Post Training Quantization of Large Language Models with Microscaling Formats

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have distinguished themselves with outstanding performance in complex language modeling tasks, yet they come with significant computational and storage challenges. This paper explores the potential of quantization to mitigate these challenges. We systematically study the combined application of three well-known post-training techniques, SmoothQuant, AWQ, and GPTQ, and provide a comprehensive analysis of their interactions and implications for advancing LLM quantization. We enhance the versatility of these techniques by enabling quantization to microscaling (MX) formats, expanding their applicability beyond their initial fixed-point format targets. We show that combining different PTQ methods enables us to quantize models to 4-bit weights and 8-bit activations using the MXINT format with negligible accuracy loss compared to the uncompressed baseline.

1 Introduction

Large Language Models (LLMs) have emerged as extremely powerful tools to comprehend and generate natural language. However, their intensive computational demand and energy consumption make widespread adoption of these models in everyday tasks to be challenging. One way to address these challenges is post-training quantization, a technique that involves reducing the precision of model parameters and/or activations from the original bit-width to formats with fewer bits. Quantization can significantly reduce the memory footprint and computational requirements of these models, making them more accessible and deployable on a wider range of hardware, including mobile devices and edge devices. However, previous work has shown that the activations of LLMs with more than 3B parameters are difficult to quantize due to the emergence of outliers with large magnitude, which leads to significant quantization errors and accuracy degradation (Dettmers et al., 2022). To address this issue, Xiao et al. proposed SmoothQuant, a quantization technique that smooths out the activation outliers by migrating the quantization difficulty from activations to weights with a mathematically equivalent transformation (Xiao et al., 2023). Lin et al., proposed AWQ, a weight only quantization algorithm that mitigates the quantization error by channel-wise scaling of the salient weights (Lin et al., 2023). Similarly, Frantar et al. proposed GPTQ, a scalable one-shot quantization method that utilizes approximate second-order information to quantize weights (Frantar et al., 2022). In this work, we systematically study the combined application of these three algorithms and provide a comprehensive analysis of their interactions and implications for advancing LLM quantization to various fixed-point and microscaling (MX) formats.

Microscaling format. The microscaling (MX) format for neural net computation was proposed by prior work, first as MSFP (Rouhani et al., 2020) and later subsumed by an emerging industry standard *microscaling formats* (Rouhani et al., 2023). Specifically, MXINT8 is a microscaling format that enables high-accuracy inference using half the memory footprint and twice the throughput of FP16. It is an emerging industry standard endorsed by Microsoft, AMD, Arm, Intel, Meta, and NVIDIA (Rouhani et al., 2023) and is already seeing adoption in today's hardware products, such as the Qualcomm cloud AI100 Accelerator (Qualcomm, 2024).

The MX format, as outlined in this paper, is characterized by three key components: 1) the scale factor data type, 2) the data type and precision of individual elements, and 3) the scaling block size. The scale factor is applied uniformly across a block of individual elements. This paper specifically focuses on MX formats employing the *INT* data type for individual elements, thus termed *MXINT*.

Notation. Throughout the paper we denote a microscaling (MX) format with scaling block size of b, *8-bit* shared scaling factor, and d bits per element by ***MXINTd-b***. For example, *MXINT6-64* represents an MX format with 6 bits per element, 8 bits shared exponent across 64 values within a block. Similarly, a fixed-point value with i integer bits and no fractional bits is denoted by ***INTi***.

Contributions.

1. We enhance SmoothQuant, AWQ, and GPTQ to support quantization to microscaling (MX) data formats, extending their compatibility beyond the initially targeted fixed-point formats in the proposed methods.
2. We study the interaction of SmoothQuant, AWQ, and GPTQ to quantize state-of-the-art models like Llama2 and Llama3 and show that SmoothQuant and GPTQ, as well as AWQ and GPTQ, are synergistic, especially at more restrictive bit-widths.

Algorithm 1 Enhanced GPTQ: Quantize $\mathbf{W}$ given inverse Hessian $\mathbf{H}^{-1} = (2\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}$, block size B_1, and micro-block size B_2.

Input: W // Weight matrix
Input: d_{row} // Row dimension of W
Input: d_{col} // Column dimension of W
Input: B_1 // Block size
Input: B_2 // Micro-block size
Input: H^{-1} // Hessian inverse information
Variable: E // Quantization error matrix
Output: Q // Quantized weight matrix
Initialize: $Q \leftarrow 0_{d_{row} \times d_{col}}$
Initialize: $E \leftarrow 0_{d_{row} \times d_{col}}$
Initialize: $H^{-1} \leftarrow Cholesky(H^{-1})^T$
for $i = 0, B_1, 2B_1, ...$ **do**
  for $j = i, i + B_2, i + 2B_2, ..., i + B_1 - 1$ **do**
    $k \leftarrow j + B_2$ *// helper index*
    $Q_{:,j:k} \leftarrow quant(W_{:,j:k})$
    $E_{:,j:k} \leftarrow (W_{:,j:k} - Q_{:,j:k})([H^{-1}]_{j:k,j:k})^{-1}$
    $W_{:,k:} \leftarrow W_{:,k:} - E_{:,j:k}[H^{-1}]_{j:k,k:}$
  end for
  $W_{:,i+B_1:} \leftarrow W_{:,i+B_1:} - E_{:,i:i+B_1}[H^{-1}]_{i:i+B_1,i+B_1:}$
end for
Return: Q

2 SmoothQuant

SmoothQuant (SQ) is a quantization method that targets both activations and weights of a model (Xiao et al., 2023). In this approach, the activation of a linear layer is scaled by a per-channel smoothing factor $s \in R^{C_i}$ to minimize quantization errors. Simultaneously, the weight of the layer is adjusted in the opposite direction to maintain the mathematical equivalence of the linear layer:

$$\mathbf{Y} = (\mathbf{X}\texttt{diag}(s)^{-1}) \cdot (\texttt{diag}(s)\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}} \quad (1)$$

In Equation 1, $\mathbf{X}$ is the original input activation with outliers, and $\hat{\mathbf{X}} = \mathbf{X}\texttt{diag}(s)^{-1}$ is the smoothed activation. To minimize the quantization error of the input activation, the smoothing factor is selected such that all channels of the smoothed input activation have the same maximum magnitude. Accordingly, s is set to:

$$s_j = \texttt{max}(|\mathbf{X}_j|), \quad j = 1, 2, ..., C_i \quad (2)$$

Where C_i is the number of input channels in the input activation and j corresponds to j^{th} input channel. Note that since the range of activations varies for different input samples, the maximum value of each channel is estimated using 128 calibration samples from the calibration dataset (see Section A for more details). By dividing the input activation by the the scaling factor of Equation 2, all channels of the scaled input activation would have the same range, making quantization of the scaled tensor to be very easy. However, this will migrate the difficulty of the quantization completely to the weight side of a linear layer. To address this issue, Xiao et al. proposed a scaling formula that balances the quantization difficulty of activations and weights:

$$s_j = \texttt{max}(|\mathbf{X}_j|)^{\alpha} / \texttt{max}(|\mathbf{W}_j|)^{1-\alpha}, \quad j = 1, 2, ..., C_i \quad (3)$$

Where α is a hyper-parameter that controls how much quantization difficulty we want to migrate from activations to weights. For quantization to the MX format using SmoothQuant, we directly calculated the SmoothQuant scaling factors, skipping the additional calibration phase required for quantization to fixed-point formats. For more details on the SmoothQuant algorithm refer to Xiao et al.'s work (Xiao et al., 2023).

3 AWQ

Activation-aware Weight Quantization (AWQ), is a weight-only quantization method for LLMs (Lin et al., 2023). In this algorithm, a small fraction (i.e., 0.1%-1%) of salient weight channels are scaled up to reduce their relative quantization error:

$$\mathbf{Y} = \mathbf{X}\mathbf{W} \approx \mathbf{X}\hat{\mathbf{W}} \approx (\mathbf{X}/s)(s\hat{\mathbf{W}}) \quad (4)$$

In Equation 4, s is a per-channel scaling factor for the salient weights. To determine the salient weights, AWQ refers to the activation distribution instead of the weight distribution, as weight channels corresponding to the outlier activations are more salient than other weights. The per-channel scaling factor is calculated using the following formula:

$$s = s_{\mathbf{X}}^{\alpha}, \quad \alpha \in [0, 1] \quad (5)$$

Act - Wgt bit-width	Format	Method	Llama2-7B	Llama2-13B	Llama3-8B
16-16	FP16, FP16	N/A	**5.12**	**4.57**	**5.54**
8-8	MXINT8-128, MXINT8-128	RTN	5.13	**4.58**	**5.55**
		GPTQ	5.13	**4.58**	**5.55**
		SmoothQuant	**5.12**	**4.58**	**5.55**
		AWQ	**5.12**	**4.58**	**5.55**
		SmoothQuant+	**5.12**	**4.58**	**5.55**
		AWQ+	**5.12**	**4.58**	**5.55**
	INT8, INT8	RTN	**5.15**	**4.60**	**5.62**
		GPTQ	**5.15**	**4.60**	**5.62**
		SmoothQuant	**5.15**	**4.60**	**5.62**
		AWQ	5.17	4.62	5.85
		SmoothQuant+	**5.15**	**4.60**	**5.62**
		AWQ+	5.17	4.62	5.84
8-4	MXINT8-128, MXINT4-128	RTN	5.55	4.82	7.13
		GPTQ	5.45	4.76	6.98
		SmoothQuant	5.60	4.93	7.05
		AWQ	5.43	4.77	6.37
		SmoothQuant+	5.48	4.84	6.51
		AWQ+	**5.37**	**4.73**	**6.16**
	INT8, INT4	RTN	5.91	4.97	8.44
		GPTQ	5.67	4.85	18.64
		SmoothQuant	6.34	5.56	9.13
		AWQ	5.61	4.85	7.33
		SmoothQuant+	5.78	5.12	7.32
		AWQ+	**5.53**	**4.80**	**7.06**

Table 1: Perplexity score on *WikiText-2-test* for the Llama2-7B, Llama2-13B, and Llama3-8B models, when quantized to fixed-point and MX formats using different post-training quantization techniques. Act, Wgt, and RTN denote activation, weight, and round to nearest, respectively. +: GPTQ weight quantization is used. We used *per-channel affine* quantization for the fixed-point formats.

Where $s_{\mathbf{X}}$ is the average magnitude of activation (per-channel), and α is a hyper-parameter which balances the protection of salient and non-salient channels. Similar to SmoothQuant, to make AWQ compatible with the MX format, we directly calculate the per-channel scaling factors, skipping the additional calibration phase required for fixed-point quantization. For more details on AWQ refer to Lin's et al. work (Lin et al., 2023)

4 GPTQ

GPTQ is a post-training quantization (PTQ) method that uses second-order Hessian information for weight quantization in LLMs (Frantar et al., 2022). It employs layer-wise quantization for each layer l in the network, seeking quantized weights $\hat{\mathbf{W}}_l$ that make the outputs $(\hat{\mathbf{W}}_l\mathbf{X}_l)$ closely approximate those of the original weights $(\mathbf{W}_l\mathbf{X}_l)$. In other words, GPTQ aims to find (Frantar et al., 2022):

$$\texttt{argmin}_{\hat{\mathbf{W}}_l} ||\mathbf{W}_l\mathbf{X}_l - \hat{\mathbf{W}}_l\mathbf{X}_l||_2^2 \tag{6}$$

To solve equation 6, GPTQ quantizes each row of the weight matrix, $\mathbf{W}$, independently, focusing on a single weight per row at a time. It consistently updates all not-yet-quantized weights to offset the error introduced by quantizing a single weight. Since the objective function in equation 6 is quadratic, its Hessian $\mathbf{H}$ can be calculated using the following formula, where F denotes the set of remaining full-precision weights:

$$\mathbf{H}_F = 2\mathbf{X}_F\mathbf{X}_F^T \tag{7}$$

Given $\mathbf{H}$, the next to be quantized weight, w_q, and the corresponding update of all remaining weights in F, δ_F, are given by the following formulas, where $\texttt{quant}(w)$ rounds w to the nearest quantized value (Frantar et al., 2022):

$$\begin{aligned} w_q &= \texttt{argmin}_{w_q} \frac{(w_q - \texttt{quant}(w_q))^2}{[\mathbf{H}_F^{-1}]_{qq}} \\ \delta_q &= -\frac{w_q - \texttt{quant}(w_q)}{[\mathbf{H}_F^{-1}]_{qq}} . (\mathbf{H}_F^{-1})_{:,q} \end{aligned} \tag{8}$$

For all rows of $\mathbf{W}$, GPTQ quantizes weights in the same order. This accelerates the process, as certain computations need to be performed only once for each column rather than once for each weight.

The GPTQ algorithm, as originally proposed, is designed for quantization to a fixed-point format. We have enhanced the algorithm to also support quantization to a *microscaling (MX) format*. Algorithm 1 provides pseudocode for the modified GPTQ, that enables MX quantization. Note that for quantizing $\mathbf{W}$ to a specific MX format, the micro-block size in the algorithm, B_2, should be a multiple of the block size of the MX format. For more details on the GPTQ algorithm refer to Frantar et al.'s work (Frantar et al., 2022).

5 Experiments

Setup. We evaluate the impact of the SmoothQuant, AWQ, and GPTQ techniques on quantization of Llama2 and Llama3 models. We employ various fixed-point and MX formats with different bit-widths for our assessment and report the perplexity of the quantized models on *WikiText-2* (Merity et al., 2016). Moreover, we study the impact of applying GPTQ, SmoothQuant, and AWQ individually, as well as the combined effects of GPTQ with AWQ and GPTQ with SmoothQuant. For more details on experiment setup refer to Section A.

Results. Table 1 illustrates perplexity of the quantized *Llama* models (Touvron et al., 2023; Meta, 2024) with three different sizes on WikiText-2-test using various MX and fixed-point formats. For all three models, aggressive quantization to small bit-widths penalizes the model performance, while quantizing to higher bit-widths has negligible effect on perplexity. For example, quantizing *Llama3-8B* to *MXINT8* preserves the baseline perplexity while quantizing to *MXINT4* increases perplexity by 29% to 7.13. Moreover, quantization results using different MX format delivers better perplexity compared to the fixed-point formats with the same bit-width. For instance, quantizing *Llama2-7B* to *INT4* increases perplexity to 5.91. Enabling AWQ, and GPTQ jointly, reduces it to 5.53, while using *MXINT4* and enabling AWQ and GPTQ we can achieve perplexity of 5.37. Additionally, we found that in all cases except for the quantization of both activations and weights to INT8, AWQ shows superior results compared to SmoothQuant. For the studied models and quantization formats, both SmoothQuant and GPTQ, as well as AWQ and GPTQ, are synergistic, an effect most prominent in more aggressive quantizations.

Similarly, we assess the impact of GPTQ, SmoothQuant, and AWQ on the quantization of the *Llama2, and Llama3* models (Touvron et al., 2023) using MX formats with the block size of *16*. We observe similar trends to those identified in this section. Detailed results of the experiment can be found in the Table 2 of the appendix.

6 Related Work

Model quantization methods. There are two primary categories of quantization techniques: Quantization-Aware Training (QAT), which leverages backpropagation to update quantized weights (Bengio et al., 2013; Choi et al., 2018; Nagel et al., 2021; Gholami et al., 2022), and Post-Training Quantization (PTQ), which typically requires no additional training. Quantization-aware training methods cannot easily scale up to quantize giant LLMs. Consequently, PTQ methods are commonly employed for quantizing LLMs (Jacob et al., 2018; Nagel et al., 2020; Wang et al., 2020; Hubara et al., 2021; Li et al., 2021; Deng et al., 2023). In this work, we studied the interaction of three PTQ methods, SmoothQuant (Xiao et al., 2023), AWQ (Lin et al., 2023), and GPTQ (Frantar et al., 2022).

Large Language Model quantization. With the recent open-source releases of language models like Llama (Touvron et al., 2023), researchers are developing cost-effective quantization methods to compress these models for inference: LLM.int8() identifies activation outliers in a few feature dimensions as a hindrance to the quantization of larger models, and proposes to preserve those dimensions in higher precision using a mixed INT8/FP16 decomposition (Dettmers et al., 2022). Similarly, SpQR (Dettmers et al., 2023) and OWQ (Lee et al., 2024) propose to retain outlier features that are difficult to quantize in full-precision, while AWQ (Lin et al., 2023) mitigates the quantization error for the outliers using grid-searched channel-wise scaling. Lee et al., explored the combined use of AWQ, SmoothQuant, and GPTQ for quantizing LLMs, focusing solely on fixed-point data types in their study (Lee et al., 2023).

7 Conclusion

To summarize, we demonstrated that for the studied models, quantizations using different MX formats deliver better perplexity compared to fixed-point formats with the same bit-width when the per-channel affine quantization scheme is employed. Particularly, for quantization to MXINT8, none of GPTQ, AWQ, or SmoothQuant are necessary to preserve the baseline accuracy. Notably, we found that for Llama2 and Llama3, when quantized to MX formats, AWQ is superior to SmoothQuant. Moreover, AWQ and GPTQ are synergistic, especially, with more aggressive quantization to 4-bit.

Throughout the paper, we have shown that by utilizing AWQ, and GPTQ and applying MX formats we can quantize the Llama2 and Llama3 models to 4-bit weights and 8-bit activations, with minimal perplexity degradation.

8 Limitations

With quantization of LLMs, we make the models accessible to more people, which generally comes with security risks, such as potential misuse for generating harmful content. This highlights the need for further investigation into responsible AI practices. On the technical side, due to space and computational resource constraints, we have only reported results for text generation with Llama2 and Llama3 models up to 13B parameters on the WikiText-2 dataset. Further investigation of broader models, datasets, and tasks remains for future work.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90% chatgpt quality.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. PACT: Parameterized clipping activation for quantized neural networks. *arXiv:1805.06085*.

Zihao Deng, Xin Wang, Sayeh Sharify, and Michael Orshansky. 2023. Mixed-precision quantization with cross-layer dependencies. *arXiv:2307.05657*.

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

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023. SpQR: A sparse-quantized representation for near-lossless llm weight compression. *arXiv:2306.03078*.

Tim Dettmers and Luke Zettlemoyer. 2023. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pages 7750–7774.

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

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. 2021. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pages 4466–4475.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2024. OWQ: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364.

Janghwan Lee, Minsoo Kim, Seungcheol Baek, Seok Hwang, Wonyong Sung, and Jungwook Choi. 2023. Enhancing computation efficiency in large language models through weight and activation quantization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14726–14739.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. 2021. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv:2102.05426*.

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:2306.00978*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv:1609.07843*.

Meta. 2024. Introducing Meta Llama 3: The most capable openly available LLM to date.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *arXiv:2106.08295*.

Qualcomm. 2024. Qualcomm Cloud AI 100 Accelerates Large Language Model Inference by 2x Using Microscaling (Mx) Formats.

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, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. 2023. Microscaling data formats for deep learning. *arXiv:2310.10537*.

Stability AI. 2023. Stable Beluga.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model.

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:2307.09288*.

Peisong Wang, Qiang Chen, Xiangyu He, and Jian Cheng. 2020. Towards accurate post-training network quantization via bit-split and stitching. In *International Conference on Machine Learning*, pages 9847–9856.

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.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Format	Method	Llama2-7B	Llama2-13B	Llama3-8B
A:FP16, W:FP16	N/A	**5.12**	**4.57**	**5.54**
A:MXINT8-16 W:MXINT8-16	RTN	**5.12**	4.58	**5.54**
	GPTQ	**5.12**	4.58	**5.54**
	SQ	**5.12**	**4.57**	**5.54**
	AWQ	**5.12**	4.58	**5.54**
	SQ+	**5.12**	**4.57**	**5.54**
	AWQ+	**5.12**	4.58	**5.54**
A:MXINT8-16 W:MXINT4-16	RTN	5.40	4.72	6.18
	GPTQ	5.41	**4.68**	5.93
	SQ	5.33	4.74	6.14
	AWQ	5.30	4.70	6.03
	SQ+	5.28	4.69	5.95
	AWQ+	**5.27**	**4.68**	**5.90**

Table 2: Perplexity score on *WikiText-2-test* for the Llama models, when quantized to MX formats with the block size of *16* using different post-training quantization techniques. *A*, *W*, SQ, and RTN denote activation, weight, SmoothQuant, and round to nearest, respectively. +: GPTQ weight quantization is used.

A Experiment Setup

Models. We evaluated various quantization methods using the Llama2, and Llama3 families (Touvron et al., 2023; Meta, 2024). These LLMs are widely accepted in the machine learning community for their superior performance compared to other open-source LLMs (Dettmers et al., 2022; Frantar et al., 2022; Xiao et al., 2023; Lin et al., 2023). Llama also serves as the foundation for many popular open-source models such as Alpaca (Taori et al., 2023), Vicuna (Chiang et al., 2023), and Stable Beluga (Stability AI, 2023).

Datasets. Following previous work (Dettmers et al., 2022; Xiao et al., 2023; Frantar et al., 2022; Lin et al., 2023; Dettmers and Zettlemoyer, 2023; Yao et al., 2022), we measured the perplexity of quantized language models on *WikiText-2* (Merity et al., 2016) as perplexity can stably reflect the performance of LLMs (Dettmers and Zettlemoyer, 2023; Lin et al., 2023). Unless otherwise stated, the *test* split of the dataset is used to evaluate the models.

Quantization formats. We evaluated models using different microscaling and fixed-point quantization formats. For the fixed-point quantization, we calibrated the models using 128 random input sentences from *WikiText-2-train* to estimate the dynamic range of activations. We utilized *MinMaxObserver* to find the range of activations, and calculated the zero-point and the scale parameters for the activations and weights in per-channel granularity levels. For the MXINT format, unless otherwise specified, the blocking dimension of a given tensor is the last dimension.

Activation smoothing. We calculated the per-channel scaling factor for activations and weights using the formula stated in Equation 1. As in the previous work, we consistently use a migration strength (α) value of 0.5 across all models throughout the paper. To calculate the scaling factors, we gathered the statistics of activations using 128 random sentences from the *WikiText-2-train* dataset. Once we calculated the scaling factors, we used the same values to evaluate the models with different quantization formats.

Targeted layers. Similar to the previous work (Xiao et al., 2023), we apply smoothing on the input activation of the self-attention and the feed-forward layers of LLMs. Unless stated otherwise, we transform all *Linear* layers to the specified quantization format while keeping the activation/weight in the original format for other layers including *GELU*, *Softmax*, and *LayerNorm*.