FPE2M2: Approaching Lossless and Efficient Quantization with Native Floating Point

Ke Yi[1,2] **Jianwei Zhang**[2] **Zhiying Xu**[2] **Xinlong Yang**[2] **Yang Zhou**[2]
Minmin Sun[2] Zengke Liu[3] Tong Zhang[1] Junyang Lin[2] Jingren Zhou [2]

Abstract

Auto-regressive decoding is a memory-bound job, meaning decoding inference performance is limited by the bandwidth rather than the computational capabilities of the GPU. Weight-only quantization is a promising method to address the memory-bound limitations. Previous studies have followed one of two approaches. Some have exclusively studied integer quantization while ignoring the Gaussian distribution nature of LLMs' weights. Others have proposed non-uniform quantization, but incurred additional memory overhead due to lookup tables. In this work, we extend the floating-point standard to the ExMy quantization schema, which allocates x bits for the exponent and y bits for the mantissa to represent a number. In terms of runtime efficiency, we demonstrate that the conversion from ExMy to FP16 can be realized through register-level operations, which can get almost the same performance as INT5. In terms of quantization loss, we analyze that of different ExMy settings, where the E2M2 schema achieves an optimal balance, offering the highest efficiency with lossless accuracy. We further propose the FPE2M2 framework that supports lossless weight-only quantization inference and validate the FPE2M2 framework on Qwen and LLaMA Models across various modalities, such as text, image, and audio tasks, which achieves a faster inference speed while maintaining nearly lossless accuracy. [1]

1 Introduction

LLMs have demonstrated outstanding performance and potential for various tasks. However, the high cost of inference has limited their application in practical commercial scenarios. Specifically, the pipeline of LLM inference can be divided into two stages: pre-filling and decoding. Owing to the auto-regressive nature of LLMs, the decoding stage is afflicted by the I/O bottleneck, as it generates only one token at a time. These challenges motivate the research on model compression techniques aiming to alleviate the bottleneck.

Quantization is a prevalent model compression technique that represents weights with lower precision. When the compression ratio is high, for instance, 4-bit quantization, the loss in accuracy becomes non-negligible. Previous studies (Frantar et al., 2022; Lin et al., 2023) employ calibration sets to adjust the weights of LLMs for reduced quantization loss. Nevertheless, the quantized LLM may overfit the calibration set and exhibit poor performance on unseen data, as noted by (Williams and Aletras, 2024). From another perspective, those works primarily focus on integer quantization, which overlooks the characteristic that model weights adhere to a Gaussian distribution. (Dettmers et al., 2024) observes the characteristic and introduces a non-uniform quantization schema to decrease quantization loss further. However, the conversion from non-uniform quantized values to full precision values relies on lookup tables. Notably, accessing the lookup table can incur significant overhead during the memory-bound decoding stage.

This work extends the IEEE 754 floating-point standard to a low-bit schema. Specifically, a floating point with an x-bit exponent and a y-bit mantissa is denoted as ExMy. ExMy differs from the previous standard by replacing Infinity and Nan with regular numbers. Based on our observation, the conversion between ExMy and FP16 can be accomplished by register-level operations, obviating the need for lookup tables. Furthermore, we comprehensively analyze the quantization loss for various non-uniform quantization schemes, and FPE2M2 can reach the sweet spot of achieving lossless quantization of the minimum bit width. We comprehensively validate the Qwen, LLaMA, and DeepSeek Distillation Models framework on text,

[1]The work was conducted during Ke Yi's internship at Qwen Team. [1] South China Univesity of Technology [2] Alibaba Group [3] University of Chinese Academy of Sciences..

image, and audio tasks. We achieve a faster inference speed with negligible loss. Our contributions are as follows:

- We review the loss of different bit width allocation schemas and validate that FPE2M2 is the lossless quantization with minimum bit width.
- We propose a novel approach for rapid conversion from FPE2M2 to FP16 with only register-level operations, which is highly efficient in the memory-bound decoding stage.
- We further comprehensively validate the framework on mainstream LLMs for text, image, and audio tasks, achieving a faster inference with negligible accuracy loss.

2 Related Work

2.1 LLM Quantization

Quantization represents a practical methodology for reducing model size and accelerating inference. From a serving perspective, quantization can be categorized into weight-only and weight-activation quantization. Weight-activation quantization can accelerate computation by leveraging low-bit GEMM kernels suitable for compute-bound scenarios, namely the pre-filling stage. Qserve (Lin et al., 2024) further implement the A8W4KV4 and better accelerate. LLM.int8() (Dettmers et al., 2022) employs mixed INT8/FP16 decomposition to handle activation outliers. Subsequent work (Yuan et al., 2023) rearranges the channels to reduce the variance within one quantization group, further enhancing accuracy. Atom (Zhao et al., 2024) integrates the reorder technique and mixed INT4/INT8 precision to maintain accuracy and accelerate compared to the FP16 baseline. (Ashkboos et al., 2024; Liu et al., 2024; Yi et al., 2024) pairwise rotates the activation and weight to suppress outliers and maintain output equalization, enabling INT4 inference with well-smoothed activations. It should be noted that the above methods only serve the compute-bound pre-filling stage. Weight-only quantization is more suitable for the I/O bound decoding stage, as it employs low-bit representations for weight matrices, thereby saving memory movement. One effective way to reduce quantization error is shrinking the quantization range, i.e., sub-channel quantization. In the general case, quantization is performed on the channel level, which has a much less impact on the accuracy than the whole weight matrix. Applying sub-channel quantization could further reduce the quantization error, but it could result in a remarkable overhead. Orthogonal to Sub-channel quantization, GPTQ (Frantar et al., 2022) used Hessian-based error compensation to reduce quantization errors. AWQ (Lin et al., 2023) compressing weight quantization error according to the activation outliers. Those methods can achieve comparable accuracy compared with sub-channel quantization under the per-channel setting. However, they depend on the calibration dataset, which leads to a potential over-fitting problem (Williams and Aletras, 2024).

2.2 Non-uniform Quantization

Previous works focus on optimizing the quantization error for integer quantization. However, uniform quantization naturally causes more errors due to the Gaussian distribution of LLMs' weights. NF (Dettmers et al., 2024) propose a non-uniform quantization structure that fits the Gaussian distribution assumption well and achieves better accuracy than Integer quantization. AFPQ (Zhang et al., 2023) observes LLM's weights as asymmetric, and applying asymmetric quantization further improves the accuracy. LLM-FP4 (Liu et al., 2023) represents FP4 in both weight-only quantization and activation-weight quantization and validates the effectiveness of FP4. The above methods are all based on 4-bit quantization, which makes it hard to achieve a negligible accuracy drop. Moreover, some of those methods ignore the importance of conversion speed (Liu et al., 2023) or find it hard to optimize the conversion speed (Zhang et al., 2023).

3 Preliminaries and Motivation

3.1 Definition of ExMy

Extending the IEEE 754 standard, the ExMy is represented as a sign bit, exponent bit, which can be represented as:

$$\mathbf{X}_{\text{normal}} = (-1)^s 2^e (1 + \frac{d_1}{2^1} + \frac{d_2}{2^2} + \cdots + \frac{d_m}{2^m})$$
$$\mathbf{X}_{\text{subnormal}} = (-1)^s (\frac{d_1}{2^0} + \frac{d_2}{2^1} + \cdots + \frac{d_m}{2^{m-1}})$$

where $s \in \{0,1\}$ denotes the sign bit, and $d_i \in \{0,1\}$ is the mantissa bits. e denotes the exponent parts; the subnormal number has $e = 0$. To be noticed that the bias of ExMy's exponent part is 0, and the special values, i.e., 'Nan' and 'Inf', are

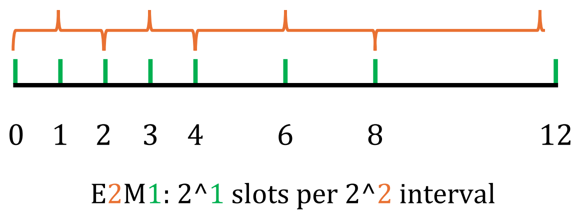


Figure 1: The positive part of E2M1.

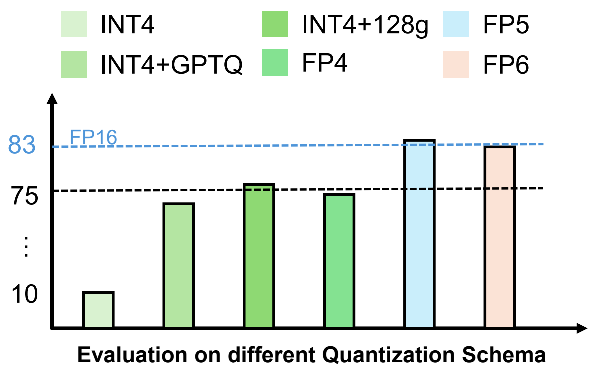


Figure 2: Preliminary ablation study on different quantization bit width. The evaluation is based on Qwen2.5-7B with the GSM8K benchmark. '128g' denotes the Sub-Channel setting with group size 128.

replaced by regular numbers. An example of E2M1 is shown in Figure 1.

3.2 Rethinking the Sweet Point of Quantization

Based on the binary system, representation with 2^n bits is more conducive for hardware implementation, e.g., 2-bit, 4-bit, 8-bit, 16-bit. FP16/BF16 is widely used for training and inference without accuracy loss. Recent works (Dettmers et al., 2022; Xiao et al., 2023) show that INT8/FP8 could also achieve lossless inference with higher efficiency than FP16/BF16. To further improve the efficiency, some works quantize the weights into 4-bit or 2-bit. The latter option suffers from severe accuracy loss, while the former balances efficiency and accuracy. As a result, previous works (Frantar et al., 2022; Lin et al., 2023) consider 4-bit as a sweet spot and attempt to restore the accuracy loss based on the 4-bit quantization.

As mentioned in Section 2.1, Sub-Channel Quantization and GPTQ are two popular methods for restoring the accuracy loss of INT4 quantization. Besides that, we also involve FP4 quantization, which adheres to the Gaussian distribution of LLMs' weights and brings less accuracy. However, as depicted in Figure 2, quantization utilizing the aforementioned methods still exhibits a significant accuracy loss (10%) compared to FP16. The accuracy loss becomes negligible when the bit width exceeds 5. In terms of providing negligible quantization error and attaining higher inference speed, 5-bit quantization represents the optimal choice and deserves more attention.

3.3 E2Mx Consistently Dominates under Different Bit Width

The issue is regarding allocating the bits for the exponent and mantissa portions for optimal performance. Under the assumption that the weight of LLMs obeys a Gaussian distribution, we examine the relationship between quantization error and Sigma of the distribution for different ExMy as shown in Figure 3. The result reflects two important observations:

- E2Mx consistently dominates other ExMy under the Sigma of LLMs' weight distribution.
- The trend of quantization error remains consistent across different bit widths.

To simplify the analysis among different quantization schemas, we scale both the quantization grids and weights to the range of [-1,1]. The sigma is collected on the scaled weights.

For the first observation, we provide a statistical analysis of the Sigma of different LLMs' weight distribution C, which is in accordance with the sweet point of E2Mx. For the second observation, we provide a simple proof in Appendix B that the quantization error of ExMy is four times that of ExM(y+1), approximating a linear relationship.

Based on the above analysis, the significance of E2M2 is highlighted, thereby motivating us to devise an efficient implementation of E2M2 quantization in the next section.

4 Method

4.1 Preliminaries

Floating Point Quantization For integer quantization, the high-precision floating point number is quantized through scaling and rounding, which can be formulated as:

$$\mathrm{s} = \frac{\mathbf{max}(|\mathbf{X}|)}{\mathbf{Q}_{max}}, \mathbf{Q} = \lfloor \frac{\mathbf{X}}{s} \rceil, \tag{1}$$

where $\mathbf{Q}_{max}$ represents the maximum value of the quantized number. However, the rounding operation could not be strictly applied to floating point

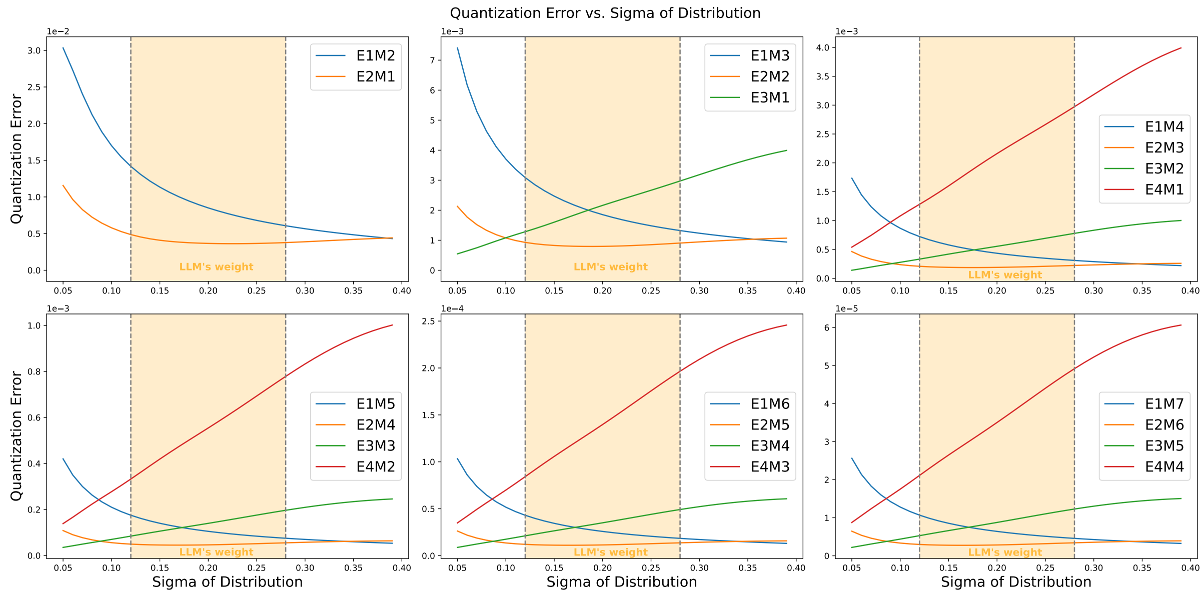


Figure 3: Preliminary analysis on the impact of different bit width allocation schemes. The quantization error is metricized by the L2 distance between the original and quantized weights. Quantization schemes with the same bit width are compared within the same figure range of the Sigma of weight results from mainstream LLMs.

quantization since the quantization slot is not uniform, as illustrated in Figure 1.

In order to efficiently determine the nearest quantized number for $\mathbf{X}$: Firstly, obtain the length of the interval $L = \mathbf{log}_2(\mathbf{max}(\mathbf{X}, 2))$. Secondly, obtain the length of slot $v = L/2^M$. Thirdly, rescale the interval and $\mathbf{X}$ to ensure that the slot $v = 1$ and $\mathbf{X}' = \mathbf{X}/v$. Fourthly, now the slot is uniform, round $\mathbf{X}'$ to the nearest integer and then scale back to the original scale. The above rounding process could be formulated as follows:

$$\mathbf{v} = \mathbf{log}_2(\mathbf{max}(\mathbf{X}, 2))/2^M$$
$$\mathbf{X}' = \mathbf{X}/\mathbf{v}, \mathbf{Q}' = \lfloor \mathbf{X}' \rceil, \mathbf{Q} = \mathbf{Q}' * \mathbf{v}$$

where M is the number of bits of the exponent part.

Dequantization Unlike the quantization process, the conversion from quantized ExMy to FP16 is performed in real-time, necessitating low time complexity. In integer dequantization, prior works have employed register-level operations to replace the low-throughput built-in conversion function, as shown in Figure 4 (a). This conversion depends on the following insight: For any unsigned integer $\mathbf{x}$ less than 8 bits, the bitwise OR operation $0x6400|\mathbf{x}$ yields a result that is numerically equivalent to $\mathbf{x} + 1024$.

Extending to the signed integer, $\mathbf{Q}_{max}$ is first added to the signed integers immediately after quantization to convert them into unsigned integers. Subsequently, the operation of subtracting $\mathbf{Q}_{max}$ is integrated into the final step, as depicted in Figure 4 (a).

For non-uniform quantization, the method above is not applicable. Previous research (Dettmers et al., 2024) has employed a lookup table for conversion, as shown in Figure 4 (b). However, accessing the lookup table is I/O intensive and inefficient for the memory-bound decoding stage.

4.2 Fast Conversion from ExMy to FP16

With the constraint of equation 3.1, we propose a fast conversion method from ExMy to FP16 based on numerical operations, as shown in Figure 4 (c). The conversion depends on the key insight:

$$\mathbf{X} = (\mathbf{Q} << 8) * 2^{15}$$

where $\mathbf{X}$ denotes the FP16 value before quantization, and $\mathbf{Q}$ denotes the quantized E2M2. The bias of the exponent part of FP16 is 15, whereas the bias of E2M2 is 0. Hence, we scale the shifted $\mathbf{Q}$ with 2^{15} in the equation above. It is worth noting that 1) The above conversion supports both normal and subnormal numbers; 2) The conversion assumes that $\mathbf{X}$ is positive. 3) The shifted $\mathbf{Q}$ is approximately 2^{-15}. Directly calculating on that without scaling can easily result in underflow; 4) It is not equivalent to directly setting some exponent bits to 1 to avoid underflow, since it would violate the definition of a subnormal number.

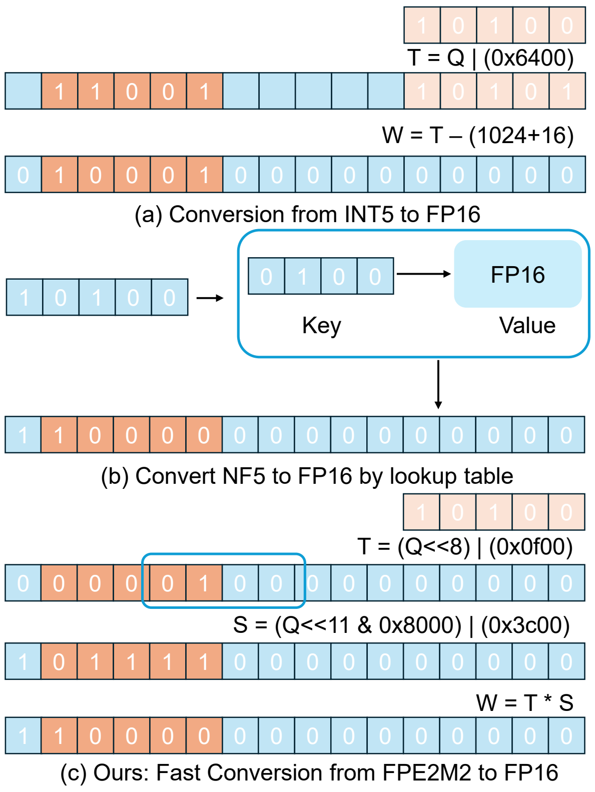

T = Q | (0x6400)
W = T – (1024+16)
(a) Conversion from INT5 to FP16
Key
FP16
Value
(b) Convert NF5 to FP16 by lookup table
T = (Q<<8) | (0x0f00)
S = (Q<<11 & 0x8000) | (0x3c00)
W = T * S
(c) Ours: Fast Conversion from FPE2M2 to FP16

Figure 4: Illustration of the conversion from quantized value back to FP16 value. (a) A fast conversion from INT5 to FP16 involves register-level operations, such as ADD and OR operations. (b) A naive conversion from non-uniform quantized value to FP16, including accessing the lookup table, which is I/O intensive. (c) The proposed conversion from E2M2 to FP16. Given that its definition is extended from the IEEE 754 standard, the conversion can be implemented by only register-level operations.

To efficiently handle both underflow and negative numbers, the sign bit is fused into a multiplication operation, as demonstrated in the final step of Figure 4 (c).

$$\mathbf{S} = (-1)^{s} 2^{15}$$
$$\mathbf{X} = ((\mathbf{Q} << 8) | (0x0f00)) * \mathbf{S}$$

where s is the sign bit.

4.3 System Level Optimization

The above description is based on single-element conversion. For real-time inference, it is necessary to optimize the bandwidth utilization by integrating multiple elements into INT32 and processing them in parallel in real time. To achieve storage of 5-bit E2M2 in INT32 without any blank space, 32 instances of 5-bit E2M2 are compressed into 5 INT32. Subsequently, the method of reconstructing FP16 from E2M2 will be demonstrated through careful design.

Algorithm 1: Dequantization from E2M2 to FP16

```
Input  : Quantized tensor $Q[5]$
Output : Dequantized tensor $DQ[16]$
1  Initialize array $e2m2[16]$;
2  for $i \leftarrow 0$ to $3$ do
3      $e2m2[i \cdot 4] \leftarrow (Q[i] << 8) \& 0x0f000f00$;
4      $e2m2[i \cdot 4 + 1] \leftarrow (Q[i] << 4) \& 0x0f000f00$;
5      $e2m2[i \cdot 4 + 2] \leftarrow (Q[i]) \& 0x0f000f00$;
6      $e2m2[i \cdot 4 + 3] \leftarrow (Q[i] >> 4) \& 0x0f000f00$;
7  end
8  for $i \leftarrow 0$ to $15$ do
9      $s \leftarrow Q[4] \& 0xf800f800$;
10     $s \leftarrow s \& 0xf3c003c00$;
11     $DQ[i] \leftarrow \mathrm{hmul2}(s, e2m2[i])$;
12     $Q[4] \leftarrow Q[4] >> 1$;
13 end
```

Sign Splitting To simplify the index calculations, the 5-bit E2M2 is divided into a 4+1 scheme; in this scheme, the first 4 bits are exponent bits and mantissa bits, i.e., E2M2, and the last bit is the sign bit. The 4-bit portion of E2M2 is stored in four INT32 continuously, and the remaining sign bits are stored in a single INT32.

Layout Organization In the mainstream system, FP16 (Half) is stored within INT32 as a pair, as shown in Figure 6 (a). This design results in efficient memory access and high memory bandwidth utilization. Under this design, it is essential to organize the data layout as shown in Figure 6 (b), which enables the parallel extraction of the pair weights. Through the masking (AND) operation, we could extract the E2M2 from the compressed INT32. Similarly, the masking operation is applied for the sign parts, and certain bits are set to 1 for the latter multiplication operation. The extracted E2M2 and sign are multiplied to obtain the final dequantized FP16. The entire layout is illustrated in Figure 6, and the pseudo-code is shown in Algorithm 1.

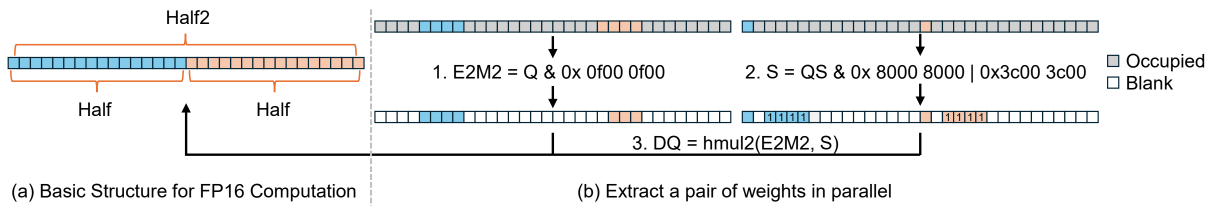

Half2
Half
Half
1. E2M2 = Q & 0x 0f00 0f00
2. S = QS & 0x 8000 8000 | 0x3c00 3c00
3. DQ = hmul2(E2M2, S)
Occupied
Blank
(a) Basic Structure for FP16 Computation
(b) Extract a pair of weights in parallel

Figure 5: Illustration of the low-level implementation. (a) The basic structure to store FP16/half value in the low-level computation. (b) To extract two quantized FPE2M2 values in parallel, we first split the sign part and E2M2 part and store them in separate INT32. For the pair of FP16 values in one INT32, their quantized E2M2/sign part should possess a gap of 16 bits.

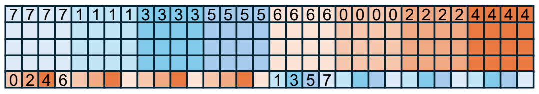


Figure 6: The overview of 32 quantized FPE2M2 value stored in five INT32.

5 Experiments

5.1 Models and Datasets

We conduct experiments on mainstream LLMs, including LLaMA families (Touvron et al., 2023) (LLaMA3-8B), Qwen families (Yang et al., 2024) (Qwen2.5-3B, Qwen2.5-7B, Qwen2.5-14B, Qwen2.5-72B) and Model distilled by DeepSeek-R1 (DeepSeek-R1-Qwen-14B, DeepSeek-R1-LLaMA3-8B). Quantization employs a per-channel symmetric scheme with a round-to-nearest (RTN) strategy. We evaluate the performance of the models on zero-shot Common Sense QA benchmarks(ARC-e, ARC-c (Clark et al., 2018), BoolQ (Clark et al., 2019), and OBQA (Mihaylov et al., 2018)), MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), Chinese benchmarks (Ceval(Huang et al., 2023)), Visual benchmarks (MME (Fu et al., 2023), POPE (Li et al., 2023), ChartQA (Masry et al., 2022), AI2D (Kembhavi et al., 2016)) and Audio benchmarks (MELD (Poria et al., 2019)).

5.2 On Text tasks

We conduct experiments on text tasks, including ArcEasy, ArcChallenge, BoolQ, OBQA, MMLU, GSM8K-Flex, GSM8K-Strict, and Ceval. We compare the performance of the models on text tasks with different quantization schemes. The results are shown in Table 1. Non-uniform quantization (NF5 and FPE2M2) consistently outperforms integer quantization (INT4, INT5) across all models and benchmarks. Notably, non-uniform quantization with 5-bit has achieved nearly lossless performance, which fluctuates around 0.3% compared with complete precision. It is worth noting that the conversion from NF5 to FP16 relies on the lookup table, which causes remarkable overhead, as shown in Section 5.5.

5.3 On Multi-modal tasks

We also compare the proposed approaches and the state-of-the-art methods on the multi-modal tasks, as presented in Table 2. We evaluate Visual tasks with Qwen2-VL-7B and audio functions with Qwen2-Audio-7B. FPE2M2 quantization consistently surpasses other integer quantization schemes and performs fluctuation around baseline performance. While INT4 has a 6% accuracy drop and INT5 has a 0.6% accuracy drop compared with complete precision. In some instances, e.g., Audio Taks, FPE2M2 even surpasses full precision performance by 2%.

5.4 Ablation Studies

Different Allocation for Exponent and Mantissa We conduct an ablation study on the allocation of exponent and mantissa for FPE2M2, as presented in 3. The results are consistent with the observation in 3.3, where the E2M2 achieves the best performance, and E4M0 has the worst.

Orthogonal Approach for Quantization Loss Optimization We conduct an ablation study that employs an orthogonal approach to alleviate the accuracy drop of quantization, as presented in 4. Sub-channel settings have a consistent performance gain on integer quantization, e.g., INT4 and INT5, while the gain is uncertain for non-uniform quantization, e.g., FPE2M1 and FPE2M2. The reason is that the Sub-channel settings have fewer elements within a quantization group, i.e.,. 128, violating the assumption that the quantization group obeys s Gaussian distribution. For GPTQ settings, we perform the algorithm with MMLU as a calibration

Table 1: Downstream accuracy (%) on text tasks including ArcEasy, ArcChallenge, BoolQ, OBQA, MMLU, GSM8K-Flex, GSM8K-Strict, Ceval. Each block is based on the same foundation model specified in the first row. The best accuracy is highlighted in red, and the second best is highlighted in blue.

Quant Method	ArcEasy	ArcChallenge	BoolQ	OBQA	MMLU	GSM8K-Flex	GSM8K-Strict	Ceval	Avg.
Qwen2.5-3B-Instruct	48.1	72.9	80.1	42.0	65.2	63.4	10.7	74.2	57.1
INT4	26.7	24.4	37.8	31.8	25.5	0	0	25.5	21.4
INT5	49.3	71.1	79.4	41	61.7	63.0	11.9	69.6	55.9
NF5	47.4	71.3	80.2	41.8	64.9	60.7	19.8	71.7	57.2
FPE2M2	47.1	69.7	78.2	41.6	64.9	64.8	19.1	74.1	57.4
Qwen2.5-7B-Instruct	55.0	81.3	86.3	48.8	72.2	82.9	76.3	79.6	72.7
INT4	47.7	66.5	80.4	43.2	64.7	11.3	6.9	67.4	48.5
INT5	54.7	79.0	85.5	47.6	71.3	75.9	68.9	78.3	70.1
NF5	55.1	80.2	86.1	47.8	71.7	82.4	77.6	78.9	72.5
FPE2M2	55.2	80.3	86.4	48.2	71.6	83.2	80.1	78.1	72.9
Qwen2.5-14B-Instruct	62.4	81.5	89.6	47.4	78.8	50.9	79.1	83.8	71.6
INT4	54.3	76.0	84.8	44.8	74.4	44.5	65.9	77.5	65.3
INT5	59.7	79.5	87.9	46.0	78.2	39.7	77.4	82.7	68.9
NF5	60.9	80.2	87.8	46.4	78.6	50.5	81.5	82.8	71.1
FPE2M2	62.5	82.9	87.9	47.6	78.4	58	76.8	83.2	72.2
Qwen2.5-72B-Instruct	63.4	83.2	90.4	48.8	83.7	89.4	90.7	89.3	79.9
INT4	57.5	77.6	88.2	45.2	80.8	55.4	63.6	84.5	69.1
INT5	63.4	82.8	90.6	47.8	83.0	84.6	90.6	88.4	78.9
NF5	63.8	82.7	90.5	50.6	83.6	88.4	90.9	88.9	79.9
FPE2M2	64.5	83.2	90.8	49.4	83.4	87.7	90.3	89.6	79.8
LLaMA3-8B	53.24	77.74	80.9	44.8	62.05	50.8	50.57	47.4	58.4
INT4	42.2	67.4	75.1	40.8	47.1	8.8	5.1	33.1	39.9
INT5	50.2	78.5	81.4	46.0	59.6	39.6	39.3	43.8	54.8
NF5	53.0	78.5	81.5	45.2	61.4	46.4	45.8	45.5	57.2
FPE2M2	53.2	78.4	80.2	45.4	61.2	46.9	46.0	45.8	57.2
DPSK-DS-Qwen-14B	53.5	74.9	87.7	43.4	73.2	87.1	86.8	76.4	72.9
INT4	47.7	70.4	85.0	40.2	68.5	69.7	75.8	69.1	65.8
INT5	53.6	73.7	87.3	43.4	71.8	66.4	86.5	73.6	69.5
NF5	53.9	75.5	87.5	42.8	72.1	87.1	87.4	75.7	72.7
FPE2M2	54.3	75.3	87.6	43.6	72.8	86.5	85.8	76.1	72.7
DPSK-DS-Llama-8B	42.4	66.1	83.0	41.6	54.1	63.9	62.1	44.2	57.2
INT4	40.6	59.2	77.4	35	43.7	53.4	48.5	34.3	49.0
INT5	42.2	64.6	82.3	41.2	52.6	63.7	60.9	41.0	56.0
NF5	42.6	64.4	83.3	40.6	53.7	64.8	62.4	43.6	56.9
FPE2M2	42.7	64.7	83.4	42.8	53.3	64.9	62.0	42.7	57.1

Table 2: Downstream accuracy (%) on multi-modal tasks including POPE, MMMU, ChartQA, RealWorldQA, MME, AI2D, OCRBench, Meld_dev, Meld_test. We evaluate visual tasks with Qwen2-VL-7B, and audio tasks with Qwen2-Audio-7B.

Method	POPE	MMMU	ChartQA	AI2D	OCRBench	Avg.	MME	Meld_dev	Meld_test
FP16	88.4	50.8	81.7	80.3	80.9	76.4	2325.3	54.5	54.7
INT4	86.9	45.3	77.2	77.7	69.9	71.4	2248.5	47.8	50
INT5	88.4	49.8	80.8	79.8	80.7	75.9	2283.6	54.3	54.3
NF5	88.4	50.4	81.6	79.7	81	76.2	2313.9	54.4	54.3
FPE2M2	88.9	50.6	81.8	80.3	81.1	76.5	2321.0	55.6	55.9

set. It performs well in cases with large accuracy drops, i.e., INT4. However, the phenomenon of over-fitting to the calibration set is seen in cases that have a small accuracy drop, leading to an overall performance drop.

5.5 Efficiency Analysis

We evaluate the GEMM kernel with low-bit quantization on RTX 4070 Ti and H100, as shown in Figure 7. We implement INT4, INT5, and NF5 as baselines and take speedup compared with FP16 as

Table 3: Ablation study on the different ExMy quantization schemes with five bits.

Quant Method	**ArcEasy**	**ArcChallenge**	**BoolQ**	**OBQA**	**MMLU**	**GSM8K-Flex**	**GSM8K-Strict**	**Ceval**	**Avg.**
Qwen2.5-7B-Instruct	54.9	81.2	86.3	48.8	72.1	82.8	76.2	79.6	72.7
FPE1M3	54.7	79.0	85.5	47.6	71.3	75.9	68.9	78.3	70.1
FPE2M2	55.2	80.3	86.4	48.2	71.6	83.2	80.1	78.1	72.9
FPE3M1	53.3	79.3	84.6	47	70.9	79.8	69.8	77.8	70.3
FPE4M0	48.5	70.9	77.6	43.4	59.8	56.4	56.0	60.9	59.2

Table 4: Downstream accuracy (%) on text tasks including ArcEasy, ArcChallenge, BoolQ, OBQA, MMLU, GSM8K-Flex, GSM8K-Strict, Ceval. Each block is based on the same quantization schema specified in the first row.

Quant Method	**ArcEasy**	**ArcChallenge**	**BoolQ**	**OBQA**	**MMLU**	**GSM8K-Flex**	**GSM8K-Strict**	**Ceval**	**Avg.**
INT4	47.7	66.5	80.4	43.2	64.7	11.3	6.9	67.4	48.5
+Sub-Channel	53.1 ↑	77.7 ↑	84.6 ↑	47.8 ↑	70.0 ↑	76.8 ↑	69.1 ↑	75.9 ↑	69.4
+GPTQ	51.7 ↑	74.9 ↑	83.6 ↑	43.3 ↑	70.0 ↑	73.1 ↑	67.8 ↑	73.3 ↑	67.2
FPE2M1	55.7	80.5	84.7	47.0	69.6	74.2	68.9	75.4	69.5
+Sub-Channel	53.2 ↓	79.3 ↓	86.2 ↓	47.8 ↑	71.1 ↑	79.8 ↓	73.2 ↑	78.8 ↑	71.2
+GPTQ	52.7 ↓	78.8 ↓	86.1 ↓	46.0 ↓	70.0 ↑	79.1 ↓	75.6 ↑	76.3 ↑	70.6
INT5	54.7	79.0	85.5	47.6	71.3	75.9	68.9	78.3	70.1
+Sub-Channel	55.0 ↑	80.5 ↑	86.2 ↑	48.2 ↑	71.6 ↑	79.2 ↑	71.6 ↑	79.4 ↑	71.4
+GPTQ	54.0 ↓	80.4 ↑	85.4 ↓	47.5↓	71.6 ↑	82.1 ↑	72.7 ↑	78.1↓	71.5
FPE2M2	55.2	80.3	86.4	48.2	71.6	83.2	80.1	78.1	72.9
+Sub-Channel	55.0 ↓	80.0 ↓	86.5 ↑	48.0 ↓	71.9 ↑	81.6 ↓	79.0 ↓	79.6 ↓	72.7
+GPTQ	54.4 ↓	81.4 ↑	86.2 ↓	49.2 ↑	71.8 ↑	81.5 ↓	74.4 ↓	78.5 ↓	72.2

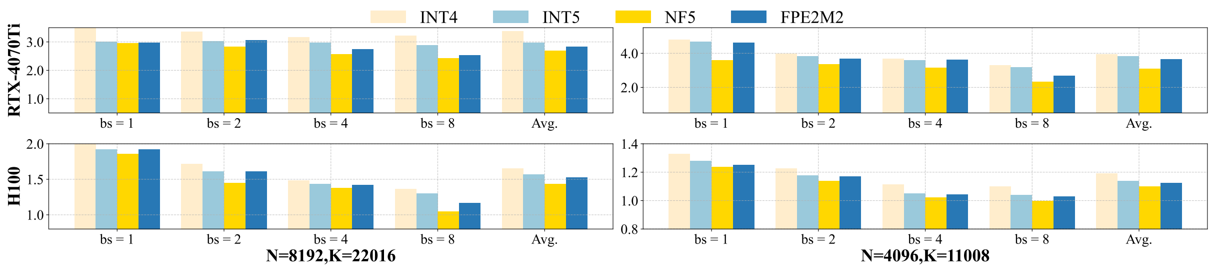


Figure 7: Same-batch throughput comparison between quantized inference and full precision inference on RTX-4070ti and H100.

the metric. Compared with INT5, FPE2M2 brings limited overhead, including a few more register-level multiplications and masking operations. NF5 brings noticeable overhead, including access to a lookup table, which is inefficient under memory-bound scenarios.

6 Conclusion

This work presents FPE2M2, extending the IEEE 754 floating-point standard to low-bit quantization and achieving lossless quantization with 5-bit FPE2M2. FPE2M2 is built on the assumption that weights of LLMs obey a Gaussian Distribution, which brings less quantization error compared with Integer quantization. Compared with other non-uniform quantizations, e.g., NF5, FPE2M2 can be easily implemented with negligible overhead and is generalized across various large language models (LLMs). Through a comprehensive analysis of different quantization schemes, we take 5-bit as a sweet spot for LLM quantization, where FPE2M2 can achieve near-lossless performance with negligible overhead.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. *Preprint*, arXiv:2404.00456.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *Preprint*, arXiv:2208.07339.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. 2023. MME: A comprehensive evaluation benchmark for multimodal large language models. *arXiv:2306.13394*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Preprint*, arXiv:2305.08322.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. *Preprint*, arXiv:1603.07396.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. *arXiv:2305.10355*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2024. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *Preprint*, arXiv:2405.04532.

Shih-yang Liu, Zechun Liu, Xijie Huang, Pingcheng Dong, and Kwang-Ting Cheng. 2023. Llm-fp4: 4-bit floating-point quantized transformers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, page 592–605. Association for Computational Linguistics.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024. Spinquant: Llm quantization with learned rotations. *Preprint*, arXiv:2405.16406.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *Preprint*, arXiv:2203.10244.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *Preprint*, arXiv:1810.02508.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Miles Williams and Nikolaos Aletras. 2024. On the impact of calibration data in post-training quantization and pruning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10100–10118, Bangkok, Thailand. Association for Computational Linguistics.

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.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Ke Yi, Zengke Liu, Jianwei Zhang, Chengyuan Li, Tong Zhang, Junyang Lin, and Jingren Zhou. 2024. Rotated runtime smooth: Training-free activation smoother for accurate int4 inference. *Preprint*, arXiv:2409.20361.

Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. 2023. Rptq: Reorder-based post-training quantization for large language models. *Preprint*, arXiv:2304.01089.

Yijia Zhang, Sicheng Zhang, Shijie Cao, Dayou Du, Jianyu Wei, Ting Cao, and Ningyi Xu. 2023. Afpq: Asymmetric floating point quantization for llms. *Preprint*, arXiv:2311.01792.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2024. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209.

A Limitations

Non-uniform quantization depends on the assumption that the weights of LLMs obey Gaussian Distribution, which is not always the case. For example, the weights of LLaMA3-70B have remarkable outliers, leading to a server accuracy drop for integer and non-uniform quantization based on per-channel settings. While GPTQ and Sub-Channel can address this issue, they still suffer from overfitting or inefficiency. As a result, the sweet point of quantization is still open to explore for the models with remarkable outliers.

B A simple proof of linear relationship between quantization error of ExMy and ExM(y+1)

First, recall that in ExMy representationwith exponent bits x and mantissa bits y. We proceed as follows::

$$\begin{aligned}\mathbf{X}_{\text{normal}} &= (-1)^s 2^e (1 + \frac{d_1}{2^1} + \frac{d_2}{2^2} + \cdots + \frac{d_m}{2^m}) \\ \mathbf{X}_{\text{subnormal}} &= (-1)^s (\frac{d_1}{2^0} + \frac{d_2}{2^1} + \cdots + \frac{d_m}{2^{m-1}})\end{aligned}$$

where $s \in \{0, 1\}$ is the sign bit, and $d_i \in \{0, 1\}$ is the mantissa bits. e denotes the exponent parts; the subnormal number has $e = 0$. Here we define the interval $L_i, i \in [2, 2^x]$, which starts from 2^{i-1} and ends at 2^i. L_1 is the first interval, starting from 0 and ending at 2. To be noticed that, the number of x defines the number of intervals, and the number of y defines the number of slots in each interval. Hence, compared with ExMy, ExM(y+1) just have two times more slots in each interval, and the length of each slot is 1/2 times smaller, as shown in 8. The quantization error can be calculated by:

$$E[Q] = \int_{-\infty}^{+\infty} p(x) * e(x) dx \tag{2}$$

where Q is the quantization error, $p(x)$ is the probability density function of the weight, and $e(x)$ is the L2 error.

Here we briefly approximate the p to be constant function $p(x) = 1$. For the first interval of E2M1, the quantization error is:

$$\begin{aligned}E[Q] &= \int_0^2 p(x) * e(x) dx \\ &= \int_0^2 1 * e(x) dx \\ &= 2 * \int_0^1 x(1-x) dx \\ &= \frac{1}{3}\end{aligned}$$

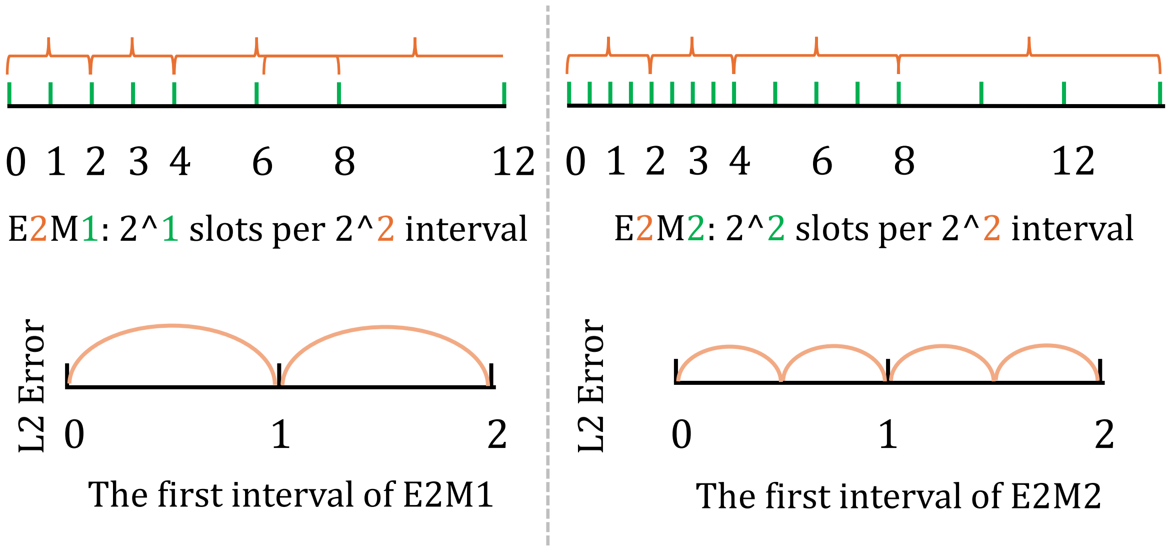
For the first interval of E2M2, the quantization error is:

$$\begin{aligned}E[Q] &= \int_0^2 p(x) * e(x) dx \\ &= \int_0^2 1 * e(x) dx \\ &= 4 * \int_0^{1/2} x(1/2 - x) dx \\ &= \frac{1}{12}\end{aligned}$$

The real quantization error with $p(x)$ being the normal distrition has been shown in the main text 3.

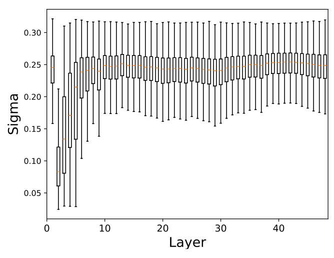
C Statistical analysis of Sigma of different LLMs' weight distribution

The quantization process entails two steps: Firstly, scale the weight to the maximum value of the quantization range. Secondly, round the weight to the nearest integer. To simply the analysis among different quantization schema, we scale the quantization grids to the range of $[-1, 1]$ and scale the weight to the range of $[-1, 1]$. Subsequently we collect the sigma of different LLMs' weight distribution after scaling, and show the results in 9. For cases where the model has a size larger than 14B, the sigma is relatively small in the very early layer. Apart from that, the sigma is typically in the range of 0.12 to 0.28, which accounts for 97% of the weight distribution. Moreover, 0.25 is the prevalent value for the sigma in most layers, which is the optimal point of E2Mx.

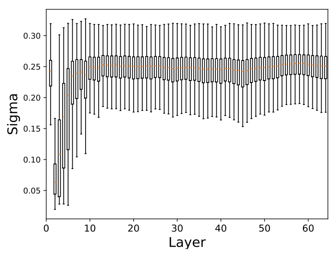


0 1 2 3 4 6 8 12
E2M1: 2^1 slots per 2^2 interval
0 1 2 3 4 6 8 12
E2M2: 2^2 slots per 2^2 interval
L2 Error
0 1 2
The first interval of E2M1
L2 Error
0 1 2
The first interval of E2M2

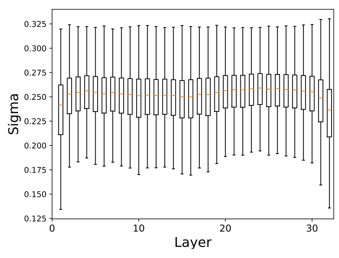
Figure 8: E2M1 and E2M2



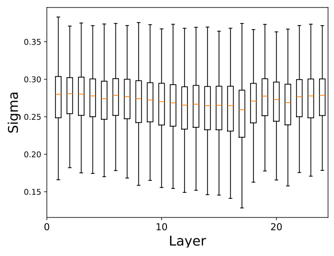
(a) DeepSeek-R1-Distill-Qwen-14B



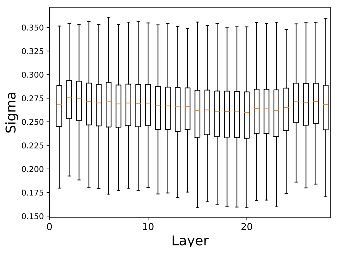
(b) DeepSeek-R1-Distill-Qwen-32B



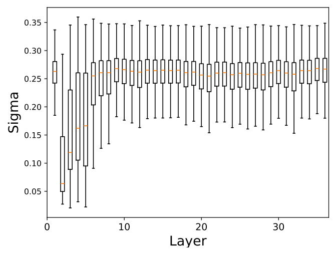
(c) Llama-3-8B



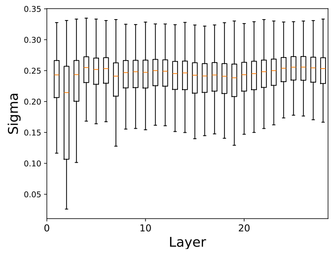
(d) Qwen-0.5B



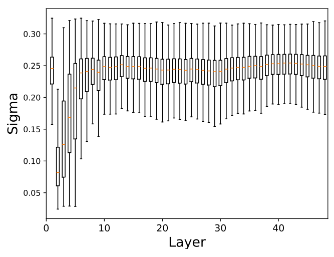
(e) Qwen-1.5B



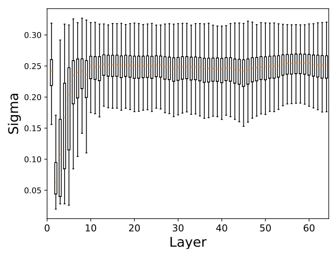
(f) Qwen-3B



(g) Qwen-7B



(h) Qwen-14B



(i) Qwen-32B

Figure 9: Sigma of different LLMs' weight distribution