Improving Efficiency in Large Language Models via Extendable Block Floating Point Representation

Dongyang Li[1], Zeyang Li[2], Bosheng Liu[1]*, Jigang Wu[1]*
[1]School of Computer Science and Technology, Guangdong University of Technology
[2]School of Automation, Guangdong University of Technology
liubosheng@gdut.edu.cn, asjgwucn@outlook.com

Abstract

Large language models (LLMs) have revolutionized natural language processing (NLP) tasks, yet their increasing size poses substantial challenges in terms of computational and memory resources. Block floating-point (BFP) arithmetic offers an effective solution by leveraging the strengths of both floating-point and fixed-point representations, leading to reductions in both storage and computational overhead. However, current low-bit BFP quantization approaches often struggle to handle extreme outliers, leading to significant accuracy degradation. To overcome this limitation, we introduce Extendable Exponent Sharing (EES), a novel BFP representation that extends the exponent bit width to capture a wider dynamic range. EES achieves this by embedding extendable exponent bits into the least significant mantissa bits, thereby increasing the shared exponent's bit width without incurring additional storage costs. To optimize the trade-off between accuracy and energy efficiency, EES employs a design space exploration strategy to optimize the configuration of extendable exponent bit widths. Experimental results show that EES outperforms representative baselines in both accuracy and computational efficiency.

1 Introduction

Large language models (LLMs) have achieved remarkable success across a variety of natural language processing tasks, such as intelligent assistants, machine translation, and sentiment analysis (Nam et al., 2024; Kim et al., 2024; Zhao et al., 2024). However, as the size of models continues to scale, the computational and memory requirements grow substantially (Liu et al., 2024). Block floating point (BFP) representation provides an effective solution to address these challenges, balancing precision with efficiency, making it particularly well-suited to meet the resource-intensive demands of LLMs (Zeng et al., 2024; Qin et al., 2024). By grouping multiple values under a shared exponent, BFP enables efficient storage and calculation of large model parameters with minimal overhead (Darvish Rouhani et al., 2023; Zhang et al., 2024). In comparison to both fixed-point and floating-point formats, BFP offers distinct advantages, especially in large-scale computations. It minimizes memory usage by sharing an exponent across values, making it more storage-efficient than full-precision floating point, while providing greater numerical stability than fixed-point representation (Noh et al., 2023; Shen et al., 2024).

Recent advancements in low-bit BFP quantization have shown great promise in reducing memory and computational costs for LLM applications (Song et al., 2018; Lo et al., 2023). BFP effectively balances numerical precision and resource efficiency (Lo and Liu, 2023), but its limited exponent range in low-bit formats presents significant challenges. For example, in an 8-bit BFP representation using the E3M4 format, a block of numbers shares a 3-bit exponent and uses 4-bit mantissas. If the shared exponent is set to 4 ($2^4 = 16$), the largest representable value in the block is 240 ($15 \times 16 = 240$). Any value exceeding this, such as 300, results in overflow, as the shared exponent cannot accommodate larger values within the given bit width. This limitation becomes particularly problematic for extreme outliers, which are prevalent in the parameters and activations of LLMs, as highlighted in Fig. 1. The inability to accurately handle these outliers often causes significant accuracy degradation (Zhang et al., 2023; Nie et al., 2019; Chen et al., 2023). Solving this issue is essential for enabling the broader application of BFP in high-precision LLM workloads.

To mitigate the challenges posed by extreme outliers in low-bit BFP quantization, various techniques have been proposed. SuperWeight (Yu et al., 2024) tackles the issue by identifying and

*Corresponding author: Bosheng Liu and Jigang Wu

Findings of the Association for Computational Linguistics: ACL 2025, pages 14861–14873

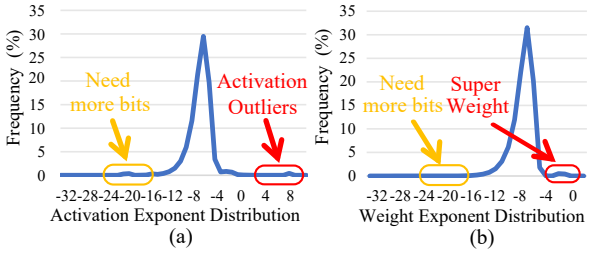

Figure 1: Statistical exponent distribution of input activations and weights based on the layers.1.mlp.down_proj of the Llama-3.1-8B model.

preserving critical outlier weights during quantization, achieving a balance between compression efficiency and accuracy. Similarly, SmoothQuant (Xiao et al., 2023) adopts a per-channel scaling approach to redistribute value ranges, effectively shifting the quantization burden of outliers from activations to weights while preserving mathematical equivalence. Building on these strategies, BiE (Zou et al., 2024) introduces a dual-exponent mechanism to address outliers, ensuring improved accuracy. However, this method incurs additional resource overhead due to the inclusion of flag bits during computational implementation. Despite these advancements, existing methods only partially alleviate the challenges of low-bit BFP quantization (Jang et al., 2024). The inherent limitation of a narrow exponent width remains a critical bottleneck, restricting the efficient handling of extreme values and limiting computational performance.

In this work, we introduce Extendable Exponent Sharing (EES), a post-training quantization approach based on BFP designed to improve dynamic range and computational efficiency. EES extends the traditional shared exponent method by increasing the exponent bit width to cover a broader range of values. However, directly expanding the exponent width increases the storage requirements. To address this, we observe that changes in lower-order bits have a minimal effect on computations and model accuracy. For example, two 8-bit BFP numbers—1101.1010 and 1101.1001—differing only in the least significant bit show a minimal impact on precision, indicating that small changes in lower-order bits do not significantly affect accuracy. Using this insight, EES embeds the extended exponent bits into the least significant bits of the partial mantissas. These bits are then combined with the shared exponent to form an extended exponent, as shown in Fig. 2. Since this approach affects the precision of mantissas, we analyze the quantization error introduced by EES compared to standard BFP. To achieve an optimal balance between accuracy and computational efficiency, we use a design space exploration strategy to obtain the best configurations for both activations and weights. Evaluations in LLMs show that EES significantly improves both accuracy and energy efficiency. Key contributions of this work include:

- Introduction of EES: We introduce a new numerical representation, Extendable Exponent Sharing (EES), which uses extendable exponents to better handle outliers, resulting in improved accuracy and computational efficiency.
- Optimized Exponent Configuration: We analyze the distribution of exponents in activations and weights, and apply a design space exploration strategy to determine the optimal bit widths for the extended exponent.
- Enhanced Performance: Experimental results indicate that EES delivers up to a 1.38× improvement in energy efficiency, alongside a maximum accuracy boost of 0.36% when compared to cutting-edge BFP techniques.

2 Related Work

Quantization is a key technique used to compress LLMs by lower-precision data formats, significantly reducing memory usage and computational costs(Rokh et al., 2022). The two dominant data formats in this space are fixed-point (Lee et al., 2024) and BFP (Zeng et al., 2024) representations. Fixed-point representation simplifies operations like multiplication and accumulation compared to floating-point formats, but it has limited precision due to the fixed offset and scaling.(Nair et al., 2021; Benmaghnia et al., 2022) Differently, BFP combines the computational efficiency of fixed-point arithmetic with the dynamic range of floating-point formats, offering an effective balance between precision and performance(Basumallik et al., 2022).

Recent progress in BFP quantization, including techniques like DBPS (Lee et al., 2023), Static BFP (Fan et al., 2021), and MSFP12 (Darvish Rouhani et al., 2020), has shown its ability to achieve high accuracy, efficient computation, and reduced storage costs. Additionally, research into low-bit BFP representations aims to improve these advantages. For example, FAST (Zhang et al., 2022a) introduces low-bit BFP quantization for activations and

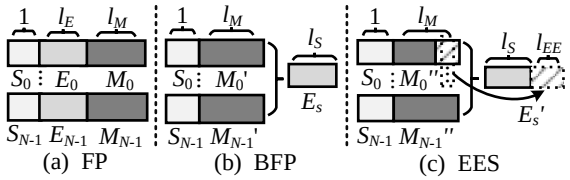

Figure 2: Profile of extended exponent sharing by compared with floating-point (FP) and BFP. The shared exponent of EES consists of l_S bits of original shared exponent and l_{EE} bits of extendable shared exponent.

weights, supporting efficient matrix multiplications with reduced precision. Similarly, Bucket (Lo and Liu, 2023) and FIGNA (Jang et al., 2024) adopt 11-bit and 8-bit BFP formats, respectively, to enhance computational performance and energy efficiency.

In the BFP representation, data can be classified into two categories: "normal values," which are accurately represented, and "extreme outlier values," which fall outside representable range and require approximation through shifting or scaling. These outliers are a major challenge in low-bit BFP quantization, as they can significantly degrade model accuracy. Techniques like AWQ (Lin et al., 2024) and SuperWeight (Yu et al., 2024) identify and preserve "super weights" — critical parameters with extreme values — to mitigate the negative effects of outliers on model quality. BiE (Zou et al., 2024) introduces dual shared exponents to manage both normal values and outliers in both activations and weights, ensuring high accuracy for LLM models.

While these methods address outlier challenges, the limited bit width of the shared exponent in BFP remains a key limitation for representing a wider range of values. We propose a scalable shared exponent strategy that extends the range of numerical representation and improves the performance of low-bit BFP quantization for LLMs.

3 Preliminary

As shown in Fig. 2(a), a set of N floating-point numbers, denoted as X, can be given by:

$$X = [(-1)^{s_0} m_0 2^{E_0}, \ldots, (-1)^{s_{N-1}} m_{N-1} 2^{E_{N-1}}], \tag{1}$$

where s_i, m_i, and E_i are the i-th sign, mantissa, and exponent ($0 \leq i \leq N-1$), respectively.

When converting floating-point data into BFP representation, we obtain the new data set X', as illustrated in Fig. 2(b), which is given by:

$$X' = [(-1)^{s_0} m'_0, \ldots, (-1)^{s_{N-1}} m'_{N-1}] \cdot 2^{E'_S}, \tag{2}$$

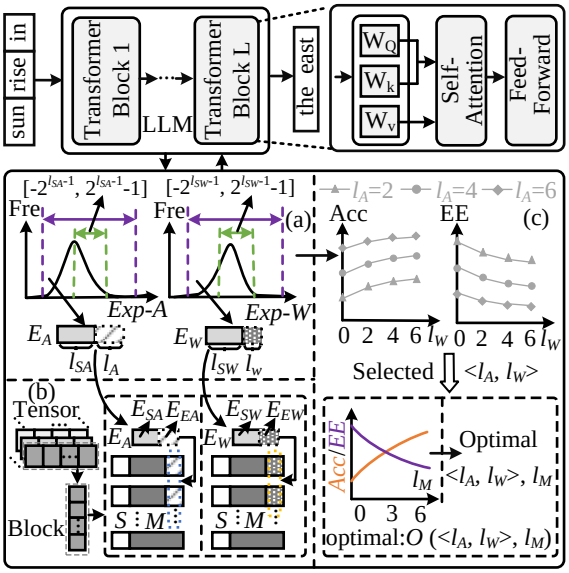

Figure 3: Workflow. (a) Establishing exponent bit width.(b) Extended exponent sharing. (c) Bit width space exploration. Acc denotes the model accuracy, while EE denotes the energy efficiency.

where E'_S is the shared exponent, calculated as:

$$E'_S = \max E_i \mid i \in 0, \ldots, N-1. \tag{3}$$

The bit width of shared exponent E'_S equals to

$$l_{E'_S} = l_S \tag{4}$$

where l_S denotes the length of the shared exponent.

For each mantissa in this BFP representation, m'_i is computed by shifting the original mantissa m_i based on the difference between the maximum shared exponent (E'_S) and the individual exponent (E_i), using the right shift operation:

$$m'_i = m_i \gg (E'_S - E_i) \tag{5}$$

where $\gg$ denotes the right shift operation.

Assuming the value of input activations and weights are known and the rounding-to-nearest method is used (Kalliojarvi and Astola, 1996; Song et al., 2018), the mean of α is zero, and the error variance (σ^2_α) can be formulated as

$$\sigma^2 = \frac{2^{-2l_m}}{12} \cdot 2^{2 \cdot E'_S} \tag{6}$$

where l_m is the bit width of BFP mantissa.

4 Methodology

This section exhibits the EES framework, detailing the quantization workflow, error analysis, and hardware architecture.

4.1 Overview

Fig. 3 depicts the EES quantization workflow, which comprises three primary components: exponent bit width establishment, EES numerical representation, and bit width space exploration. The exponent bit width establishment component determines the bit widths for activations and weights by analyzing their exponent distributions. The EES numerical representation component defines the extendable exponent format during the transformation from FP to EES format. The bit width space exploration component optimizes the trade-off between model accuracy and energy efficiency to determine the optimal EES quantization configuration.

Exponent Bit Width Establishment. Fig. 3(a) illustrates the process of determining the exponent bit widths for activations and weights based on their statistical distribution ranges. The original shared exponent bit widths, l_{SA} for activations and l_{SW} for weights, restrict the representable ranges to $[-2^{l_{SA}-1}, 2^{l_{SA}-1}-1]$ and $[-2^{l_{SW}-1}, 2^{l_{SW}-1}-1]$, respectively. These ranges, however, are insufficient to fully capture data with large outliers, which poses challenges for accurate representation.

To address this, the shared exponent bit-width for activations and weights are augmented by introducing extendable shared exponent bits, thereby expanding the dynamic range. The total shared exponent bit width includes two components: the original shared exponent bit width (l_{SA} and l_{SW}) and the extendable shared exponent bit width (l_A for activations and l_W for weights). This unified representation is expressed as:

$$l_{E_S} = l_S + l_{E_E} \tag{7}$$

where l_{E_S} represents the total shared exponent bit width, comprising the original bit width l_S and the extendable bit width l_{E_E}.

Following this expansion, the range of the shared exponent E_S is given by:

$$E_S \in [-2^{l_S+l_{E_E}-1}, 2^{l_S+l_{E_E}-1}-1] \tag{8}$$

EES Numerical Representation. Fig. 3 (b) illustrates the EES numerical representation, which is derived from the conversion of original floating-point data within a tensor to the extendable exponent sharing format in a block. The EES representation, denoted as X'', is expressed as:

$$X'' = [(-1)^{s_0} m''_0, ..., (-1)^{s_{N-1}} m''_{N-1}] \cdot 2^{E_S} \tag{9}$$

where E_S represents the shared exponent.

The incorporation of the extendable exponent into the partial least significant bits of the private mantissas modifies the mantissa representation. The updated mantissa, m''_i, is defined as:

$$m''_i = \begin{cases} \{m'_i[l_m-1:1], E_E[i]\}; & 0 \le i \le l_{E_E}-1 \\ m'_i[l_m-1:0]; & l_{E_E} \le i \le N-1 \end{cases} \tag{10}$$

where $m'_i[l_m-1:1]$ denotes the bits ranging from the first to the (l_m-1)-th position of the original mantissa m'_i. Here, E_E, the value of the extendable exponent, is derived from the least significant bits of the shared exponent E_S and is represented as:

$$E_E = E_S[l_{E_E}-1:0] \tag{11}$$

Vector Multiplication in Blocks. Given two blocks X_1 and X_2 with N elements in EES format, the dot product is computed as:

$$X''_1 \cdot X''_2 = 2^{E_A+E_W} \sum_{i=0}^{N-1} (-1)^{S_{Ai} \oplus S_{Wi}} (M_{Ai} \cdot M_{Wi}) \tag{12}$$

where E_A and E_W represent the effective exponents for X''_1 and X''_2, respectively, defined as:

$$\begin{aligned} E_A &= (E_{SA} << l_A) + E_{EA} \\ E_W &= (E_{SW} << l_W) + E_{EW} \end{aligned} \tag{13}$$

where E_{SA} and E_{SW} denote the original shared exponent bit width for X''_1 and X''_2, respectively. E_{EA} and E_{EW} respectively correspond to the extendable exponent values for X''_1 and X''_2. l_A and l_W indicate the bit widths of extendable exponents.

Bit width Space Exploration. To optimize the extendable exponent bit width, the design space exploration process for activations and weights is illustrated in Fig. 3(c). The objective is to optimize the function O, formulated as:

$$\text{optimal}: O = \begin{cases} AP(<l_A, l_W>, l_{m''_i}) + \alpha \cdot EE(& \\ <l_A, l_W>, l_{m''_i}) & AP = Acc \\ \frac{1}{AP(<l_A, l_W>, l_{m''_i})} + \alpha \cdot EE(& \\ <l_A, l_W>, l_{m''_i}) & AP = Per \end{cases} \tag{14}$$

where AP take values of Acc and Per. Acc, Per, and EE respectively denote the accuracy, perplexity, and energy efficiency. α is a balancing factor. l_A and l_W correspond to the bit widths of the extendable exponents for activations and weights, respectively. $l_{m''_i}$ specifies the length of the mantissa.

4.2 Error Analysis

The transition from FP to EES introduces errors originating from two primary sources. First, rounding errors arise during the quantization of FP values into the BFP format, governed by the specified mantissa and exponent lengths. Second, additional numerical errors occur due to the integration of the extendable exponent component (E_E) into the least significant bits of the BFP mantissas.

For a block containing N elements, the error α for the conversion from FP to EES is given by:

$$\begin{aligned}\alpha &= \sum_{i=0}^{N-1} |x_i'' - x_i| \leq \sum_{i=0}^{N-1} ||x_i'' - x_i'| + |x_i' - x_i|| \\ &\leq \sum_{i=0}^{l_{E_E}-1} |m_i'' - m_i'| \cdot 2^{-l_{m_i''}} + \sum_{i=0}^{N-1} |m_i' - m_i| \cdot 2^{E_S} \\ &\leq l_{E_E} \cdot 2^{-l_{m_i''}} + N \cdot (2^{-l_m' + E_S} - 2^{-l_m + E_S})\end{aligned} \tag{15}$$

where x_i, x_i' and x_i'' are the i-th number of the FP, quantified by the shared exponent and EES representations, respectively, while l_{mi}, l_{mi}' and l_{mi}'' represent the mantissa width of x_i, x_i' and x_i''. l_{E_E} and E_S respectively denote the extendable exponent bit width and extended shared exponent.

Based on Equations (6)-(15), the error variance (σ_α^2) for the transition from FP to EES is given by:

$$\frac{4^{-l_{m_i''}-1+E_S}}{3} \leq \sigma_\alpha^2 \leq \frac{4^{-l_{m_i''}+E_S}}{3} \tag{16}$$

This range arises from replacing the least significant bit of the original mantissa in EES. Specifically, when the mantissa bit-width is l_m'', the error variance σ_α^2 is at least as large as the error variance introduced during the FP-to-BFP conversion. However, when accounting for the bit replacement, the error variance remains no greater than the variance observed in the FP-to-BFP conversion with a reduced mantissa bit-width of $l_m'' - 1$.

4.3 Hardware Architecture

Fig. 4 depicts a hardware architecture designed to support EES-based computations, consisting of three primary components: the on-chip memory (AB, OB, and WB), the processing element (PE), the decoder (Dc) and encoder (Ec). The memory units (AB, OB, WB) temporarily store input activations, output activations and weights, respectively. The PE performs EES-based multiplication and accumulation. Dc converts EES-format data into BFP format for computation, while Ec encodes output data from FP format back into EES representation.

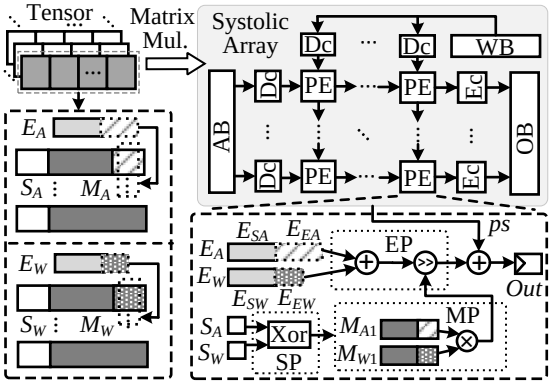

Figure 4: EES-based hardware architecture. ps denotes the partial sum propagated from the preceding PE.

The PE is structured into three key components: SP, MP, and EP. SP handles the XOR operations for the sign bits, MP performs the mantissa multiplications, and EP processes the combined exponent, including both the original and the extended shared exponent. The intermediate results from these computations are stored in the Out registers, where they are accumulated to generate the final outputs.

Fig. 5 depicts the Dc and Ec components. The Dc component (Fig. 5(a)) decodes both activations and weights by converting EES-format data into BFP format for calculations. It uses a selector to extract the last bit of mantissas and employs two adders and shifters to concatenate the extended exponent. Ec (Fig. 5(b)) encodes the output activations back into the EES format. It first uses a selector and a shifter to convert the original data into BFP format. After that, two selectors separately choose the mantissa and shared exponent bits. Finally, an adder and a shifter concatenate the data into the final EES representation.

5 Evaluation

5.1 Experimental setup

Baselines. We evaluate the EES framework across three key aspects. First, we compare EES against three representative post-training low-bit quantization baselines: Tender (Lee et al., 2024), FINGA (Jang et al., 2024), and BiE (Zou et al., 2024). Tender employs INT4 quantization with tensor decomposition and implicit re-quantization to achieve efficient low-precision acceleration of LLMs. FINGA adopts a BFP format with optimized multiplication operations to reduce storage overhead and en-

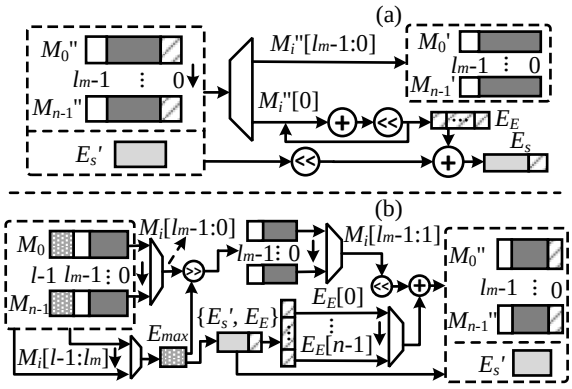

Figure 5: Dc (a) and Ec (b) components for decoding and encoding, respectively.

Extendable exponent bit width (l_{E_E})		mantissa bit width (l''_{mi})
Activation (l_A)	Weight (l_W)	{2, 3, 4, 5, 6}
{1, 2, 3, 4, 5, 6}	{1, 2, 3, 4, 5, 6}	

Table 1: Quantization space exploration under different extendable exponents (l_A, l_W) and mantissas (l''_{mi}).

hance computational efficiency. BiE incorporates a double-exponential mechanism to handle outliers while preserving model accuracy. Second, we assess hardware performance, memory efficiency, and energy and area efficiency, benchmarking EES against these baselines. Finally, we perform a quantization space exploration for extendable shared exponents for activations and weights under various bit-width configurations (Table 1) for optimal accuracy performance and energy efficiency.

Models and Datasets. EES is evaluated on two major LLM families: the OPT suite (6.7B and 13B variants) (Zhang et al., 2022b) and Llama (Llama2-7B, Llama2-13B, and Llama3.1-8B) (Touvron et al., 2023). Benchmarks include Hellaswag (Zellers et al., 2019), ARC_challenge (Min, 2023), LAMBADA_OpenAI (Nguyen et al., 2024), and LAMBADA_standard (Paperno et al., 2016). All accuracy results are conducted in a high-performance computing environment with an NVIDIA RTX 4090 GPU, leveraging the PyTorch 2.1.0 framework.

Implementations. To simulate EES, activations and weights are quantized using PyTorch. Baseline configurations include Tender with 4-bit integer, FINGA with 4-bit private mantissa and 3-bit shared exponent, and BiE with a 4-bit private mantissa, two 5-bit shared exponents and a flag bit. EES uses a W4A4 configuration with 4-bit mantissas for both weights and activations. All BFP baselines and EES are evaluated in a block size of 16.

Architecture	Config	Frequency [HZ]	Tech [nm]	Power [mw]	Area [μm^2]
Tender	W4A4	1G	28	0.266	488.28
FIGNA	W4A4	1G	40	0.317	367.08
BiE	W4A4	1G	45	0.426	534.93
EES	W4A4	1G	45	0.334	384.37

Table 2: Power and area consumption of one PE. The results of baselines come from their literature. The configuration labeled as "W4A4" indicates that both the weights and activations utilize 4-bit mantissas.

Component	Number	Power [w]	Power Ratio(%)	Area [mm^2]	Area Ratio(%)
Ec	16	0.00339	0.0078	0.00255	0.0164
Dc	32	0.00489	0.0112	0.00359	0.0343
EES-PE	256	0.428	0.9810	0.0984	0.9413

Table 3: Power and area breakdown of EES.

To evaluate hardware behavior, the EES architecture was implemented in Verilog RTL and verified through simulations. Power analysis was conducted using Synopsys Design Compiler with 45nm TSMC technology library. Table 2 lists parameter settings for comparisons. The power of baselines for each PE comes from their literature. Table 3 breaks down area and power consumption of EES, showing that encoding and decoding components contribute minimally (0.02%-0.03%). Performance and energy efficiency were assessed using the MAESTRO (Kwon et al., 2020) simulator, with LLM models as inputs. MAESTRO provides a framework for analyzing LLM deployments on hardware architectures, delivering detailed insights into performance and energy efficiency.

5.2 Simulation Results

Accuracy and Perplexity Comparison. Table 4 compares accuracy and perplexity across methods. While EES shows a slight accuracy decline compared to the original FP16 baseline, this is primarily due to the extendable shared exponent strategy introduced during quantization. Despite this small trade-off, EES significantly enhances energy efficiency via low-bit extendable shared exponent quantization. Compared to the INT4-based Tender method, EES achieves higher accuracy, thanks to the broader dynamic range of its data format.

EES demonstrates superior performance over BFP-based FINGA in both accuracy and perplexity. For example, in the OPT-6.7B model, using the Hellaswag dataset, EES improves accuracy by

Method		FP	Tender	FINGA	BiE	EES
OPT-6.7B	Hellaswag (↑)	48.06%	45.58%	46.50%	47.33%	47.88%
	ARC_challenge(↑)	30.63%	29.82%	29.21%	29.73%	29.89%
	LAMBADA_OpenAI(↓)	4.25	5.30	6.12	4.93	5.11
	LAMBADA_Standard(↓)	5.38	6.55	6.72	6.59	5.68
OPT-13B	Hellaswag (↑)	48.40%	45.58%	46.89%	47.37%	47.88%
	ARC_challenge(↑)	33.31%	31.99%	31.90%	33.06%	32.78%
	LAMBADA_OpenAI(↓)	4.08	6.62	5.67	5.69	5.01
	LAMBADA_Standard(↓)	5.67	9.19	7.13	6.49	6.48
Llama2-7B	Hellaswag (↑)	49.88%	48.50%	49.24%	48.91%	49.34%
	ARC_challenge(↑)	44.99%	40.69%	44.42%	43.01%	44.45%
	LAMBADA_OpenAI(↓)	3.34	5.42	5.10	3.97	4.44
	LAMBADA_Standard(↓)	4.27	7.13	6.10	5.84	5.60
Llama2-13B	Hellaswag (↑)	50.11%	48.03%	48.40%	49.64%	49.49%
	ARC_challenge(↑)	49.00%	45.30%	47.80%	48.75%	48.31%
	LAMBADA_OpenAI(↓)	2.94	4.36	3.08	3.66	3.08
	LAMBADA_Standard(↓)	3.77	6.10	3.94	4.76	3.83
Llama3.1-8B	Hellaswag (↑)	52.20%	50.11%	50.80%	51.18%	51.73%
	ARC_challenge(↑)	51.19%	50.31%	50.14%	50.19%	50.67%
	LAMBADA_OpenAI(↓)	3.05	4.95	3.22	3.62	3.06
	LAMBADA_Standard(↓)	4.01	6.50	4.34	5.47	4.16

Table 4: Accuracy/perplexity comparison between EES and baselines. (↓) indicates that lower perplexity is better for the datasets, while (↑) indicates that higher accuracy is better.

Benchmark	OPT -6.7B	OPT -13B	Llama2 -7B	Llama2 -13B	Llama3.1 -8B
Config	W4A4	W4A4	W4A4	W3A3	W4A4
$<l_{E_{SA}}, l_{E_{SW}}>$	$<5,4>$	$<5,4>$	$<5,5>$	$<5,4>$	$<6,5>$
Accuracy (%)	47.88	47.67	49.45	48.76	51.86
Energy Efficiency	13.60×	13.60×	13.59×	17.75×	13.58×

Table 5: The optimal accuracy and energy efficiency (normalized with FP16) are based on a bit-width space exploration conducted under different mantissa and extendable exponent bit-width setting on benchmarks and Hellaswag dataset. $< l_{E_{SA}}, l_{E_{SW}} >$ denotes the combination of activation and weight of extended shared exponent bit-width. The exploration results on ARC_challenge, LAMBADA_OpenAI and LAMBADA_Standard datasets are shown in Appendix A.3.

1.38%, while on the LAMBADA_Standard dataset, it reduces perplexity by 1.11. This is attributed to the extendable shared exponent strategy, which ensures accurate representation of outliers, preventing errors from their misrepresentation. When compared to BiE, which also addresses outlier handling, EES achieves comparable accuracy while offering notable advantages in energy efficiency. Specifically, in the Llama3.1-8B model, using the Hellaswag dataset, EES boosts accuracy by up to 0.55% and reduces perplexity by 0.56% on the LAMBADA_Standard dataset. These results highlight the effectiveness of EES in maintaining high accuracy and energy efficiency, even with slight accuracy losses on certain benchmarks.

Bit-width Space Exploration. Table 5 summarizes results from exploring the bit-width space with different EES parameter configurations. The optimal configuration includes a mantissa bit-width of 4 and extendable shared exponent widths of 5 for activations and 4 for weights. During this exploration, accuracy loss between adjacent configurations was constrained to be no greater than 1.00%. On average, EES improves energy efficiency by 13.6× compared to FP16. The bit-width exploration is conducted offline, allowing the optimal configuration to be selected for online inference without adding hardware overhead. For the activation values generated online, we can obtain estimates through statistical data analysis and used them as the prior information for the guideline of the offline exploration. Details of the exploration process across benchmarks are provided in Appendix A.3.

The optimal configuration is derived using Equation (14), which balances precision and hardware overhead. The balance factor α plays a key role in determining the trade-off between accuracy and energy efficiency. In this study, $\alpha = 0.2$ is used as

Benchmark	OPT-6.7B				OPT-13B				Llama2-7B			
Config	W2A2	W3A3	W5A5	W6A6	W2A2	W3A3	W5A5	W6A6	W2A2	W3A3	W5A5	W6A6
$<l_{E_{SA}}, l_{E_{SW}}>$	$<5,5>$	$<6,4>$	$<5,4>$	$<6,5>$	$<5,5>$	$<5,4>$	$<5,4>$	$<6,5>$	$<5,5>$	$<6,4>$	$<5,4>$	$<5,5>$
Accuracy (%)	43.84	46.78	47.92	47.96	44.89	46.52	48.01	48.11	43.27	48.47	49.74	49.87
Energy Efficiency	25.33×	17.74×	11.03×	8.02×	25.33×	17.75×	11.03×	8.02×	25.33×	17.74×	11.02×	8.03×
Benchmark	Llama2-13B				Llama3.1-8B							
Config	W2A2	W3A3	W5A5	W6A6	W2A2	W3A3	W5A5	W6A6				
$<l_{E_{SA}}, l_{E_{SW}}>$	$<5,5>$	$<5,5>$	$<5,4>$	$<6,5>$	$<5,5>$	$<5,4>$	$<5,4>$	$<6,5>$				
Accuracy (%)	43.78	48.76	49.57	49.76	45.11	50.55	52.10	52.22				
Energy Efficiency	25.33×	17.74×	11.03×	8.02×	25.33×	17.75×	11.03×	8.02×				

Table 6: Optimal accuracy and energy efficiency (normalized with FP16) results are based on a bit-width space exploration conducted under the optimal extended shared exponent bit width setting with different mantissa bit width on benchmarks and Hellaswag dataset. The optimal results of bit-width space exploration with different mantissa on ARC_challenge, LAMBADA_OpenAI and LAMBADA_Standard datasets are shown in Appendix A.3.

Method	Config	Throughput	Memory Efficiency	Energy Efficiency	Area Efficiency
FP16	-	1.00×	1.00×	1.00×	1.00×
Tender	W4A4	3.84×	4.00×	15.36×	14.21×
FIGNA	W4A4	3.58×	3.82×	13.68×	17.61×
BiE	W4A4	3.51×	2.80×	9.83×	11.85×
EES	W4A4	**3.56**×	**3.82**×	**13.60**×	**16.73**×

Table 7: Hardware efficiency of the size PE array with different numerical representations compared with FP16. We highlight the hardware efficiency of EES.

Method	Config	Hellaswag	arc_challenge
FP	-	52.16%	51.19%
BFP	W4A4	50.80%	50.14%
EES-EEE	W4A4	51.71%	50.82%
EES	W4A4	51.56%	50.74%

Table 8: Effect of shared exponent extending and embedding operations. “EES-EEE" denotes EES under the situation of removing the embedding.

the equilibrium value, with sensitivity analysis on α provided in Appendix A.4.

Table 6 further presents the optimal extended shared exponent bit-widths and corresponding results across benchmarks under various mantissa bit-widths. For a fixed mantissa bit-width, changes in extendable shared exponent widths have minimal impact on energy efficiency. For instance, in the W3A3 configuration, shifting from an extended exponent combination of $<6,4>$ to $<5,4>$ reduces energy efficiency by only 0.01%. Thus, with a fixed mantissa bit-width, selecting the optimal extended exponent width is driven mainly by accuracy considerations. By systematically identifying the best configuration for both mantissa and extended shared exponents, EES ensures a balanced trade-off between precision and energy efficiency.

Hardware Efficiency. Table 7 highlights the normalized hardware performance of EES compared to baseline. EES achieves remarkable improvements across key metrics: throughput is increased by 3.6×, memory efficiency by 3.8×, energy efficiency by 13.6×, and area efficiency by 16.7×. These gains stem from the adoption of low-bit-width representations and innovative extendable exponent strategy. When compared to Tender, EES shows a slight decrease in energy efficiency (0.9×), primarily due to the added complexity of EES computations compared to simpler integer multiplication. Against FINGA, EES delivers comparable throughput and efficiency but outperforms in accuracy. Relative to BiE, EES improves both energy and area efficiency by 1.4×, as BiE's use of an additional flag bit and dual-exponent mechanism incurs higher storage and computational costs.

Impact of Extendable Exponent Embedding. As shown in Table 8, removing the embedding operation for the extendable exponent leads to an average accuracy drop of only 0.12%. This indicates that embedding the extendable exponent into the last bit of partial mantissas minimally impacts accuracy while significantly reducing storage overhead. This result underscores the effectiveness of EES in balancing precision and hardware efficiency.

6 Conclusion

In this paper, we propose an Extendable Exponent Sharing (EES) method to overcome the limitations of existing low-bit quantization techniques. EES enhances dynamic range capture by embedding extendable exponent bits into lower-order bits of mantissa, without introducing additional overhead. To further optimize performance, we design a spatial exploration strategy to determine optimal bit-width configuration for both extendable exponent and mantissa. Comprehensive evaluation show that EES outperforms representative baseline methods in terms of both accuracy and energy efficiency.

7 Limitations

While EES offers an extended dynamic range, its accuracy can be affected by replacement errors when part of the mantissa is embedded within the extendable exponent bits. Furthermore, the optimal configuration of the extendable exponent and mantissa during the EES bit-width exploration can vary depending on the specific benchmarks or datasets considered. Currently, the exploration process involves searching the entire model, rather than tailoring it to individual layers, which both increases the exploration complexity and slows down the overall process. Future work will explore more efficient search strategies to accelerate this process.

On the hardware front, EES has shown improvements in both computational and energy efficiency. However, it may necessitate custom-designed computing units, encoders, and decoders, potentially adding extra overhead and complexity compared to conventional hardware designs. Future efforts will aim to refine the EES approach and address these challenges.

8 Acknowledgments

This work was supported in part by National Natural Science Foundation of China under Grants 62302102, in part by Guangdong Basic and Applied Basic Research Foundation (2023A1515012844, 2025A04J5208).

References

Ayon Basumallik, Darius Bunandar, Nicholas Dronen, Nicholas Harris, Ludmila Levkova, Calvin McCarter, Lakshmi Nair, David Walter, and David Peter Widemann. 2022. Adaptive block floating-point for analog deep learning hardware. *ArXiv*, abs/2205.06287.

Hanane Benmaghnia, Matthieu Martel, and Yassamine Seladji. 2022. Code generation for neural networks based on fixed-point arithmetic. *ACM Transactions on Embedded Computing Systems (TECS)*.

Chung-Chi Chen, Hiroya Takamura, Ichiro Kobayashi, and Yusuke Miyao. 2023. Improving numeracy by input reframing and quantitative pre-finetuning task. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 69–77.

Bita Darvish Rouhani, Daniel Lo, Ritchie Zhao, Ming Liu, Jeremy Fowers, Kalin Ovtcharov, Anna Vinogradsky, Sarah Massengill, Lita Yang, Ray Bittner, et al. 2020. Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point. *Advances in neural information processing systems*, 33:10271–10281.

Bita Darvish Rouhani, Ritchie Zhao, Venmugil Elango, Rasoul Shafipour, Mathew Hall, Maral Mesmakhosroshahi, Ankit More, Levi Melnick, Maximilian Golub, Girish Varatkar, et al. 2023. With shared microexponents, a little shifting goes a long way. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–13.

Hongxiang Fan, Shuanglong Liu, Zhiqiang Que, Xinyu Niu, and Wayne Luk. 2021. High-performance acceleration of 2-d and 3-d cnns on fpgas using static block floating point. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):4473–4487.

Jaeyong Jang, Yulhwa Kim, Juheun Lee, and Jae-Joon Kim. 2024. Figna: Integer unit-based accelerator design for fp-int gemm preserving numerical accuracy. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 760–773. IEEE.

Kari Kalliojarvi and Jaakko Astola. 1996. Roundoff errors in block-floating-point systems. *IEEE transactions on signal processing*, 44(4):783–790.

Callie Y Kim, Christine P Lee, and Bilge Mutlu. 2024. Understanding large-language model (llm)-powered human-robot interaction. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pages 371–380.

Hyoukjun Kwon, Prasanth Chatarasi, Vivek Sarkar, Tushar Krishna, Michael Pellauer, and Angshuman Parashar. 2020. Maestro: A data-centric approach to understand reuse, performance, and hardware cost of dnn mappings. *IEEE micro*, 40(3):20–29.

Jungi Lee, Wonbeom Lee, and Jaewoong Sim. 2024. Tender: Accelerating large language models via tensor decomposition and runtime requantization. pages 1048–1062.

Seunghyun Lee, Jeik Choi, Seockhwan Noh, Jahyun Koo, and Jaeha Kung. 2023. Dbps: Dynamic block size and precision scaling for efficient dnn training supported by risc-v isa extensions. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Fangxin Liu, Ning Yang, Haomin Li, Zongwu Wang, Zhuoran Song, Songwen Pei, and Li Jiang. 2024. Spark: Scalable and precision-aware acceleration of neural networks via efficient encoding. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 1029–1042. IEEE.

Yun-Chen Lo, Tse-Kuang Lee, and Ren-Shuo Liu. 2023. Block and subword-scaling floating-point (bsfp): An

efficient non-uniform quantization for low precision inference. In *The Eleventh International Conference on Learning Representations*.

Yun-Chen Lo and Ren-Shuo Liu. 2023. Bucket getter: A bucket-based processing engine for low-bit block floating point (bfp) dnns. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 1002–1015.

Tan John Chong Min. 2023. An approach to solving the abstraction and reasoning corpus (arc) challenge.

Pravin Nair, Ruturaj G. Gavaskar, and Kunal Narayan Chaudhury. 2021. Fixed-point and objective convergence of plug-and-play algorithms. *IEEE Transactions on Computational Imaging*, 7:337–348.

Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13.

Trong-Hieu Nguyen, Anh-Cuong Le, and Viet-Cuong Nguyen. 2024. Villm-eval: A comprehensive evaluation suite for vietnamese large language models. *ArXiv*, abs/2404.11086.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*.

Seock-Hwan Noh, Jahyun Koo, Seunghyun Lee, Jongse Park, and Jaeha Kung. 2023. Flexblock: A flexible dnn training accelerator with multi-mode block floating point support. *IEEE Transactions on Computers*, 72(9):2522–2535.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and R. Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *ArXiv*, abs/1606.06031.

Yubin Qin, Yang Wang, Zhiren Zhao, Xiaolong Yang, Yang Zhou, Shaojun Wei, Yang Hu, and Shouyi Yin. 2024. Mecla: Memory-compute-efficient llm accelerator with scaling sub-matrix partition. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 1032–1047. IEEE.

Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. 2022. A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Transactions on Intelligent Systems and Technology*, 14:1–50.

Xuan Shen, Peiyan Dong, Lei Lu, Zhenglun Kong, Zhengang Li, Ming Lin, Chao Wu, and Yanzhi Wang. 2024. Agile-quant: Activation-guided quantization for faster inference of llms on the edge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18944–18951.

Zhourui Song, Zhenyu Liu, and Dongsheng Wang. 2018. Computation error analysis of block floating point arithmetic oriented convolution neural network accelerator design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Mengxia Yu, De Wang, Qi Shan, and Alvin Wan. 2024. The super weight in large language models. *arXiv preprint arXiv:2411.07191*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Shulin Zeng, Jun Liu, Guohao Dai, Xinhao Yang, Tianyu Fu, Hongyi Wang, Wenheng Ma, Hanbo Sun, Shiyao Li, Zixiao Huang, et al. 2024. Flightllm: Efficient large language model inference with a complete mapping flow on fpgas. In *Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 223–234.

Cheng Zhang, Jianyi Cheng, Ilia Shumailov, George A Constantinides, and Yiren Zhao. 2023. Revisiting block-based quantisation: What is important for sub-8-bit llm inference? *arXiv preprint arXiv:2310.05079*.

Hengrui Zhang, August Ning, Rohan Baskar Prabhakar, and David Wentzlaff. 2024. Llmcompass: Enabling efficient hardware design for large language model inference. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 1080–1096. IEEE.

Sai Qian Zhang, Bradley McDanel, and HT Kung. 2022a. Fast: Dnn training under variable precision block floating point with stochastic rounding. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 846–860. IEEE.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022b. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Youpeng Zhao, Di Wu, and Jun Wang. 2024. Alisa: Accelerating large language model inference via sparsity-aware kv caching. *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 1005–1017.

Lancheng Zou, Wenqian Zhao, Shuo Yin, Chen Bai, Qi Sun, and Bei Yu. 2024. Bie: Bi-exponent block floating-point for large language models quantization. In *International Conference on Machine Learning*.

A Appendix

A.1 Hardware Overhead

Table 9 presents a comparison of the encoding overhead for converting original FP format data to INT8, INT4, and BFP8, as well as the decoding overhead for restoring data to its original format. Taking the FP16-INT8 type as an example, it represents the overhead of encoding FP16 into INT8. From Table 9, it can be observed that, compared to conversion to other simpler data formats, the overhead of the encoder and decoder in the EES accelerator in terms of power consumption and area is relatively low, and can be considered negligible compared to the overall overhead of the accelerator.

Type	Component	Power [w]	area [mm^2]
FP16-INT8	Encoder	0.0025	0.00362
INT8-FP16	Decoder	0.0069	0.00484
FP16-INT4	Encoder	0.0015	0.00289
INT4-FP16	Decoder	0.0039	0.00281
FP16-BFP8	Encoder	0.0036	0.00301
BFP8-FP16	Decoder	0.0035	0.00272
FP16-EES	Encoder	0.0034	0.00255
EES-FP16	Decoder	0.0049	0.00359

Table 9: Comparison of Encoding and Decoding Costs with INT8, INT4, and BFP8.

A.2 Results of Accuracy and Perplexity

Table 10 demonstrates the experimental results of all baselines and our method on the more advanced LLM models DeepSeek_R1_Disstill_Qwen_1.5B and DeepSeek_R1_Distill_Llama8B. Consistent with the results observed in other LLM models, EES has improved in both accuracy and perplexity compared to baseline, while maintaining performance close to the original accuracy.

Table 11 presents a comparison of the best performance data of basic BFP and our method EES across five models. It can be observed that, compared to the best-performing configuration of basic BFP, EES shows significant improvements in both accuracy and perplexity, while demonstrating similar performance in terms of energy efficiency.

A.3 Bit-width Space Exploration Results

Table 12 presents the optical accuracy and energy efficiency results obtained through the EES bit-width search across different benchmarks and datasets. Consistent with the findings on the Hellaswag dataset, The optimal configuration includes a mantissa bit-width of 4 and extendable shared exponent widths of 5 for activations and 4 for weights. Table 13 further describes the optimal accuracy and energy efficiency results for each benchmark and dataset under different mantissa bit-width settings obtained through the bit-width space search. As observed in the Hellaswag dataset, increasing the mantissa bit-width improves accuracy but also leads to a decrease in energy efficiency.

Fig. 6 presents the accuracy and normalized energy efficiency results (compared to FP) for different mantissa bit width Lm using the Hellaswag dataset. It can be observed that for most benchmarks, when $Lm \geq 4$, EES can maintain relatively high accuracy, but the energy efficiency significantly decreases. For example, on the Llama3.1-8B benchmark, increasing Lm from 4 to 6 only improves accuracy by 0.32%, while energy efficiency drops by $1.7\times$. When $Lm > 4$, energy efficiency improves, but this comes with a significant loss in accuracy. Therefore, the performance across different benchmarks provides valuable guidance for selecting the optimal mantissa bit width that balances accuracy and energy efficiency effectively.

Fig. 7 shows the changes in perplexity and normalized energy efficiency for different values of Lm on the LAMBADA_OpenAI dataset. Consistent with the findings on the Hellaswag dataset, increasing Lm in EES leads to a decrease in perplexity, but energy efficiency also declines. This is primarily due to the use of different mantissa bit width in the EES implementation.

Fig. 8 illustrates the results of the extendable exponent bit-width search under a fixed mantissa bit-width using the Hellaswag dataset on the Llama3.1-8B model. As shown in Fig. 8 (a), when the mantissa bit-width is 6, the optimal extendable exponent configuration is $<4, 3>$. Fig. 8 (b) shows that when the mantissa bit-width is 4, the optimal extendable exponent configuration is $<2, 1>$. Since the impact of the extendable exponent on energy consumption is negligible when the mantissa bit-width is the same, we primarily determine the optimal extendable exponent bit-width configuration based on accuracy.

A.4 Determination of Balance Factor α

Fig. 9 shows the impact of different balance factors α on accuracy and energy efficiency across multi-

Tasks		FP	Tender	FIGNA	BiE	EES
DeepSeek_R1_ Disstill_Qwen_1.5B	Hellaswag(↑)	41.78%	39.01%	40.31%	41.09%	41.57%
	ARC_challenge(↑)	24.38%	23.00%	24.01%	24.12%	24.23%
	lambada_openai(↓)	6.95	20.92	8.18	8.04	7.76
	lambada_standard(↓)	10.65	42.17	12.54	12.31	11.98
Tasks		FP	Tender	FIGNA	BiE	EES
DeepSeek_R1_ Distill_Llama8B	Hellaswag(↑)	48.99%	46.31%	48.14%	48.46%	48.61%
	ARC_challenge(↑)	40.61%	39.35%	39.67%	40.30%	39.94%
	lambada_openai(↓)	5.51	9.08	6.29	6.94	5.87
	lambada_standard(↓)	9.18	12.79	11.06	10.37	9.94

Table 10: Accuracy/perplexity comparison between EES and baselines. (↓) indicates that lower perplexity is better for the datasets, while (↑) indicates that higher accuracy is better.

Model	method	Hellaswag(↑)	ARC_challenge(↑)	lambada_openai(↓)	lambada_standard(↓)	energy efficiency
OPT-6.7B	BFP	46.61%	29.41%	5.83	6.90	13.62×
	EES	47.88%	29.89%	5.11	5.68	13.60×
OPT-13B	BFP	47.40%	31.72%	5.63	7.08	13.62×
	EES	47.88%	32.58%	5.01	6.48	13.60×
LLama2-7B	BFP	48.80%	43.71%	5.61	6.67	13.62×
	EES	49.34%	44.45%	4.44	5.60	13.60×
LLama2-13B	BFP	49.11%	47.83%	3.91	4.42	13.62×
	EES	49.49%	48.31%	3.08	3.83	13.60×
LLama3.1-8B	BFP	50.87%	49.50%	3.98	5.31	13.62×
	EES	51.73%	50.67%	3.06	4.16	13.60×

Table 11: The comparison between the best performance results of the basic BFP and EES, across five models is as follows, while the energy efficiency is expressed as a multiple relative to FP16.

Benchmark	OPT-6.7B			OPT-13B			Llama2-7B			Llama2-13B			Llama3.1-8B		
	ARC_challenge	LAMBADA_OpenAI	LAMBADA_Standard	ARC_challenge	LAMBADA_OpenAI	LAMBADA_Standard	ARC_challenge	LAMBADA_OpenAI	LAMBADA_Standard	ARC_challenge	LAMBADA_OpenAI	LAMBADA_Standard	ARC_challenge	LAMBADA_OpenAI	LAMBADA_Standard
Config	W4A4	W4A4	W4A4	W4A4	W4A4	W4A4	W4A4	W4A4	W4A4	W3A3	W3A3	W3A3	W4A4	W4A4	W4A4
$<l_{E_{SA}}, l_{E_{SW}}>$	<5, 4>	<5, 4>	<5, 4>	<5, 4>	<5, 4>	<5, 4>	<5, 5>	<5, 5>	<5, 5>	<5, 4>	<5, 4>	<5, 4>	<6, 5>	<6, 5>	<6, 5>
Accuracy(%) / perplexity	29.89	5.11	5.68	32.78	5.01	6.48	44.53	4.41	5.56	47.54	3.35	3.87	50.72	3.02	4.07
Energy Efficiency	13.60×	13.60×	13.60×	13.60×	13.60×	13.60×	13.59×	13.59×	13.59×	13.60×	13.60×	13.60×	13.58×	13.58×	13.58×

Table 12: The optimal accuracy, perplexity and energy efficiency (normalized with FP16) results are based on a quantization space exploration conducted under different mantissa and extendable exponent bit width setting on benchmarks evaluated using the ARC_challenge, LAMBADA_OpenAI and LAMBADA_Standard datasets. $<l_{E_{SA}}$, $l_{E_{SW}}>$ denotes the combination of activation and weight of the extended shared exponent bit width.

Benchmark	OPT-6.7B				OPT-13B				Llama2-7B				Llama2-13B				Llama3.1-8B			
Config	W2A2	W3A3	W5A5	W6A6	W2A2	W3A3	W5A5	W6A6	W2A2	W3A3	W5A5	W6A6	W2A2	W3A3	W5A5	W6A6	W2A2	W3A3	W5A5	W6A6
$<l_{E_{SA}}, l_{E_{SW}}>$	<5, 5>	<6, 4>	<5, 4>	<5, 4>	<5, 5>	<5, 4>	<5, 4>	<6, 5>	<5, 5>	<6, 4>	<5, 4>	<5, 5>	<5, 5>	<5, 5>	<5, 4>	<6, 5>	<5, 5>	<5, 4>	<5, 4>	<6, 5>
arc_challenge(↑)	25.11%	29.23%	30.12%	30.43%	30.19%	32.47%	32.83%	32.99%	33.68%	42.96%	44.52%	44.59%	42.23%	47.54%	48.65%	48.79%	38.11%	48.77%	52.88%	52.01%
LAMBADA_OpenAI (↓)	5.76	5.47	4.66	4.32	5.71	5.34	4.8	4.66	5.83	5.06	3.69	3.05	3.62	3.21	3.03	3.01	6.73	4.75	3.11	3.02
LAMBADA_Standard (↓)	6.77	6.21	5.99	5.37	6.98	6.67	6.90	5.87	7.36	5.33	4.56	5.29	4.05	3.87	3.83	3.74	8.61	4.98	4.22	4.16
Energy Efficiency	25.33×	17.74×	11.03×	8.02×	25.33×	17.75×	11.03×	8.02×	25.33×	17.74×	11.02×	8.03×	25.33×	17.74×	11.03×	8.02×	25.33×	17.75×	11.03×	8.02×

Table 13: The optimal accuracy and energy efficiency (normalized with FP16) results are based on a bit-width space exploration conducted under the optimal extended shared exponent bit width setting with different mantissa bit width on benchmarks evaluated using the ARC_challenge, LAMBADA_OpenAI and LAMBADA_Standard datasets. $<l_{E_{SA}}, l_{E_{SW}}>$ denotes the combination of activation and weight of the extended shared exponent bit width.

ple benchmarks. We selected six representative values ($\alpha \in \{0.015, 0.05, 0.2, 0.5, 1, 2\}$, ranging from 0 to 2) for sensitivity analysis at different levels of α. The results indicate that when α is between 0.015 and 0.2, the objective function tends to favor accuracy, whereas when α is between 0.5 and 2, it favors energy efficiency. Only when α is between 0.2 and 0.5, the objective function can achieve a better balance between accuracy and energy efficiency. Based on the results across all benchmarks, we ultimately chose α=0.2 as the equilibrium value between accuracy and energy efficiency.

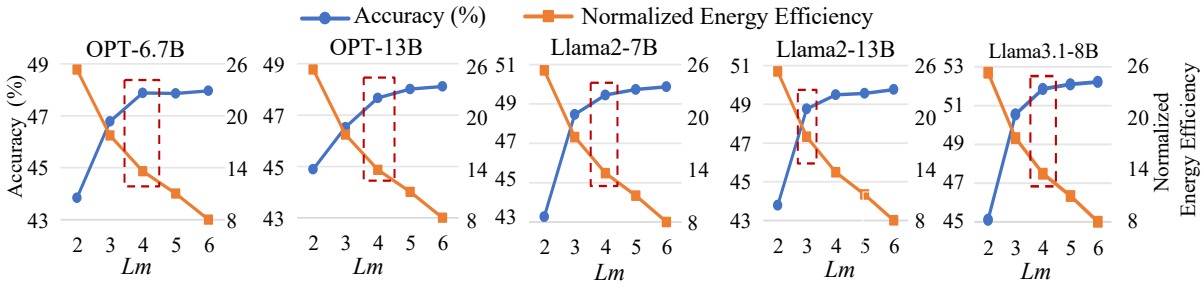

Figure 6: Bit-width space exploration results under different mantissa bit-width (Lm) on benchmarks evaluated using the Hellaswag dataset, with the optimal extended exponent bit-width settings. The red dashed box represents the optimal (Lm) configuration for benchmarks.

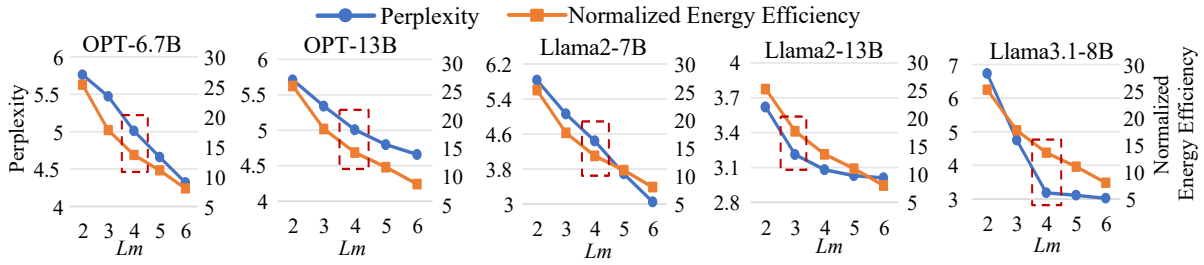

Figure 7: Bit-width space exploration results under different mantissa bit-width (Lm) on benchmarks evaluated using the LAMBADA_OpenAI dataset, with the optimal extended exponent bit-width settings. The red dashed box represents the optimal (Lm) configuration for benchmarks.

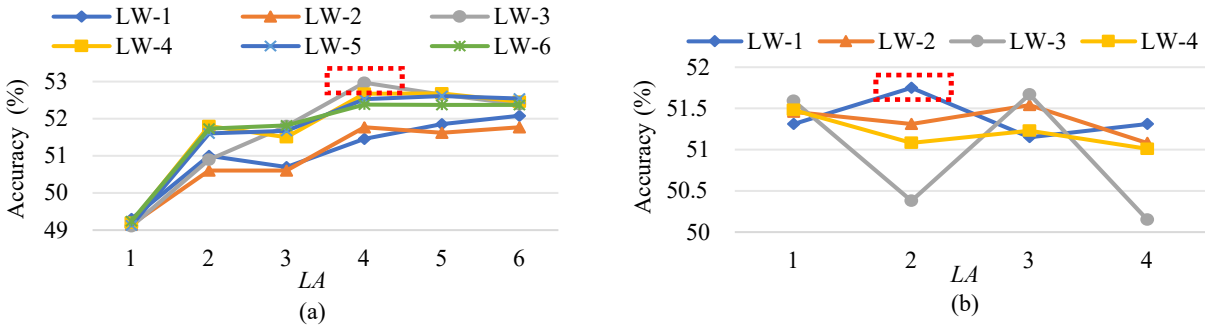

Figure 8: Bit-width space exploration results under different extendable exponent bit-width (LA, LW) on Llama3.1-8B evaluated using the Hellaswag dataset, with the same mantissa bit-width settings. (a) The original data format consists of a 6-bit mantissa and a 1-bit shared exponent. (b) The original data format consists of a 4-bit mantissa and a 3-bit shared exponent. LA and LW represent the extendable exponent bit width of activation and weight, respectively. The red dashed box represents the optimal extendable exponent bit-width configuration for benchmark.

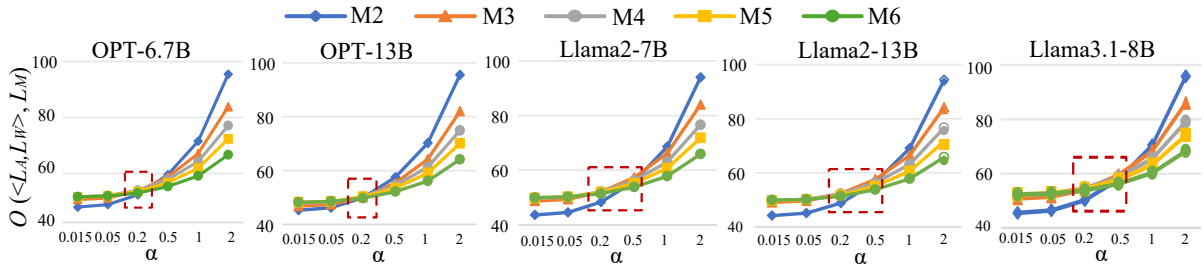

Figure 9: Comprehensive sensitivity analysis of different α values for accuracy and energy efficiency under different mantissa width on the benchmarks evaluated using the Hellaswag dataset. The red dashed box represents the optimal balance factor ($\alpha = 0.2$) for benchmarks.