BLOCK AND SUBWORD-SCALING FLOATING-POINT (BSFP) : AN EFFICIENT NON-UNIFORM QUANTIZATION FOR LOW PRECISION INFERENCE

Yun-Chen Lo, Tse-Kuang Lee, Ren-Shuo Liu
Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan

ABSTRACT

In this paper, we propose **B**lock and **S**ubword-Scaling **F**loating-**P**oint (**BSFP**), a datatype with a non-uniform quantization scheme for the skewed and non-uniform distribution of weight vectors in neural networks. By quantizing each weight vector as the superposition of multiple subword vectors (in two's complement) with scaling factors (in Low-bit Floating-Point, LBFP), BSFP can effectively fit the distribution of weight vectors while maintaining high computation efficiency. Furthermore, we present a grid search-based MSE-optimal quantization flow and a scaled serial processing engine to complete the quantization pipeline and the infrastructure.

The experimental results on the ImageNet classification task show that our proposed method outperforms state-of-the-art Microsoft Floating Point (MSFP) by up to 18.57% top-1 accuracy at the same weight precision and reduces up to 10.3% model size. Furthermore, BSFP outperforms MSFP by up to 2.0× computing throughput and up to 5.3× energy efficiency under the same silicon area budget.

1 INTRODUCTION

Deep Neural Networks (DNNs) have continuously enabled more and more eye-catching artificial intelligence (AI) applications Johnson et al. (2016); Lin et al. (2014); Deng et al. (2009). However, their large model size and high computational complexity hinder the wide deployment of DNNs to latency-sensitive cloud services and energy-constrained edge devices. To address the performance and energy challenges, in addition to compacting neural network structures Sandler et al. (2018); Ma et al. (2018), reducing the bitwidths of weights or activations also have been extensively explored Jacob et al. (2018); Darvish Rouhani et al. (2020); Tambe et al. (2020); Li et al. (2020).

Particularly, non-conventional datatypes and custom hardware are emerging to optimize the performance, energy efficiency, area efficiency, and memory requirements of DNN inference. Prior industry and academia researches have explored low-bit floating-point datatypes Kalamkar et al. (2019); Jouppi et al. (2020); NVIDIA (2022); Tambe et al. (2020), block-based floating-point datatypes Darvish Rouhani et al. (2020); Köster et al. (2017), low-bit fixed-point datatypes NVIDIA (2020); Jacob et al. (2018), and power-of-two fixed-point datatypes Miyashita et al. (2016); Zhou et al. (2017); Li et al. (2020) as the potential candidates in efficient DNN inference. Among many datatypes, Microsoft Floating Point (MSFP), a kind of block-based floating-point type as shown in Figure 1(b), claims to achieve the state-of-the-art tradeoff among dynamic range, DNN accuracy, and hardware complexity Darvish Rouhani et al. (2020).

This work focuses on post-training quantization, which is preferable in practice. First, for end users, it involves no data (including private data) and enables a low-friction deployment pipeline Nagel et al. (2019). Second, according to our discussions with an IC design house that tapes out AI chips in advanced technology nodes, the industry (at least their application-side customers) does appreciate post-training quantization because in most cases, AI application companies are reluctant to release AI models and training data to AI accelerator companies. Although we focus on post-training quantization, we still include the finetuning results in Appendix A.

This paper proposes **Block and Subword-Scaling Floating-Point (BSFP)**, a new class of dataypes with an bit-efficient, non-uniform quantization method and custom hardware to improve the energy

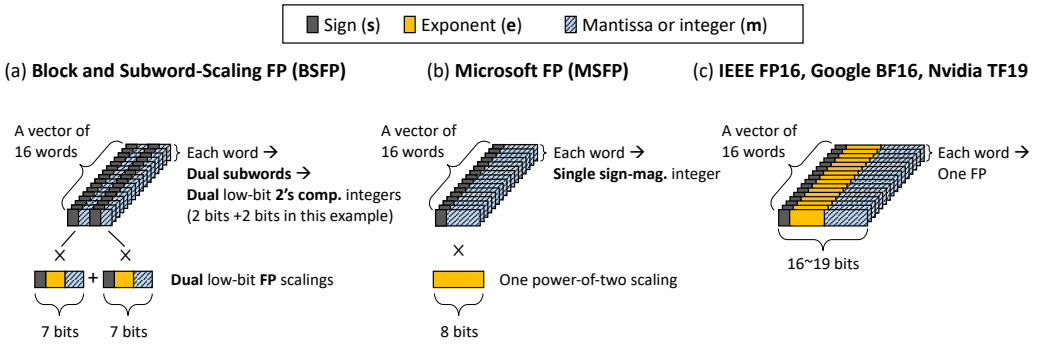


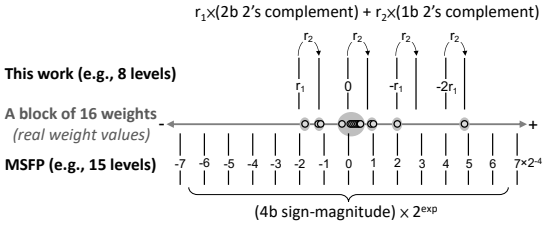
Figure 1: Number system comparison between (a) the proposed Block and Subword-Scaling Floating-Point (BSFP), (b) Microsoft FP (MSFP Darvish Rouhani et al. (2020)), and (c) floating-point numbers (IEEE 754 FP16, Google BF16 Jouppi et al. (2020), and Nvidia TensorFloat (TF19) NVIDIA (2022)).

efficiency and performance over state-of-the-art MSFP. As shown in Figure 1(a), the key idea of BSFP is to *approximate each full-precision weight vector using the sum of two subword vectors with two scalings, respectively. More specifically, each subword is a low-bit (e.g., 2-bit), signed (two's complement) integer, and each scaling is a low-bit floating-point (LBFP) number (e.g., a 7-bit one).* We will show that BSFP is superior to MSFP in capturing the nonuniformity and skewness of per-vector weight distributions, which are common cases for a vector made of a small number (e.g., 16) of weights. In addition, although BSFP adopts two scalings and two subword vectors, it can still be efficiently computed for the following three reasons. First, the computation cost of scaling is amortized over 16 weights. Second, each scaling is an LBFP and involves only low-bit operations, e.g., multiplications with a 3-bit mantissa. Third, the subword vector structure happens to fit bit-serial computation architectures Qian Zhang et al. (2022); Judd et al. (2016).

One property that BSFP exhibits is to approximate the desired weight vector using both coarse and fine vectors. One subword vector with a large scaling captures large weights, and the other subword vector with a small scaling mitigates the remaining deviations. Therefore, BSFP can adapt to large outliers and small resolutions simultaneously.

Figure 2(a) compares the quantization results of a real 16-element weight vector from ShuffleNet-v2 in either 8-level BSFP or 15-level MSFP. This example clearly demonstrates the potential that even the BSFP with relatively fewer quantization levels can achieve smaller quantization errors (e.g., in terms of MSE) than MSFP with more quantization levels. We summarize the rationales of BSFP's superiority below:

- **No waste of quantization level**: BSFP utilizes two's complement for each subword and does not waste precious quantization levels. On the other hand, MSFP resembles sign-magnitude and wastes one quantization level (i.e., duplicated $+0$ and -0). Even worse, the impact of wasting quantization levels increases as the bitwidth goes down. For instance, a 3-b two's complement number can represent 8 quantization levels, 12.5% more than the 7 levels of the 3-b sign-magnitude number.
- **Adaptation to skewed distribution**: BSFP exploits the asymmetrical nature of two's complement numbers (e.g., -2, -1, 0, 1 for 2-b two's complement numbers) and the sign of the associated scaling to adapt to the asymmetrical weight distribution of in weight vectors. In comparison, MSFP is permanently restricted to symmetrical quantization levels and leads to a waste of quantization levels fitting asymmetrical distributions.
- **Adaptation to non-uniform distribution**: BSFP can offer non-uniform quantization levels by combining two subword-scaling vectors. In comparison, MSFP always quantizes weight vectors, which may exhibit non-uniform weight distributions, uniformly.
- **Better freedom of quantization step size**: The quantization step size of BSFP is defined by the two scalings, which are (low-bitwidth) floating-point values. In contrast, the quantization step size of MSFP cannot be any value other than power-of-two, e.g., 0.5, 0.25, 0.125.



Properties	This Work	MSFP
No Wasted Quantization Level	**Yes**	1 Wasted Level (Because of Sign-Magnitude)
Can Adapt to Skewed Weight Distribution	**Yes**	No (Always Symmetrical)
Can Adapt to Non-Uniform Weight Distribution	**Yes**	No (Always Uniform)
Freedom of Quantization Step Size	**High**	Low (Must be the Power of Two)

Figure 2: (a) Quantizing **16 real weights of ShuffleNet-v2** using 3-b Block and Subword-Scaling Floating-Point (BSFP) can achieve both lower quantization error (MSE) and lower storage than 4-b Microsoft Floating Point (MSFP). (b) Properties comparison of BSFP with MSFP.

BSFP can be deployed to servers and edge devices, achieving high performance and energy efficiency. Please note that the main advantages of BSFP are not from specific MAC architecture. Therefore, we select one MAC architecture and focus on comparing the BSFP to MSFP datatype on that architecture (Same as MSFP v.s. BF16 and INT in their paper). Our experiments will demonstrate the robustness and generality of BSFP using several mainstream DNNs on the ImageNet classification task. In summary, this work makes the following contributions:

- We propose Block and Subword-Scaling Floating-Point (BSFP), a hardware-algorithm co-designed numerical datatype for DNNs that can achieve higher accuracy with better computing throughput and energy efficiency than MSFP.
- We design a scaled serial processing engine to support the proposed BSFP. This custom hardware enjoys a small area footprint and can support various configurations of the BSFP.
- We identify mean squared error (MSE) as an effective criterion in determining the LBFP scaling factors of BSFP and a grid search-based MSE-optimal quantization flow to enable a low-friction deployment pipeline. We present both post-training and fine-tuning procedures for preparing DNNs in the proposed BSFP format.
- We perform extensive evaluations on various DNNs of the ImageNet classification task and demonstrate that BSFP successfully outperforms MSFP in model size, quantization error, accuracy, throughput, and energy efficiency.

2 Related Works

There is rising attention to designing custom datatypes for efficient inferencing. The wide range of formats can be categorized into four classes, i.e., floating-point, fixed-point, power-of-two, and block-based floating-point.

The first category, floating-point numbers (FP32, FP16, Bfloat16 (BF16), and TensorFloat), adopts or modifies the IEEE-754 floating-point formats. This data format enjoys a large dynamic range at the cost of a significant hardware area. The representative commercial hardware are Google's TPUs (BF16) Jouppi et al. (2020) and NVIDIA's A100 GPUs (Tensorfloat, TF19) NVIDIA (2022).

The second class of datatype is low-bit fixed-point datatype (e.g., INT4), whose operations are equivalent to integer operations and can achieve low hardware cost and high performance. The fix-point numbers have evolved from 8-b Jacob et al. (2018), 4-b NVIDIA (2020); Dai et al. (2021), 3-b Mellempudi et al. (2017), and ultimately binary Hubara et al. (2016). Although fixed-point datatype achieves small hardware area and receives great popularity, it requires careful model re-calibration and suffers from large accuracy drop when representing values with a high dynamic range.

The third class is power-of-two numbers, including Power-of-Two Zhou et al. (2017); Miyashita et al. (2016) and Additive Power-of-Two Li et al. (2020), which utilizes one or two power-of-two terms to approximate floating-point weights. The power-of-two format is appealing for replacing multipliers with low-cost shifters. However, the available values are limited by the power-of-two form.

Additionally, some prior research also explores non-uniform quantization. For example, LQ-Nets Zhang et al. (2018) learns quantization levels to minimize the quantization error. Distillation Polino et al. (2018) optimizes the quantization levels by learning to minimize a task loss with

their teacher networks. However, these methods use a limited number of floating-point numbers to quantize the full-precision data points, leading to significantly larger computation overhead.

The last category is block-based floating-point, which forces a block of (e.g., 16) floating-point numbers to share one exponent. The block-based floating-point offers low area overheads and a large dynamic range. This format has enabled state-of-the-art accuracy-to-area Pareto frontier. The representative examples are Intel's Lake Crest (Flexpoint Köster et al. (2017)) and Microsoft's Brainwave Fowers et al. (2018) (MSFP Darvish Rouhani et al. (2020)).

The closest work related to this paper is MSFP in NeurIPS 2020 Darvish Rouhani et al. (2020), which is the current state-of-the-art format in terms of area-to-accuracy trade-offs. Our BSFP design is fundamentally different and novel compared to MSFP for the following reasons: 1) BSFP approximates full-precision weight vector using the superposition of multiple subword-scaling vectors, which MSFP and other prior works do not explore before. Please refer to the properties comparison in Figure 2 and the corresponding discussion of BSFP's superiority. 2) We design a bit-serial processing engine that can support different configurations of BSFP. In comparison, the MSFP work considers a bit-parallel processing engine and does not support format changes.

3 BLOCK AND SUBWORD-SCALING FLOATING-POINT (BSFP)

We propose BSFP that quantize an l-element full-precision weight vector ($\overrightarrow{W_{fp}}$) using N_{sub} l-element subword-scaling vectors ($\overrightarrow{W_{sub_i}} * scale_i = \overrightarrow{W_{sub_i}} * (-1)^{s_i} 2^{e_i} m_i$), where the subwords adopt the two's complement format and the scaling factors adopt the LBFP format. The default l is 16 unless explicitly mentioned. These scaling factors $(-1)^{s_i} 2^{e_i} m_i$ determine the quantization points that a BSFP vector can represent. We find that using two subwords achieves good accuracy-storage trade-offs and set $N_{sub} = 2$.

$$\overrightarrow{W_{fp}} \approx \sum_{i=1}^{N_{sub}} \overrightarrow{W_{sub_i}} * (-1)^{s_i} 2^{e_i} m_i \xlongequal{N_{sub}=2} (\overrightarrow{W_{sub_1}} * (-1)^{s_1} 2^{e_1} m_1) + (\overrightarrow{W_{sub_2}} * (-1)^{s_2} 2^{e_2} m_2)$$

We apply BSFP to weights, and we leave activations as MSFP as the MSFP paper does Darvish Rouhani et al. (2020). The rationales behind this decision are as follows. Weights differ from activations in that weights are available offline. Thus, weight can and also should enjoy a longer quantization time budget. Clearly, if both weights and activations adopt MSFP, as the MSFP paper does, it directly suggests that some optimization opportunities for weights are left on the table.

Computing with the BSFP format. Dot products are the fundamental operations in DNNs. Here we show the dot product of a l-element $\overrightarrow{W_{bsfp}}$ weight vector and a $\overrightarrow{A_{msfp}}$ activation vector.

$$\begin{aligned}\overrightarrow{W_{bsfp}} \cdot \overrightarrow{A_{msfp}} &= (\sum_{i=1}^{N_{sub}} \overrightarrow{W_{sub_i}} * (-1)^{s_i} 2^{e_i} m_i) \cdot (\overrightarrow{A_{msfp}} * 2^{e_{shared}}) \\ &= \sum_{i=1}^{N_{sub}} ((\overrightarrow{W_{sub_i}} \cdot \overrightarrow{A_{msfp}}) * ((-1)^{s_i} m_i 2^{e_i + e_{shared}}))\end{aligned}$$

For each subword vector of the BSFP, it first multiplies with the MSFP activation vector using compact multipliers and an adder tree to obtain the partial sum. Secondly, the LBFP scaling factor of BSFP is combined with the shared exponent of MSFP to get the correct scaling factor. More specifically, the exponent fields of BSFP and MSFP are summed together. Finally, we scale the partial sum with the combined scaling factor. The hardware overheads of BSFP over MSFP is to multiply the partial sum by a low-bit (3-b or 4-b) mantissa of the scaling factor.

Scaled Serial Processing Engine (S^2PE). Figure 3 shows the proposed scaled serial processing engine (S^2PE) and its example in computing a 4b BSFP weight vector. Within a S^2PE, the 2b multipliers are chosen to enable flexibility in computing different bitwidth configurations with bit-serial computation Qian Zhang et al. (2022). This design choice achieves a compact silicon footprint and high computation flexibility.

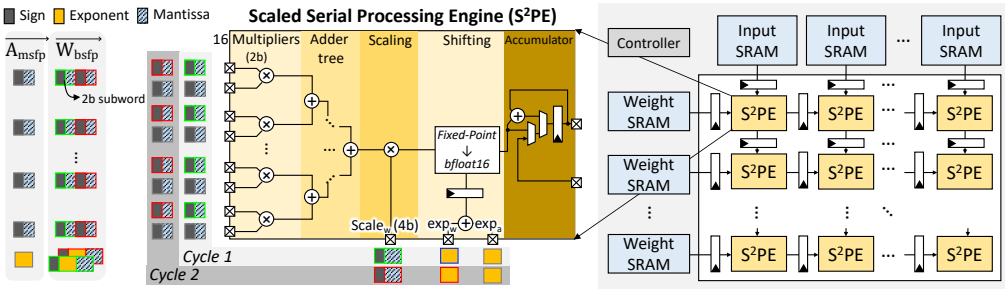


Figure 3: Proposed scaled serial processing engine (S^2PE) and systolic architecture.

In addition, the proposed PE only requires an integer multiplier to scale the partial sum by the (at most) 4-b mantissa of the LBFP scaling factor. The exponent terms of BSFP and MSFP are added together to convert the scaled partial sum back to BF16 for accumulation. For BSFP with a bitwidth larger than 2, the proposed PE computes the result using multiple cycles. We finally present a systolic architecture, which integrated multiple scaled serial processing engines.

Criterion-optimal quantization flow. The flexibility of the BSFP requires a flow to determine a suitable LBFP scaling factors setup. We observe that the number of combinations for LBFP scaling factors, which exhaustively evaluates all of them on GPU is feasible. Specifically, every weight vectors are parallelly quantized and evaluated using the same LBFP scaling setup. For example, an optimal setup for ShuffleNet-v2 can be found within 30 minutes using one NVIDIA V100 (ResNet-50: less than 1 hour; ViT: less than 1.5 hours).

Figure 4 shows the quantization flow, which consists of four steps: 1) *Generating subword configurations*, which generate several potential subword configurations that satisfy the target weight bitwidth. 2) *Generating LBFP scaling combinations*, which creates search space by exploring all potential combinations (grids) of LBFP scaling factors. 3) *Iterative rounding*, which quantizes the full-precision weight vectors using generated scaling factor combinations. 4) *Criterion evaluation*, which evaluates the difference between the full-precision weights and quantized counterparts using chosen criterion. The criterion candidates are L1, Mean Square Error (MSE), and Cosine Similarity, where MSE is selected as final criterion. The details of the flow is presented in Appendix D.

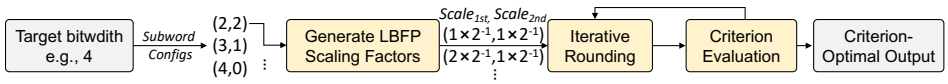


Figure 4: Criterion-optimal quantization flow for BSFP format.

4 BSFP Configurations

In this section, we discuss the effects of different configuration setups for the BSFP.

Block Size (or vector length, l). The block size determines the model size, the PE area efficiency, and the model accuracy. In general, a larger vector length amortizes the hardware overheads and reduces the storage overheads of LBFP scaling factors. In practice, we found a vector length of 16-64 to be effective for BSFP in preserving the accuracy while incurring a moderate hardware cost.

Encoding Criterion. The quantization criterion plays an important role in our quantization flow to decide the final accuracy. Common criterion candidates are Manhattan distance (L1 distance), Mean square error (MSE, Euclidean distance, L2 distance) Zhao et al. (2019), and Cosine Similarity Zhu et al. (2019); Zhang et al. (2019). We identify MSE as a better criterion in our quantization flow. Furthermore, users can easily define their evaluation criterion in the framework.

Number of Subwords. We can tune the number of subwords (N_{sub}) factor to explore the trade-offs for the BSFP datatype. To fully obtain the benefit of BSFP, we suggest setting N_{sub} larger than 1, e.g., 2-4. In practice, the number of subwords is set to 2 to balance storage overheads and accuracy.

Low-bit Floating-point (LBFP) Scaling Factor Format. The configuration for LBFP scaling factor format is also crucial in determining our quantization flow's accuracy and search space. Appendix H shows that scaling format with 1s4m3e and 1s3m3e achieves good trade-offs.

Besides, we propose to equip separate exponent biases for different LBFP scaling factors. The key idea is to set suitable biases to minimize the LBFP scaling factor's precision while achieving satisfying accuracy. We empirically set exponent biases of first and second subword scaling factors to be -3 and -8, respectively. It is noteworthy that both biases are the same across the whole model, which incurs negligible storage overheads. Setting biases on a per-layer basis is left as future work.

5 EVALUATIONS

Baselines and Network Architectures. We compare the BSFP with MSFP Darvish Rouhani et al. (2020), DSQ Nagel et al. (2019), TFlite Krishnamoorthi (2018), and APoT Li et al. (2020) in accuracy-to-precision trade-off. Furthermore, we also compare area and power efficiency with other strong baselines, such as BF16 Jouppi et al. (2020), fixed-point Nagel et al. (2019) and Power-of-Two PEs Zhou et al. (2017); Li et al. (2020).

We select six mainstream DNNs for evaluating BSFP and MSFP. ShuffleNet-v2 Ma et al. (2018) and MobileNet-v2 Sandler et al. (2018) are chosen to represent compact DNNs. ResNet-18 and ResNet-50 He et al. (2015) are selected to represent classical DNNs. EfficientNet-v2 Tan & Le (2019) and VIT Dosovitskiy et al. (2020) are chosen to represent modern DNNs. All of the results are based on ImageNet classification dataset Deng et al. (2009).

Hardware Evaluations. We implement the BSFP PE and baseline PEs in Verilog and validate the behavior against the software functional simulator. All designs are synthesized at 500 MHz under TSMC $40nm$ using Synopsys Design Compiler (Topographical mode). We set the parallelism to be 16 for all number systems.

5.1 MODEL SIZE-TO-ACCURACY AND VECTOR LENGTH-TO-MODEL SIZE TRADE-OFFS

Figure 5(a) shows the storage-to-accuracy Pareto frontier of BSFP and MSFP on ShuffleNet-v2 (post-training quantization). The BSFP consistently improves the accuracy using a smaller model size. For example, BSFP simultaneously obtains 0.4% top-1 accuracy gain and saves 5% model size over MSFP. The superiority of BSFP is consistent across four evaluated models, where other results are omitted for brevity.

Figure 5(b) compares the accuracy-to-storage trade-offs of different vector lengths for BSFP and MSFP on ShuffleNet-v2 (post-training quantization). On BSFP, the vector length ranging from 16 to 64 leads to only a marginal accuracy drop, while a vector length of 128 significantly decreases the accuracy. Thus, we suggest that the vector length for BSFP be 16-64. On the other hand, MSFP drops accuracy significantly from a length of 16 to 32, and the accuracy does not vary significantly while further enlarging the vector length. It is also noteworthy that BSFP consistently outperforms the Pareto frontier regardless of vector length compared to MSFP.

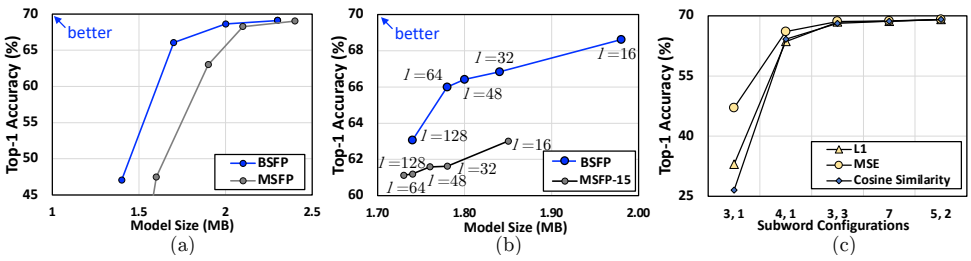


Figure 5: (a) Model size-to-accuracy Pareto frontiers of BSFP and MSFP with various bitwidths (fixing vector length). (b) Model size-to-accuracy Pareto frontiers of BSFP and MSFP with various vector length, l (fixing bitwidth). (c) Quantization using different criteria. All experiments are conducted on ShuffleNet-v2 using post-training quantization.

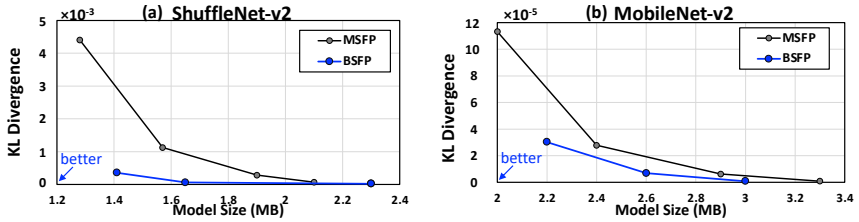


Figure 6: KL Divergence-to-model size comparison of MSFP and BSFP on (a) ShuffleNet-v2 and (b) MobileNet-v2. The KLD of BSFP is consistently lower than MSFP while using a smaller model size.

5.2 Quantizing using Different Criterion

Figure 5(c) shows the accuracy of quantizing ShuffleNet-v2 using different criteria, that is, L1 distance, MSE, and Cosine Similarity. The MSE continuously serves as the best criterion in our grid search-based quantization flow. It is noteworthy that different criteria have little accuracy difference in higher bitwidth and vary considerably in lower bitwidths. One reason of MSE's superiority is that it punishes large element-wise distortion, which other two criteria cannot. In comparison, selecting L1 norm optimizes overall distortion and selecting Cosine Similarity optimizes the angular difference.

5.3 Kullback-Leibler Divergence Analysis

Kullback-Leibler Divergence (KL Divergence) is chosen for evaluating quantization quality in MSFP paper. Intuitively, lower KL Divergence demonstrates the quantization can fit the original data much better, which results in higher accuracy. We compare the KL Divergence of MSFP and BSFP to demonstrate the superiority of BSFP.

Figure 6(a) and Figure 6(b) sample a layer from ShuffleNet-v2 and MobileNet-v2 to compare the KL Divergence-to-model size Pareto frontier of BSFP and MSFP. The key takeaway is that BSFP consistently obtains better KL Divergence for both models while allocating smaller model sizes. The discrepancy between MSFP and BSFP enlarges as the model size reduces.

5.4 Per-vector Absolute Pearson's Skewness Analysis

Figure 7 analyzes the per-vector absolute Pearson's skewness coefficient (SK) for each layer of ShuffleNet-v2 and ResNet-18. We report the proportion of vectors with different degrees of skewness, i.e., moderate ($0.5 < SK < 1.0$), high ($1.0 < SK < 1.5$), and very high ($1.5 < SK$). Two takeaways are: 1) The weight vectors on modern networks, i.e., ShuffleNet-v2 and ResNet-18, are skewed. Specifically, up to $50\% \sim 75\%$ of the weight vectors are skewed. The proposed BSFP can adapt to skewed distribution, which can outperform prior works with uniform number systems. 2) Even on weight vectors that are less skewed, the BSFP can still outperform prior works because it does not waste quantization levels and provides better step size flexibility.

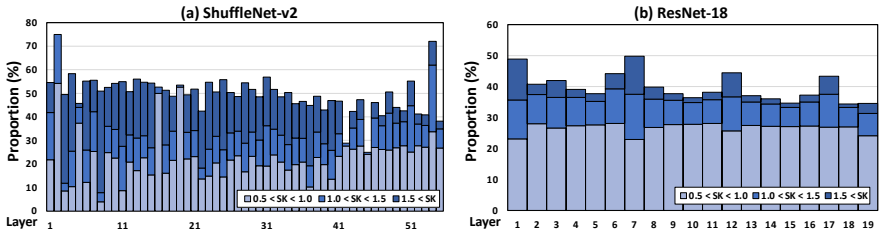


Figure 7: Per-vector absolute Pearson's skewness coefficient (SK) of (a) ShuffleNet-v2 and (b) ResNet-18. In general, very highly skewed means $SK > 1.5$, highly skewed means $SK > 1.0$, and moderately skewed means $1.0 > SK > 0.5$. Appendix G shows that slicing weight into vectors is one potential rationale for the large skewness.

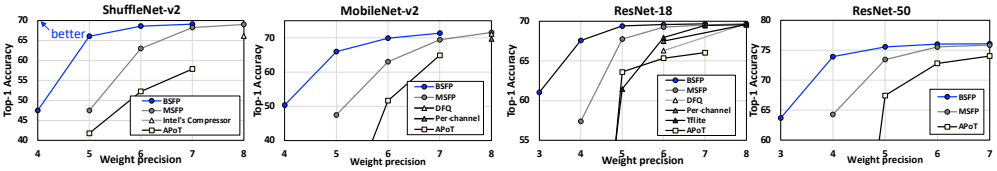


Figure 8: Precision-to-Accuracy comparison of BSFP with MSFP, DFQ, TFlite, APoT. All number systems utilize post-training quantization. The accuracy of DFQ Nagel et al. (2019) and TFlite Krishnamoorthi (2018) are derived from the papers. We implement APoT to obtain the accuracy.

Table 1: Post-training comparison of accuracy and hardware performance on ImageNet with MSFP and BSFP. We normalize sub-8b ops to 8b ones, where a 4 by 6-bit multiplication is equivalent to ${}^{4\times6}/_{8\times8} = {}^{3}/_{8}$ FixOP (8b).

	Method	Precision (W / A)	Top-1 (%)	Model Size	FLOPs/ FixOPs (8b)	Method	Precision (W / A)	Top-1 (%)	Model Size	FLOPs/ FixOPs (8b)
ShuffleNet-v2	FP	32 / 32	69.18	9.1 MB	148.8M					
	MSFP	8 / 8	69.03	2.4 MB	148.8M	BSFP (OURS)	7 [5+2] / 8	**69.10**	**2.3 MB**	**130.2M**
		7 / 7	68.27	2.1 MB	113.9M		6 [3+3] / 6	**68.62**	**2.0 MB**	**83.7M**
		6 / 6	63.01	1.9 MB	83.7M		5 [4+1] / 4	**66.06**	**1.7 MB**	**46.5M**
		5 / 5	47.49	1.6 MB	58.1M		4 [3+1] / 4	**47.50**	**1.4 MB**	**37.2M**
MobileNet-v2	FP	32 / 32	71.84	14.0 MB	314.1M					
	MSFP	8 / 8	71.58	3.7 MB	314.1M	BSFP (OURS)	**7** [4+3] / 8	71.37	**3.5 MB**	**274.8M**
		7 / 7	69.42	3.3 MB	240.5M		**6** [4+2] / **6**	**69.93**	**3.0 MB**	**176.7M**
		6 / 6	63.03	2.9 MB	176.7M		**5** [4+1] / **4**	**66.01**	**2.6 MB**	**98.2M**
		5 / 5	47.49	2.4 MB	122.7M		**4** [2+2] / **4**	**50.35**	**2.2 MB**	**78.5M**
ResNet-18	FP	32/32	69.76	46.8 MB	1.82G					
	MSFP	8 / 8	69.69	12.4 MB	1.82G	BSFP (OURS)	**7** [5+2] / 8	69.67	**11.6 MB**	**1.59G**
		7 / 7	69.54	11.0 MB	1.39G		**6** [4+2] / **6**	**69.58**	**10.1 MB**	**1.02G**
		6 / 6	69.27	9.5 MB	1.02G		**5** [3+2] / **4**	**69.40**	**8.7 MB**	**568.8M**
		5 / 5	67.75	8.0 MB	710.9M		**4** [2+2] / **4**	67.57	**7.2 MB**	**455.0M**
		4 / 4	57.40	6.6 MB	455.0M		**3** [2+1] / **3**	**61.04**	**5.8 MB**	**255.9M**
ResNet-50	FP	32/32	76.13	102.2 MB	4.14G					
	MSFP	8 / 8	76.06	27.2 MB	4.14G	BSFP (OURS)	**7** [5+2] / 8	**76.08**	**25.4 MB**	**3.6G**
		7 / 7	75.86	24.0 MB	3.17G		**6** [4+2] / **6**	**76.02**	**22.2 MB**	**2.3G**
		6 / 6	75.54	20.8 MB	2.3G		**5** [3+2] / **4**	**75.57**	**19.0 MB**	**1.6G**
		5 / 5	73.45	17.6 MB	1.6G		**4** [3+1] / **4**	**73.93**	**15.8 MB**	**1.0G**
		4 / 4	64.28	14.4 MB	1.0G		**3** [2+1] / **3**	63.68	**12.6 MB**	**582.2M**
EfficientNet-v2 (s)	FP	32/32	84.23	88.0 MB	8.8G					
	MSFP	8 / 8	84.09	23.4 MB	8.8G	BSFP (OURS)	**7** [5+2] / 8	84.06	**21.8 MB**	**7.7G**
		7 / 7	83.94	20.6 MB	6.7G		**6** [4+2] / **6**	**83.96**	**19.1 MB**	**5.0G**
		6 / 6	83.08	17.9 MB	5.0G		**5** [4+1] / 6	**83.12**	**16.3 MB**	**4.2G**
		5 / 5	77.45	15.1 MB	3.4G		**4** [2+2] / **4**	76.08	**13.6 MB**	**2.2G**
		4 / 4	3.27	12.4 MB	2.2G		**3** [2+1] / 4	**27.20**	**10.8 MB**	**1.7G**
Vision Transformer (ViT-B/16)	FP	32/32	81.07	348.1 MB	56.0G					
	MSFP	8 / 8	81.01	92.4 MB	56.0G	BSFP (OURS)	**7** [5+2] / 8	80.92	**86.3 MB**	**49.0G**
		7 / 7	80.94	81.6 MB	42.9G		**6** [4+2] / **6**	80.87	**75.4 MB**	**31.5G**
		6 / 6	80.84	70.7 MB	31.5G		**5** [4+1] / **4**	80.41	**64.6 MB**	**17.5G**
		5 / 5	80.13	59.8 MB	21.9G		**4** [2+2] / **4**	**80.21**	**53.7 MB**	**14.0G**
		4 / 4	76.89	48.9 MB	14.0G		**3** [2+1] / 4	**79.77**	**42.8 MB**	**10.5G**

5.5 Weight Precision-to-Accuracy Pareto Frontier

Figure 8 compares the weight precision-to-accuracy Pareto frontier of MSFP and BSFP on four DNNs. The BSFP consistently outperforms MSFP on the accuracy, given the same weight precision. Furthermore, the accuracy gap between MSFP and BSFP enlarges while reducing the weight precision. Let us take ShuffleNet-v2 as an example. When allocates 7b weight precision for both BSFP and MSFP, BSFP achieves 69.10% top-1 accuracy, outperforming MSFP's 68.27% accuracy by 0.83%. When further narrows down the weight bit to five, the accuracy benefit of BSFP over MSFP enlarges. Further, BSFP consistently outperforms MSFP and other numerical systems under same precision.

Table 1 summarizes the accuracy, model size, and the number of operations on six DNNs. For full precision models, we report their computation complexity in FLOPs. We define a FixOP as one operation between an 8-bit fixed-point weight and an 8-bit fixed-point activation, which takes 64 binary operations for quantized models. To sum up, BSFP consistently achieves higher accuracy with smaller model size and computation complexity than MSFP. Smaller FixOPs improve the computation steps on serial PE or reduce hardware overheads on parallel PE.

Table 2: BSFP versus other mainstream number formats for DNN inference. Memory and MAC density of various formats are normalized to BF16. The results listed are based on topographical synthesis results using TSMC $40nm$ process at 500 MHz.

	16× BF16	16× INT4	16× INT8	16× Power-of-Two	Bit-Serial MSFP PE	Scaled Serial PE (BSFP)	
						2b-mult	1b-mult
Area (per PE)	1.0×	22.6×	8.7×	7.8×	13.0×	**26.6×**	**33.1×**
Power (per PE)	1.0×	13.0×	5.1×	4.8×	8.5×	**14.1×**	**24.6×**

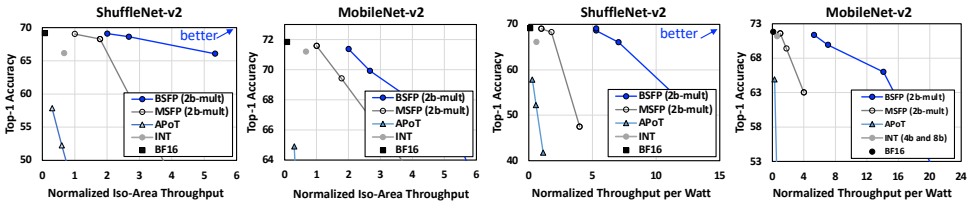


Figure 9: Iso-area Throughput to Accuracy (left) and Throughput per Watt to Accuracy (right) comparison of BSFP with other strong baselines. We normalize the throughput and throughput/W to MSFP-16.

5.6 Accuracy to Performance and Accuracy to Energy Efficiency Comparisons

Table 2 shows the area and power of PEs with different number formats. We normalize all of the designs with BF16 PE. In general, scaled serial PE achieves smallest area and power because of the compact MAC circuit. The 2-b scaled serial PE can outperform 2-b serial MSFP PE Qian Zhang et al. (2022) because of the following reasons:

- Larger adder tree induced by additional XOR gate: Besides area overheads of XOR gates, the bitwidth of the multiplication results is required to enlarge because of the potential negation. Consequently, the 16x4-b adder tree has to increase each input port by 1-b, causing 25% more complexity than the original adder tree. In brief, the XOR gates, the negation logic, and the enlarged adder tree jointly cause the area and energy in-efficiency.
- The alignment overheads for MSFP are more significant than for BSFP: Although the bitwidth of scaling factors are similar for BSFP and MSFP, MSFP allocates all 8-b to be the exponent. In contrast, BSFP allocates only 3-b to be exponent and lets the rest be sign and mantissa. As a result, the alignment procedure before accumulation is largely simplified for BSFP, which improves energy efficiency.

Figure 9 (left) shows the iso-area throughput to accuracy comparison. The BSFP consistently outperforms the MSFP across four models. More specifically, BSFP outperforms MSFP by 1.3×-2.0× throughput while achieving higher accuracy. The throughput improvement is a result of smaller PE area and lower weight precision. Figure 9 (right) shows the throughput per W to accuracy comparison. The BSFP also achieves significantly better throughput per Watt than MSFP. More specifically, BSFP outperforms MSFP by 1.3×-5.3× while achieving higher accuracy.

6 Conclusions

This paper introduces the Block and Subword-Scaling Floating-Point (BSFP) for quantizing weight vectors in neural networks, which typically exhibit a skewed and non-uniform distribution. The quantized vector of BSFP is the sum of a set of subword-scaling vectors, bringing up 2× speedup and 5.3× energy efficiency improvements compared with MSFP.

In addition, our grid search method guarantees to find optimal scaling setups for optimizing the criterion, which captures dense data points and outliers. Our proposed scaled serial PE can obtain optimal area efficiency and energy efficiency compared to prior number systems. As a result, BSFP successfully reaches state-of-the-art model size, throughput, and energy efficiency.

REFERENCES

Arm, 2020. URL https://developer.arm.com/Architectures/Arm%20Custom%20Instructions.

Yu-Hsin Chen, Tushar Krishna, Joel S. Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2017. doi: 10.1109/JSSC.2016.2616357.

Lucian Codrescu. Qualcomm hexagon dsp: An architecture optimized for mobile multimedia and communications. In *2013 IEEE Hot Chips 25 Symposium (HCS)*, pp. 1–23, 2013. doi: 10.1109/HOTCHIPS.2013.7478317.

Steve Dai, Rangha Venkatesan, Mark Ren, Brian Zimmer, William Dally, and Brucek Khailany. Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference. In A. Smola, A. Dimakis, and I. Stoica (eds.), *Proceedings of Machine Learning and Systems*, volume 3, pp. 873–884, 2021. URL https://proceedings.mlsys.org/paper/2021/file/f0935e4cd5920aa6c7c996a5ee53a70f-Paper.pdf.

Bita Darvish Rouhani, Daniel Lo, Ritchie Zhao, Ming Liu, Jeremy Fowers, Kalin Ovtcharov, Anna Vinogradsky, Sarah Massengill, Lita Yang, Ray Bittner, Alessandro Forin, Haishan Zhu, Taesik Na, Prerak Patel, Shuai Che, Lok Chand Koppaka, XIA SONG, Subhojit Som, Kaustav Das, Saurabh T, Steve Reinhardt, Sitaram Lanka, Eric Chung, and Doug Burger. Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 10271–10281. Curran Associates, Inc., 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, Stephen Heil, Prerak Patel, Adam Sapek, Gabriel Weisz, Lisa Woods, Sitaram Lanka, Steven Reinhardt, Adrian Caulfield, Eric Chung, and Doug Burger. A configurable cloud-scale dnn processor for real-time ai. pp. 1–14, 06 2018. doi: 10.1109/ISCA.2018.00012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Imperas, 2019. URL https://riscv.org/wp-content/uploads/2019/02/Imperas-EW-2019-Custom-Instructions-booth-slides-KM.pdf.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016. URL http://arxiv.org/abs/1603.08155.

Norman P. Jouppi, Doe Hyun Yoon, George Kurian, Sheng Li, Nishant Patil, James Laudon, Cliff Young, and David Patterson. A domain-specific supercomputer for training deep neural networks. *Commun. ACM*, 63(7):67–78, jun 2020. ISSN 0001-0782. doi: 10.1145/3360307. URL https://doi.org/10.1145/3360307.

Patrick Judd, Jorge Albericio, and Andreas Moshovos. Stripes: Bit-serial deep neural network computing. *IEEE Computer Architecture Letters*, 16:1–1, 08 2016. doi: 10.1109/LCA.2016.2597140.

Dhiraj D. Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A study of BFLOAT16 for deep learning training. *CoRR*, abs/1905.12322, 2019. URL http://arxiv.org/abs/1905.12322.

Urs Köster, Tristan J. Webb, Xin Wang, Marcel Nassar, Arjun K. Bansal, William H. Constable, Oğuz H. Elibol, Scott Gray, Stewart Hall, Luke Hornof, Amir Khosrowshahi, Carey Kloss, Ruby J. Pai, and Naveen Rao. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1740–1750, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342, 2018. URL http://arxiv.org/abs/1806.08342.

Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BkgXT24tDS.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. URL http://arxiv.org/abs/1405.0312. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Naveen Mellempudi, Abhisek Kundu, Dipankar Das, Dheevatsa Mudigere, and Bharat Kaul. Mixed low-precision deep learning inference using dynamic fixed point, 2017.

Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *CoRR*, abs/1603.01025, 2016. URL http://arxiv.org/abs/1603.01025.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. *CoRR*, abs/1906.04721, 2019. URL http://arxiv.org/abs/1906.04721.

NVIDIA. Int4 precision for ai inference (NVIDIA Technical Blog), Aug 2020. URL https://developer.nvidia.com/blog/int4-for-ai-inference/.

NVIDIA. TensorFloat-32 in the A100 GPU Accelerates AI Training, HPC up to 20x, 2022. URL https://blogs.nvidia.com/blog/2020/05/14/tensorfloat-32-precision-format/.

NVIDIA, 2023. URL https://github.com/nvdla.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *CoRR*, abs/1802.05668, 2018. URL http://arxiv.org/abs/1802.05668.

Sai Qian Zhang, Bradley McDanel, and H. T. Kung. Fast: Dnn training under variable precision block floating point with stochastic rounding. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 846–860, 2022. doi: 10.1109/HPCA53966.2022.00067.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Sayeh Sharify, Alberto Delmas Lascorz, Mostafa Mahmoud, Milos Nikolic, Kevin Siu, Dylan Malone Stuart, Zissis Poulos, and Andreas Moshovos. Laconic deep learning inference acceleration. In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pp. 304–317, 2019.

Synopsys, 2020. URL https://www.synopsys.com/designware-ip/processor-solutions/arc-700-family.html.

Thierry Tambe, En-Yu Yang, Zishen Wan, Yuntian Deng, Vijay Janapa Reddi, Alexander Rush, David Brooks, and Gu-Yeon Wei. Algorithm-hardware co-design of adaptive floating-point encodings for resilient deep learning inference. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020. doi: 10.1109/DAC18072.2020.9218516.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/tan19a.html.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow (extended abstract). In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4839–4843. International Joint Conferences on Artificial Intelligence Organization, 8 2021. URL https://doi.org/10.24963/ijcai.2021/662. Sister Conferences Best Papers.

Jianxun Yang, Zhao Zhang, Zhuangzhi Liu, Jing Zhou, Leibo Liu, Shaojun Wei, and Shouyi Yin. Fusekna: Fused kernel convolution based accelerator for deep neural networks. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 894–907, 2021. doi: 10.1109/HPCA51647.2021.00079.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. *ArXiv*, abs/1807.10029, 2018.

Tianyu Zhang, Lei Zhu, Qian Zhao, and Kilho Shin. Neural networks weights quantization: Target none-retraining ternary (tnt). *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, pp. 62–65, 2019.

Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. *ArXiv*, abs/1901.09504, 2019.

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights, 2017. URL https://arxiv.org/abs/1702.03044.

Feng Zhu, Ruihao Gong, Fengwei Yu, Xianglong Liu, Yanfei Wang, Zhelong Li, Xiuqi Yang, and Junjie Yan. Towards unified INT8 training for convolutional neural network. *CoRR*, abs/1912.12607, 2019. URL http://arxiv.org/abs/1912.12607.

APPENDICES

A QUANTIZATION-AWARE FINETUNING (QAT)

Fine-tuning Setups. BSFP can also be applied to quantization-aware finetuning. The fine-tuning procedure chooses SGD as the optimizer and sets the learning rate to be 10^{-6}. In addition, we set the momentum to 0.9 and weight decay to 10^{-5}. The models are fine-tuned on NVIDIA's V100 GPUs using a batch size of 100. We extend Pytorch to conduct the algorithmic experiments.

Table 3 shows the quantize-aware finetuning comparison between MSFP and BSFP for wide range of models. The key takeaway is:

- **Consistency of the benefits:** The BSFP consistently achieves higher accuracy using smaller model size and fewer operations. This demonstrates that the benefits of BSFP is consistent regardless of post-training quantization or quantize-aware training. Further, we also point out that BSFP outperforms MSFP in accuracy given ultra low bitwidth.

Table 3: Quantization-aware fine-tuning comparison of accuracy performance as well as hardware performance of ShuffleNet-v2 Ma et al. (2018), MobileNet-v2 Sandler et al. (2018), ResNets He et al. (2015) on ImageNet with MSFP and BSFP.

	Method	Precision (W / A)	Top-1 (%)	Model Size	FLOPs/ FixOPs (8b)	Method	Precision (W / A)	Top-1 (%)	Model Size	FLOPs/ FixOPs (8b)
ShuffleNet-v2	FP	32 / 32	69.18	9.1 MB	148.8M					
	MSFP	8 / 8	68.89	2.4 MB	148.8M					
		7 / 7	68.30	2.1 MB	113.9M	BSFP (OURS)	7 [5+2] / 8	**69.10**	2.3 MB	130.2M
		6 / 6	67.56	1.9 MB	83.7M		6 [3+3] / 6	**68.63**	2.0 MB	83.7M
		5 / 5	64.17	1.6 MB	58.1M		5 [4+1] / 4	**65.69**	1.7 MB	46.5M
		4 / 4	44.99	1.3 MB	37.2M		4 [3+1] / 4	**62.73**	1.4 MB	37.2M
MobileNet-v2	FP	32 / 32	71.84	14.0 MB	314.1M					
	MSFP	8 / 8	71.71	3.7 MB	314.1M					
		7 / 7	71.19	3.3 MB	240.5M	BSFP (OURS)	7 [5+2] / 8	**71.80**	3.5 MB	274.8M
		6 / 6	70.27	2.9 MB	176.7M		6 [3+3] / 6	**71.17**	3.0 MB	176.7M
		5 / 5	65.25	2.4 MB	122.7M		5 [4+1] / 4	**67.38**	2.6 MB	98.2M
		4 / 4	45.73	2.0 MB	78.5M		4 [2+2] / 4	**66.29**	2.2 MB	78.5M
ResNet-18	FP	32/32	69.76	46.8 MB	1.82G					
	MSFP	8 / 8	69.84	12.4 MB	1.82G					
		7 / 7	69.73	11.0 MB	1.39G	BSFP (OURS)	7 [5+2] / 8	**70.01**	11.6 MB	1.59G
		6 / 6	69.64	9.5 MB	1.02G		6 [3+3] / 6	**69.85**	10.1 MB	1.02G
		5 / 5	68.94	8.0 MB	710.9M		5 [3+2] / 4	**69.09**	8.7 MB	568.8M
		4 / 4	64.76	6.6 MB	455.0M		4 [2+2] / 4	**69.02**	7.2 MB	455.0M
		3/ 3	46.59	5.1 MB	255.9M		3 [2+1] / 3	**64.85**	5.8 MB	255.9M
ResNet-50	FP	32/32	76.13	102.2 MB	4.14G					
	MSFP	8 / 8	76.26	27.2 MB	4.14G					
		7 / 7	76.17	24.0 MB	3.17G	BSFP (OURS)	7 [5+2] / 8	**76.30**	25.4 MB	3.6G
		6 / 6	76.06	20.8 MB	2.3G		6 [4+2] / 6	**76.17**	22.2 MB	2.3G
		5 / 5	75.12	17.6 MB	1.6G		5 [4+1] / 4	**76.00**	19.0 MB	1.6G
		4 / 4	71.57	14.4 MB	1.0G		4 [2+2] / 4	**75.04**	15.8 MB	1.0G
		3 / 3	53.47	11.2 MB	582.2M		3 [2+1] / 3	**70.95**	12.6 MB	582.2M

B COMPARISON BETWEEN STANDARD QAT AND THE PROPOSED LOW-COST QAT

To amortize the overheads incurred by grid search during quantization-aware training (QAT), we propose to quantize only the first batch for every 100 batches to amortize the quantization overheads. We refer this proposal as low-cost QAT.

To answer the question whether the proposed low-cost QAT will affect the training process, Figure 10 compares the training curve of the standard QAT and low-cost QAT on finetuning ShuffleNet-v2 and MobileNet-v2. The takeaway is that the training curve of low-cost QAT is similar to the curve

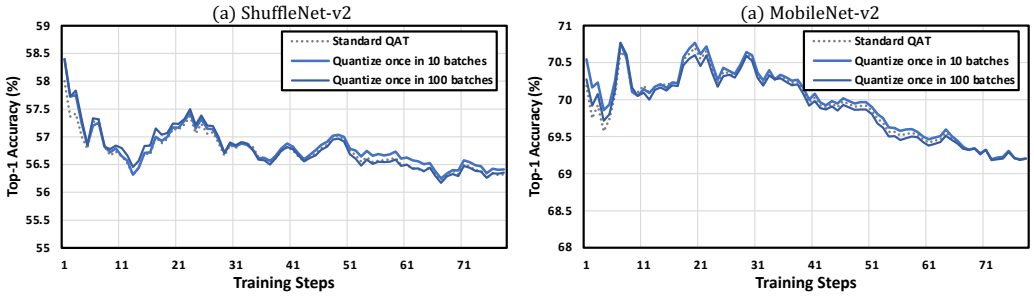


Figure 10: Training curve comparison between standard quantization-aware training and low-cost quantization-aware training for (a) ShuffleNet-v2 and (b) MobileNet-v2.

standard QAT, which quantizes before every batch. This demonstrates that the low-cost QAT can preserve the training curve while requiring significantly lower search overheads.

C Evaluations on Other Tasks

C.1 Optical Flow Estimation using RAFT on KITTI dataset

Since our research direction focuses on computer vision, we choose to extend evaluate for BSFP on DNN-based optical flow estimation Teed & Deng (2021) (KITTI dataset Menze & Geiger (2015)). Table 4 shows that the BSFP consistently outperforms the MSFP using same weight precision, which demonstrates the benefit of BSFP is consistent regardless of the tasks.

Table 4: MSFP and BSFP on RAFT Teed & Deng (2021) for KITTI dataset Menze & Geiger (2015). The BSFP consistently improves the error given same weight precision.

Optical Flow (RAFT)	MSFP	BSFP
Precision (W)	Average end-point error	
9	0.658	0.632
7	0.895	0.682
5	2.068	0.905

D Details of the Criterion-optimal quantization flow

In order to find the optimal subword mantissas and scalings, we adopt iterative rounding, which contains the following steps:

1. Enumerate the two scaling factors.
2. List the quantization levels the two scaling factors can generate. The number of quantization levels is moderate. For example, there are only 64 levels if the two subwords are 2+4 bits.
3. Calculate the MSE between the 16 original weights and their nearest quantization levels. Finding the nearest quantization level in a list is viable as other quantization schemes, such as APoT Li et al. (2020), also employ it.
4. Keep the scaling factors that achieve the lowest MSE and go to step 1.

The pseudo code of the criterion-optimal quantization flow is presented in Algorithm 1

E Supporting BSFP on Bit-parallel Architecture

Since some of the hardware, e.g., CPU and GPU, prefers bit-parallel PE architecture, Figure 11 presents the bit-parallel BSFP PE design. The difference between serial PE architecture and parallel

Algorithm 1 Criterion-optimal quantization flow

Input: Full-precision weight W_{fp}
Output: Criterion-optimal scalings sc_{opt}, Criterion-optimal subword mantissas man_{opt}
Require: Subword configurations $config_{sub}$, Scaling configurations $config_{sc}$

```
Criterion ← max
SC ← Enum(config_sc)                          ▷ Enumerate all scaling setups to form search space SC
for sc_cur ∈ SC do                            ▷ For every sc_cur of the search space SC
    list_quant ← BuildQlist(sc_cur, config_sub)   ▷ Find all quantization levels using sc_cur
    W_quant ← Quant(W_fp, list_quant)         ▷ Quantize W_fp to its nearest levels in list
    Criterion_cur ← MSE(W_fp, W_quant)        ▷ Compute the MSE criterion
    if Criterion_cur < Criterion then.        ▷ Update the scale and subword mantissas
        sc_opt ← sc_cur
        man_opt ← man_cur
        Criterion ← Criterion_cur
    end if
end for
```

PE architecture is that the scaling units need to be unrolled, which incurs some overheads. The BSFP's benefits of smaller model size and lower arithmetic complexity remain unchanged.

Further, we can achieve uneven subword configurations by fixing the upper multipliers and adjusting only the weight precision of lower multipliers (red weights).

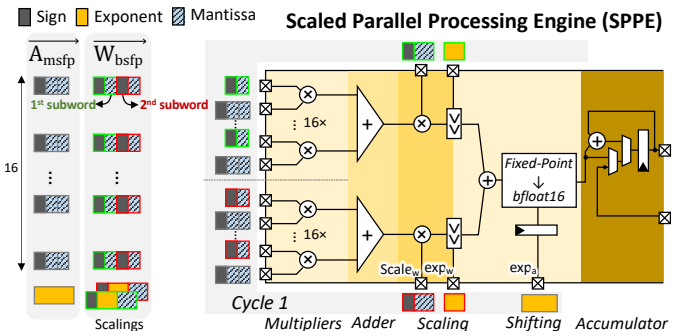


Figure 11: Proposed scaled parallel processing engine.

BSFP can be efficiently deployed in newly-manufactured devices, including customized AI accelerator IPs, which represent a significant amount of AI products including smartphones and smart appliances. For example, Nvidia NVDLA NVIDIA (2023), Qualcomm Hexagon Tensor Accelerator Codrescu (2013), and MIT Eyeriss Chen et al. (2017). Such IPs are embedded in emerging AI chips for smartphones, appliances, robots, etc. Since customized, embedded AI accelerator IPs are not obligated to support standard data types, the proposed BSFP type, which improves weight quantization quality (including post-training quantization, which is even difficult and desirable), will stand out as a candidate for chip designers to consider and adopt in their new design.

As for CPUs, although they are not our only focus, letting newly-manufactured CPUs support BSFP is not difficult, and the computation will be efficient. Modern CPUs, including RISC-V, ARM, and Synopsys ARC, allow custom ISA extensions Imperas (2019); Arm (2020); Synopsys (2020). Designers can provide an add-on digital circuit that natively and efficiently supports BSFP using low-bit, fixed-point adders and multipliers. Vendors already offered the required design/simulation flow, CPU generator, and toolchain generator to facilitate designers to employ the extended instructions, e.g., vector product instructions between BSFP weights and MSFP activations.

F Details of the Hardware Comparison

Processing Engine Details: Table 2 compares 16-wide PEs in different number formats, i.e., **BF16, INT4, INT8, Power-of-Two, MSFP, and BSFP**, following similar setup in FAST Qian Zhang et al. (2022). The architectural details for these PEs are summarized below:

- The **BF16 (brain float)** PE adopts bit-parallel architecture, which computes 16-wide BF16 multiplications and reduces them to a single partial sum. The accumulator is in BF16 format.
- The **INT4** and **INT8** PEs also adopt bit-parallel architectures, which perform 16-wide MAC and generate 12b and 20b partial sum, respectively. The accumulator then accumulates it using INT32 format.
- The **Power-of-Two** PE shifts 16 inputs based on the weight values and reduces 16 products to a single partial sum. The accumulator is in INT32 format.
- The serial **MSFP** PE Qian Zhang et al. (2022) computes 16 2b-to-2b multiplications per cycle and shifts them to accumulate for the correct partial sum. We adopt the BF16 accumulator.
- The proposed S^2IP supports **BSFP**, which computes 16-wide 2b or 1b multiplications per cycle and scale-and-shifts them to accumulate for correct partial sum. The accumulator is in BF16 format.

Additional Notes:

- The adder tree output precisions of every PEs are optimized to their minimum.
- ALL of the PEs obtain similar slack profiles.

Performance Analysis Setup: We adopt iso-area performance setup to fairly compare different number systems, which is widely adopted by top architecture papers Sharify et al. (2019); Yang et al. (2021). In other words, if the size of BF16 PE is 26× larger than that of the BF16 PE, we are allowed to allocate 26× more PEs for BSFP. Besides PE parallelism, we also consider the multi-cycle computation of serial architecture to reasonably estimate the performance.

Power Analysis Setup: We implement all of the above PEs using Verilog and synthesis them using Synopsys Design Compiler (topographical mode) on the TSMC 40 nm node. Specifically, the power of each module is analyzed with average switching activity. We then utilize the power profiles to estimate throughput per Watt.

G Per-Layer Skewness Analysis

Figure 12 analyzes the absolute skewness for weights of each layer without slicing weights into vectors. We observe that most layers' skewness is low (i.e., $\leq$0.5), which matches the summary of prior works that weight values follow bell-shape and non-skewed distribution.

Through a side-by-side comparison between Figure 12 and Figure 13 (same as Figure 7; We copy it here for easier comparison), we clearly observe that vectorization is one source of more considerable skewness on Figure 13.

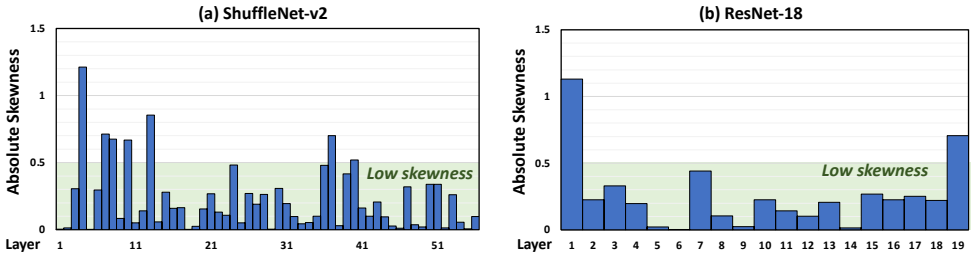


Figure 12: Per-layer absolute skewness of (a) ShuffleNet-v2 and (b) ResNet-18. Most of the layers obtain low skewness, which matches the analysis reported by prior works.

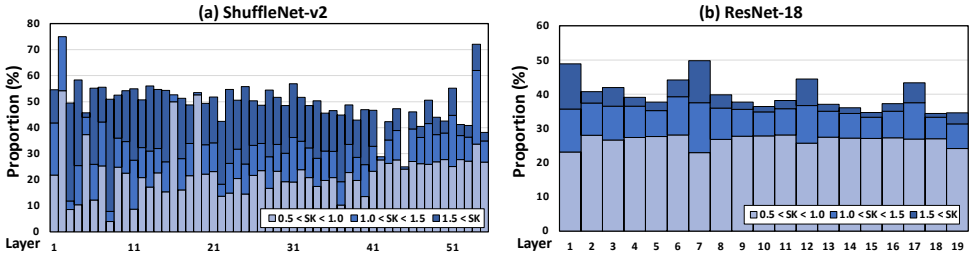


Figure 13: Per-vector absolute Pearson's skewness coefficient (SK) of (a) ShuffleNet-v2 and (b) ResNet-18. In general, very highly skewed means $SK > 1.5$, highly skewed means $SK > 1.0$, and moderately skewed means $1.0 > SK > 0.5$.

H Searching For The BSFP Configurations

To obtain the scaling factor configurations (1-4-3, 1-3-3) and the exponent biases (-8, -3), we use the following steps to empirically set them. This method is acceptable as neural networks naturally posses many hyper parameters.

1. Begin with setting scaling factors to BF16 (1-7-8), a sufficiently precise format.
2. Given the scaling factors, find the narrowest bitwidths of subwords that causes an accuracy drop less than a first budget. For example, 5+2 bits are selected.
3. Given the subword settings, reduce the bitwidths of scaling factors and sweep the the biases to a point that the overall accuracy drop is less than a second budget. This step lead us to (1-4-3, 1-3-3) and biases (-3,-8).