (GPT-3) [2] has 175 billion parameters and requires 3.14×10^{23} FLOPs to train. With the slowdown in Moore's law, researchers and vendors have begun to search for alternate ways to improve the arithmetic density of the underlying hardware platforms. Narrower bit-width (with lower precision) number formats [24], [25], [31], [32], [35] have emerged as a promising approach to increase arithmetic density, as well as, reduce the required operand storage and communication bandwidth while maintaining high training accuracy.

Recently there have been several proposals for block floating point [7], [20], [38], a numerical encoding that groups a block of mantissas which rely on only fixed-point arithmetic with a single exponent. Block floating point asymptotically approaches the arithmetic density of fixed point with larger block sizes and naturally lends itself well to mixed-precision hardware where a block with the same number of exponent bits can have a fixed-point datapath which is bitsliced for various multiples of mantissa bit encodings (e.g., the same way as today's CPU cores implement SIMD). While block floating point has been promising in use for inference (e.g., Microsoft Floating Point [6]), most proposals to train with block floating point have either failed to reach its full potential by requiring small blocks and/or just fall short of reaching FP32 accuracy.

One specific proposal, Hybrid Block Floating Point (HBFP) [10], uses a mixed-precision format where the dominant fraction of training which are the dot products, happen in block floating point (e.g., convolutions, matrix multiplications), and FP32 is used for other less frequent operations requiring larger numerical ranges (e.g., activations, regularizations). HBFP simultaneously offers the high accuracy of floating point and the superior hardware density of fixed point, delivering up to $8.5\times$ higher throughput than FP16 with $2\times$ more compact models [11]. Prior work on HBFP only presented a preliminary design space analysis for power-of-two mantissa bit widths (e.g., 2, 4, 8 bits).

In this paper, we make the observation that the parameter space for HBFP is quite rich, presenting several opportunities for further improving efficiency and density in hardware platforms. First, custom accelerators can support non-power-of-two numerical formats, and minimizing the number of bits improves operand storage and communication linearly and arithmetic logic quadratically. Second, there is an interplay between the block size and the number of mantissa bits, allowing for an overall denser numerical format with smaller blocks while maintaining accuracy. Finally, HBFP allows for mixed-mantissa block floating point encodings. Prior work studies training with various HBFP formats in isolation; however, the design space of mixed-mantissa HBFP is yet to be explored.

We fully explore the parameter space of HBFP and show the boundaries of block floating point by studying the interplay between the block size and the number of mantissa bits. To the best of our knowledge, this is the first paper conducting a full design space exploration for training DNNs with block floating point. We show that HBFP6 (HBFP with 6 bits of mantissa) is the smallest HBFP format achieving competitive accuracies with no sensitivity to block size. Our main contribution is the design of Accuracy Boosters, a DNN training mechanism performing a large fraction of epochs in low precision, i.e. HBFP4. Our method improves epoch-wise mixed-precision training by introducing high precision, i.e. HBFP6, to the training process only at the last epoch. Accuracy Boosters enable increasing arithmetic density by up to $21.3\times$ compared to FP32, and up to $4.4\times$ compared to another SOTA format BFloat16, while preserving or outperforming FP32 accuracy.

II. HBFP Parameter Space

HBFP is a mixed-precision DNN training technique that uses block floating point for all dot product operations and

FP32 for the rest of the operations, enabling accurate training with dense fixed-point arithmetic. We observe that HBFP is also suitable for inference for the popular CNN and Transformer models without any accuracy loss, in line with prior work on inference with block floating point [6], showing that HBFP is a versatile technique for both training and inference. Prior work on HBFP shows that the area and energy expenditure of HBFP8 is around an order of magnitude lower than FP32 [11]. Exploring the parameter space of HBFP and pushing its boundaries can increase this ratio dramatically.

HBFP has a rich parameter space, including the number of mantissa bits, block size, and the number of exponent bits. The hardware area and energy expenditure of HBFP accelerators are determined by the number of mantissa bits and the block size because the overhead of the exponent bits is negligible due to blocking[1]. One of the key advantages of HBFP is that we can conservatively find a lower bound for the number of exponent bits that covers all of the design space exploration for block size and the number of mantissa bits. Therefore, we work with 10-bit exponents as in prior work [10] and explore the HBFP design space by varying the mantissa bit width and the block size. Once we fix the number of exponent bits, we can vary other parameters, which enables a reconfigurable microarchitecture and gives rise to mixed-mantissa HBFP.

Smaller mantissa bit widths and larger block sizes are key to improving block-floating-point hardware efficiency due to the increasing fraction of fixed-point operations [6]. There is an interplay between the number of mantissa bits and block sizes, allowing for an overall denser numerical format with smaller blocks while maintaining accuracy. This interplay is the result of how the block floating point conversion works. Block floating point shares a single exponent across a block of values using the exponent of the largest element. Since block floating point format does not apply normalization (It is calculated as $2^{exponent} \times 0.mantissa$ instead of $2^{exponent} \times 1.mantissa$), the precision within a block is highly dependent on the largest element in that block, which decides the exponent value. The interval between two consecutive representable numbers is calculated as in Equation 1.

$$interval = \frac{2^{largest\ exponent}}{2^{\#\ of\ mantissa\ bits}} \qquad (1)$$

As the number of elements sharing the same exponent increases, the likelihood of disparity in the magnitude of elements also increases, leading to a precision loss for the small elements in the block. As the number of mantissa bits decreases, the model's sensitivity to the block size increases with the corresponding increase in the interval leading to a higher quantization error. More mantissa bits make the distribution more resilient to the quantization error and larger block size, as each element can be represented more accurately.

HBFP's power footprint is not only a function of the HBFP parameters but also of outside factors. Mixed-precision training has emerged as a popular technique to increase the fraction of leaner arithmetic formats within the training process, motivating us to explore the design space of mixed-mantissa HBFP; because HBFP provides the opportunity to fix the exponent width and vary the number of mantissa bits across layers and epochs.

For CNN models, prior work indicates that the first convolution layer and the last fully connected layer have a larger impact on the final model accuracy, and keeping these layers in high precision allows for reducing precision in the rest of the layers [3], [24], [34], [39]. The first layer takes the input images and filters the images with several convolution kernels and returns feature maps. Thus, it is critical for the final model to keep the input information fully accurate and to preserve the data in the initial feature map. Similarly, for Transformers, the first layer is the input embedding layer, where input tokens are mapped to dense word vectors. The last layer of DNN models returns a probability distribution over the possible outcomes for the underlying DNN task. The important roles of the first and last layers in DNN models make it crucial to retain information better for these layers.

In addition to layers, each training epoch has a different effect on the final model's accuracy [13], [14]. [28] and [36] show that DNNs first learn low-frequency components, where the frequency is defined for the coordinates of the input space. [36] also empirically show that for CNN models, the high-frequency components have higher complexities and are learned in the last epochs. In light of these findings, we hypothesize that high-frequency functional components are more sensitive to quantization errors. Thus, higher precision is required for the last stage of DNN training, where the optimization occurs after an appropriate amount of generalization in the network. After reaching a certain loss value in low-precision training, switching the tensors to high precision enable the sensitive fine-tuning performed in the final epochs and help increase the accuracy even more.

III. Minimizing HBFP

Our goal is to minimize HBFP to increase the hardware efficiency of training without losing accuracy. For a block size of 576, even though HBFP4 incurs a $2.4\times$ improvement in area/power relative to HBFP8, it lacks the precision to reach FP32 accuracy. As prior work on HBFP [10], [11] only investigated power-of-two mantissa bits and focused mostly on the design space of HBFP8, the interplay between the number of mantissa bits and the block size is left unexplored. While power-of-two-bit numbers align naturally with the memory structure and encode matrices in a tightly-packed way, non-power-of-two-bit mantissas can improve the arithmetic density even further, as studied by [6] and can be easily integrated into custom accelerators. We investigate the whole design space of HBFP by varying both parameters and claim that reducing the block size will enable reducing the number of mantissa bits, and thus improve hardware efficiency. In this section, we show how to minimize HBFP step by step, give an explanation of the limitations of HBFP and propose a new mixed-precision schema to minimize HBFP further.

[1]Even for the block size of 4, HBFP4 with 5-bit exponent is only $1.1\times$ more area-efficient than HBFP4 with 10-bit exponent

To study the relationship between model accuracy and HBFP parameters, we measure the similarity between block-floating-point and FP32 distributions of various tensors using Wasserstein distance, mathematically defined as in Equation 2.

$$W(P,Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||] \quad (2)$$

where $\Pi(P,Q)$ is the set of all joint distributions $\gamma(x,y)$ whose marginal distributions are equal to P and Q. $\gamma(x,y)$ can be interpreted as the amount of mass that must be transported from x to y to transform P to Q [1]. Unlike KL-Divergence, which is commonly used to compare quantized tensors to their full-precision counterparts [6], [26], Wasserstein distance is symmetric, and thus is mathematically a metric. Moreover, because DNNs often deal with distributions where KL Divergence is not defined (or infinite), we need to add a noise term to the model distribution to be able to use KL Divergence, which causes disturbance in the results.

We observe that the tensor distribution is preserved when the elements are converted to block floating point format with 6 bits of mantissa and wider for reasonably large block sizes[2]. Figure 1 shows Wasserstein distances between FP32 and HBFP6 and HBFP4 with various block sizes for the weight tensors of four different layers of ResNet20 trained on CIFAR10. For all the tensors, HBFP6 has a much smaller distance to FP32, and the distances are fairly close to each other for a given tensor across all block sizes. However, the Wasserstein distance of HBFP4 is more than $3.5\times$ higher than HBFP6 across all block sizes, and the distances dramatically increase with the block size. Indeed, the R-squared (the strength of the relationship between two data sets) values between the model accuracy and various Wasserstein distances are around 0.99, validating the strength of our metric.

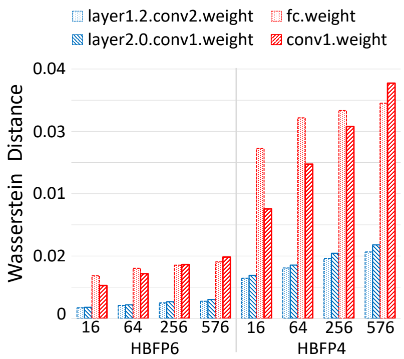

Fig. 1: Wasserstein distance between FP32 and HBFP with various block sizes for various layers.

Even though reducing the block size incurs smaller Wasserstein distances and helps increase the accuracy, HBFP4 still fails to reach FP32 accuracy because it does not have enough precision to minimize the loss and has a high generalization error. [22] introduce a methodology to visualize loss landscapes in order to better understand the effect of loss landscapes on generalization. Figure 2 shows log-scale loss landscapes for various configurations, sliced along the x-axis (y=0) for simplicity. The center of the plot corresponds to the current state of the minimizer, and the axis parameterizes a random direction with filter-wise normalization. HBFP4 converges to a much worse minimum compared to HBFP6 and FP32 indicating poor accuracy. Although the minimum of HBFP4 is flat, it does not indicate better generalization because the minimum itself is much worse.

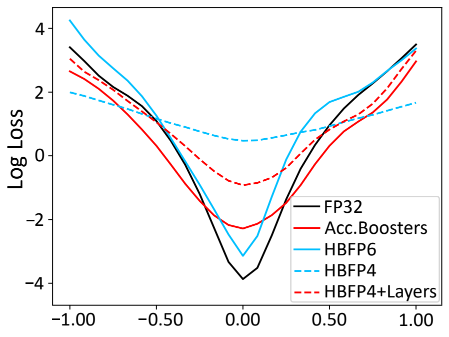

Fig. 2: Loss landscapes of ResNet20 on CIFAR10 for various configurations, sliced along the x-axis.

Following the insights from prior work, we study the effect of the first and last layers of CNNs on model accuracy. The dotted and solid red lines in Figure 1 show the first and last layers, respectively, and it is clear that these layers are the most affected by lowering the precision, especially for HBFP4. Thus, we keep the first and last layers of CNNs in HBFP6 during HBFP4 training to increase its accuracy. However, the increase in precision HBFP6 provides for the first and last layers still does not achieve enough optimization to reach FP32 accuracy. In Figure 2, the red dashed curve shows this configuration, and the curve gets sharper and lower compared to HBFP4-only training. However, the generalization and optimization power of the model is still unbalanced, leading to convergence to another bad local minima.

We introduce Accuracy Boosters, an epoch-driven mixed-mantissa HBFP that uses HBFP6 only in the last epoch and converts 99.7% of all arithmetic operations in training to HBFP4. We hypothesize that using HBFP6 for the last epoch is sufficient to boost the accuracy, while the rest of the epochs are trained using HBFP4. We leverage the insight that last epochs have more effect on the final model's accuracy [13], [14], [28], [36]. We claim that training with 4-bit mantissas helps the model generalize and reach a certain loss value. Afterward, switching to 6-bit mantissas helps the model optimize and fine-tune in the final epochs and increase accuracy to the FP32 level. The loss landscape for Accuracy Boosters (the red solid curve in Figure 2) supports our hypothesis. We see that the curve gets really close (note that the plot is in log scale, thus -2 is closer to -4 compared to 0) to HBFP6 and FP32 curves and finally achieves FP32 accuracy.

[2]Block sizes of up to 256 already achieves more than 95% of the maximum hardware benefit for HBFP6

IV. Experimental Results

We experiment on the state-of-the-art models and datasets for various DNN tasks to test our hypotheses. We train ResNet20/74 [15], and DenseNet40 [16] on CIFAR10 and CIFAR100 [19] datasets for image classification. We also train a Transformer-Base [33] on the WMT16 English-German dataset for machine translation. Models trained on CIFAR10 are trained for 160 epochs, whereas for CIFAR100, the total number of epochs for all models is 300. The transformer is trained for 70 epochs. We use FP32 as the baseline for both model accuracies and hardware comparisons. For the image classification experiments, we report the Top1 validation accuracies; for machine translation, we report the BLEU scores. Moreover, to show the impact of our method, we tune the hyperparameters of FP32 models and then train the same models from scratch with the same hyperparameters in HBFP, showing that our method can be used out of the box without further hyperparameter tuning.

We use an analytic model to estimate the area of the most basic operation in DNN training—Dot product followed by activation unit for different encodings. Fixing the operation enables us to compare the arithmetic density ((operations/s)/area) solely on the amount of area. Thus, we define the gain in arithmetic density to be the same as the gain in the area. For FP32 dot product units of size N, we estimate the hardware cost as the sum of the cost of $N-1$ FP32 adders, N FP32 multipliers, one FP32 accumulator (adder), and one floating-point activation unit. For HBFP dot product units, we estimate the hardware cost as the sum of the cost of $N-1$ fixed-point adders, N fixed-point multipliers, one FP32 accumulator (adder), one floating-point activation unit, and one adder for signed exponents. We also add the costs of conversions between FP32 and fixed-point numbers by modeling the converter blocks.

A. Minimizing HBFP

Table I shows the Top1 validation accuracies for ResNet20, ResNet74, and DenseNet40 on CIFAR10 and CIFAR100 datasets trained with various HBFP configurations. We observe that HBFP6 is the smallest HBFP configuration that gives accuracies within 2% of FP32 accuracy for block sizes up to 256. Larger blocks will contain a larger variety of values in terms of magnitude (affected e.g., by outliers), so it will result in larger approximation errors than smaller blocks and lower accuracy in training.

We also report HBFP4 accuracies to show the limitations of HBFP. Even for the small models like ResNet20, with a block size of 16, the accuracy drops more than 9%. As the accuracy drop for ResNet74 and DenseNet40 on CIFAR100 is considerably high even with HBFP5 (not shown here for compactness), we did not train these models with HBFP4. We observe that for HBFP4, the sensitivity to the block size increases for all the models because the distortions in the tensor distributions increase (see Section II).

TABLE I: Top-1 validation accuracies of various CNN models for various HBFP configurations

		Models and Datasets			
		CIFAR10		CIFAR100	
Number Format	**Block /Area**	ResNet20	ResNet74	ResNet74	DenseNet40
FP32	-	91.72	93.57	74.55	72.42
HBFP8	576/10.0	91.52	93.36	74.32	73.73
HBFP6	16/11.2	91.12	93.38	73.51	72.08
	25/12.3	91.09	92.54	73.20	71.77
	36/13.1	91.29	92.61	72.87	71.83
	49/13.6	91.33	92.93	72.40	71.87
	64/13.9	91.12	92.93	72.40	71.81
	256/14.8	91.38	92.79	72.53	71.50
	576/15.0	90.65	92.19	72.51	71.02
HBFP4	16/15.5	82.59	76.85	-	63.70
	25/17.8	81.82	78.62	-	64.25
	36/19.3	80.84	76.64	-	63.34
	49/20.4	79.32	71.19	-	65.55
	64/21.3	80.18	74.35	-	62.37
	256/23.4	76.96	60.65	-	60.02
	576/23.9	75.33	66.70	-	59.77
Total Number of FLOPs required to train the model		41M	174M	326M	542M

B. Accuracy Boosters

Considering HBFP hardware model, a block size of 64 is within 90% of the maximum area/power gain while achieving accuracies with less than 1% degradation for HBFP6. Thus, we choose block size of 64 as the sweet spot and test Accuracy Boosters using this block size. We perform the last epoch of the training in HBFP6 and the rest in HBFP4 for all the experimental settings. We also trained by keeping the last 10 epochs in HBFP6 to observe the improvement in accuracy for the CNN models. We keep all CNN models' first and last layers in HBFP6. The first and last layers of the CNN models account for a negligible amount of computation; thus, keeping them in slightly higher precision during HBFP training does not result in a significant increase in the hardware area or energy consumption. We can see that for most of the CNN models, Accuracy Boosters outperforms FP32. When we keep the last 10 epochs in HBFP6, we observe that the accuracies slightly increase (see Table II).

TABLE II: Top-1 validation accuracies of various CNN models for Accuracy Boosters

	Models and Datasets			
	CIFAR10		CIFAR100	
Epochs using HBFP6	ResNet20	ResNet74	ResNet74	DenseNet40
Only last	91.24	92.62	73.74	**73.61**
Last 10	91.36	93.02	74.28	**74.10**
FP32	91.72	93.57	74.55	72.42

Table III shows the results of applying Accuracy Boosters to the Transformer. We observe that for the Transformer, HBFP6 performs similarly to FP32. While standalone HBFP4 does not incur a significant accuracy loss, Accuracy Boosters still help further bridge the gap to FP32 and even outperform it.

TABLE III: BLEU Scores for Transformer-Base trained on IWSLT'14 De$\rightarrow$En task for various encodings

	FP32	HBFP6	HBFP4	Booster
BLEU Score	34.77	34.47	32.64	36.08

We observe that mixed-mantissa training using Accuracy Boosters can be carried out on arithmetic units designed for HBFP4. The small fraction of total training operations that use HBFP6 can be bit-sliced to fit on the 4-bit arithmetic units, similar to techniques proposed in prior work [38], while maintaining the same throughput. Thus, we claim the arithmetic density of a hardware accelerator using Accuracy Boosters will be approximately equal to the arithmetic density of HBFP4.

In conclusion, Accuracy Boosters offers up to $21.3\times$ higher arithmetic density compared to FP32 by using only 4 bits for 99.7% of total training computations while achieving comparable or better accuracy. Our analytic model estimates another state-of-the-art reduced precision format—BFloat16 only offers $4.9\times$ higher arithmetic density compared to FP32. Hence, the much superior arithmetic density of HBFP4 enables Accuracy Boosters to offer a further $4.4\times$ higher arithmetic density compared to BFloat16 . Apart from arithmetic density, 4-bit mantissas promise significant memory savings, but the exact value depends on the layout and scheme and is outside the scope of this work.

V. Related Work

In recent years, there has been a significant amount of research on inference and training [4], [5], [8], [17], [21], [23], [29], [39] with narrow numerical representations. Google Brain's bfloat16 [35], NVIDIA's mixed-precision training with FP16 [25], and another mixed-precision scheme using FP8 [31] are the most commonly-used ones. Recent research advocates the use of Block Floating-Point for DNN training [11] and inference [6]. Flexpoint [20] and Dynamic Fixed-Point [7] propose block-floating-point formats for training with a 16-bit mantissa and a shared exponent. Prior work proposed a novel format for training DNNs with BFP, called Hybrid Block Floating-Point (HBFP) [10]. In this paper, we argue that reducing the mantissa bit width in HBFP significantly improves silicon efficiency while designing hardware for DNN training.

Many have proposed techniques to compensate for the data loss introduced by narrower numerical representations [12], [24], [31], [32]. Mixed-precision training has emerged as a popular technique to recover the information loss caused by quantization. Several techniques vary the precision layer-wise by using higher precision arithmetic for layers with greater significance [18], [30], [37]. Specifically, [3], [24], [34], [39] use FP32 for the first and last layers. [13] employ fixed-point arithmetic with different bit widths epoch-wise over the course of training. Combining the layer-wise and epoch-wise approaches, [14], [27], [38] vary the precision adaptively per epoch and layer at the same time using control mechanisms. While all the aforementioned studies employ leaner arithmetic for a fraction of the training process, they fail to make leaner arithmetic the common case of the training process.

Recent work [9] suggests that during mixed-precision FP16 training, the optimizer states can be reduced to 8 bits by using a block-wise quantization method. This observation is in line with our work that applies quantization by extracting the largest exponent per block. Similarly, FAST [38] uses a block-floating-point-based layer-wise mixed precision approach using 2 and 4-bit mantissa. Unlike our work, FAST requires fine-tuning several additional hyperparameters for its training algorithm, making it difficult to apply to other DNN models. Another block-floating-point-based work, FlexBlock [27], uses 4 and 8-bit mantissa with various block sizes and also needs higher-precision block-floating-point formats only for weight gradient calculations that suffer more from quantization errors.

VI. Conclusion

Several low-precision training techniques and specialized numerical formats have been introduced over the past decade to increase the arithmetic density of the DNN accelerators. One such format, Hybrid Block Floating-Point (HBFP), which allows for a majority of the DNN's arithmetic operations (i.e., dot products) to be performed using fixed-point arithmetic has been shown to achieve FP32 accuracy with 8-bit mantissa. However, a smaller number of mantissa bits allow for exceptional improvements in arithmetic density. In this paper, we perform a full-scale exploration of the HBFP design space for emerging models and datasets. We show that HBFP6 is the smallest HBFP format achieving FP32 accuracy for all block sizes. We propose the Accuracy Boosters technique to bring HBFP4 into training, using HBFP6 in the last epoch, leveraging the insight that each epoch has a different effect on training. We show that the last stage of training requires more precision than the rest. We use an analytic model to show that our method achieves up to $21.3\times$ higher arithmetic density over FP32 and $4.4\times$ higher density over BFloat16 , while maintaining or outperforming FP32 accuracy.

Acknowledgements

The authors thank the anonymous reviewers and the members of PARSA at EPFL for their precious comments and feedback. We would also like to thank Nicholas Sperry for his contributions to the loss landscape experiments. This work has been partially funded by a Microsoft PhD Fellowship, and the following grant: "Unified Accelerators for Post-Moore Machine Learning" from the Swiss National Science Foundation (SNSF).

References

[1] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *CoRR*, vol. abs/1701.07875, 2017. [Online]. Available: http://arxiv.org/abs/1701.07875

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[3] J. Choi, Z. Wang, S. Venkataramani, P. I. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: parameterized clipping activation for quantized neural networks," *CoRR*, vol. abs/1805.06085, 2018. [Online]. Available: http://arxiv.org/abs/1805.06085

[4] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 3123–3131. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/3e15cc11f979ed25912dff5b0669f2cd-Abstract.html

[5] M. Courbariaux, Y. Bengio, and J. David, "Low precision arithmetic for deep learning," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.7024

[6] B. Darvish Rouhani, D. Lo, R. Zhao, M. Liu, J. Fowers, K. Ovtcharov, A. Vinogradsky, S. Massengill, L. Yang, R. Bittner, A. Forin, H. Zhu, T. Na, P. Patel, S. Che, L. Chand Koppaka, X. SONG, S. Som, K. Das, S. T, S. Reinhardt, S. Lanka, E. Chung, and D. Burger, "Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 10 271–10 281. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/747e32ab0fea7fbd2ad9ec03daa3f840-Paper.pdf

[7] D. Das, N. Mellempudi, D. Mudigere, D. D. Kalamkar, S. Avancha, K. Banerjee, S. Sridharan, K. Vaidyanathan, B. Kaul, E. Georganas, A. Heinecke, P. Dubey, J. Corbal, N. Shustrov, R. Dubtsov, E. Fomenko, and V. O. Pirogov, "Mixed precision training of convolutional neural networks using integer operations," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=H135uzZ0-

[8] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," *CoRR*, vol. abs/2208.07339, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.07339

[9] T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer, "8-bit optimizers via block-wise quantization," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=shpkpVXzo3h

[10] M. Drumond, T. Lin, M. Jaggi, and B. Falsafi, "Training DNNs with Hybrid Block Floating Point," *arXiv:1804.01526 [cs, stat]*, Dec. 2018, arXiv: 1804.01526. [Online]. Available: http://arxiv.org/abs/1804.01526

[11] M. P. Drumond, "Coltrain: Co-located dnn training and inference," p. 115, 2020. [Online]. Available: http://infoscience.epfl.ch/record/280118

[12] S. Fox, S. Rasoulinezhad, J. Faraone, D. Boland, and P. H. W. Leong, "A block minifloat representation for training deep neural networks," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=6zaTwpNSsQ2

[13] Y. Fu, H. Guo, M. Li, X. Yang, Y. Ding, V. Chandra, and Y. Lin, "CPT: efficient deep neural network training via cyclic precision," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=87ZwsaQNHPZ

[14] Y. Fu, H. You, Y. Zhao, Y. Wang, C. Li, K. Gopalakrishnan, Z. Wang, and Y. Lin, "Fractrain: Fractionally squeezing bit savings both temporally and spatially for efficient DNN training," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/8dc5983b8c4ef1d8fcd5f325f9a65511-Abstract.html

[15] 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, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[16] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 4107–4115. [Online]. Available: https://proceedings.neurips.cc/paper/2016/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html

[18] S. Khoram and J. Li, "Adaptive quantization of neural networks," p. 13, 2018.

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009.

[20] U. Köster, T. Webb, X. Wang, M. Nassar, A. K. Bansal, W. Constable, O. Elibol, S. Gray, S. Hall, L. Hornof, A. Khosrowshahi, C. Kloss, R. J. Pai, and N. Rao, "Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://papers.nips.cc/paper/2017/hash/a0160709701140704575d499c997b6ca-Abstract.html

[21] F. Li and B. Liu, "Ternary weight networks," *CoRR*, vol. abs/1605.04711, 2016. [Online]. Available: http://arxiv.org/abs/1605.04711

[22] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 6391–6401. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/a41b3bb3e6b050b6c9067c67f663b915-Abstract.html

[23] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, ser. JMLR Workshop and Conference Proceedings, M. Balcan and K. Q. Weinberger, Eds., vol. 48. JMLR.org, 2016, pp. 2849–2858. [Online]. Available: http://proceedings.mlr.press/v48/linb16.html

[24] N. Mellempudi, S. Srinivasan, D. Das, and B. Kaul, "Mixed Precision Training With 8-bit Floating Point," *arXiv:1905.12334 [cs, stat]*, May 2019, arXiv: 1905.12334. [Online]. Available: http://arxiv.org/abs/1905.12334

[25] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed Precision Training," *arXiv:1710.03740 [cs, stat]*, Feb. 2018, arXiv: 1710.03740. [Online]. Available: http://arxiv.org/abs/1710.03740

[26] S. Migacz, "8-bit Inference with TensorRT," May 2017. [Online]. Available: https://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf

[27] S.-H. Noh, J. Koo, S. Lee, J. Park, and J. Kung, "Flexblock: A flexible dnn training accelerator with multi-mode block floating point support," 2022. [Online]. Available: https://arxiv.org/abs/2203.06673

[28] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. C. Courville, "On the spectral bias of neural networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach,*

California, USA, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 5301–5310. [Online]. Available: http://proceedings.mlr.press/v97/rahaman19a.html

[29] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, ser. Lecture Notes in Computer Science, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9908. Springer, 2016, pp. 525–542. [Online]. Available: https://doi.org/10.1007/978-3-319-46493-0_32

[30] J. Shen, Y. Wang, P. Xu, Y. Fu, Z. Wang, and Y. Lin, "Fractional skipping: Towards finer-grained dynamic CNN inference," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 5700–5708. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6025

[31] X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani, V. V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan, "Hybrid 8-bit Floating Point (HFP8) Training and Inference for Deep Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/65fc9fb4897a89789352e211ca2d398f-Abstract.html

[32] X. Sun, N. Wang, C.-Y. Chen, J. Ni, A. Agrawal, X. Cui, S. Venkataramani, K. El Maghraoui, V. V. Srinivasan, and K. Gopalakrishnan, "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., 2020, pp. 1796–1807. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/13b919438259814cd5be8cb45877d577-Paper.pdf

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[34] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 7686–7695.

[35] S. Wang and P. Kanwar, "BFloat16: The secret to high performance on Cloud TPUs," Aug. 2019.

[36] Z. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma, "Frequency principle: Fourier analysis sheds light on deep neural networks," *CoRR*, vol. abs/1901.06523, 2019. [Online]. Available: http://arxiv.org/abs/1901.06523

[37] L. Yang and Q. Jin, "Fracbits: Mixed precision quantization via fractional bit-widths," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 10 612–10 620. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17269

[38] S. Q. Zhang, B. McDanel, and H. T. Kung, "FAST: DNN Training Under Variable Precision Block Floating Point with Stochastic Rounding," *arXiv:2110.15456 [cs]*, Oct. 2021, arXiv: 2110.15456. [Online]. Available: http://arxiv.org/abs/2110.15456

[39] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients," *arXiv:1606.06160 [cs]*, Feb. 2018, arXiv: 1606.06160.