A Leap Forward in LLMs Post-Training W4A8 Quantization Using Floating-Point Formats

Xiaoxia Wu*, **Zhewei Yao***, **Yuxiong He**
Microsoft
{zheweiyao, xiaoxiawu, yuxhe}@microsoft.com

Abstract

In the complex domain of large language models (LLMs), striking a balance between computational efficiency and maintaining model quality is a formidable challenge. Navigating the inherent limitations of uniform quantization, particularly when dealing with outliers, and motivated by the launch of NVIDIA's H100 hardware, this study delves into the viability of floating-point (FP) quantization, particularly focusing on FP8 and FP4, as a potential solution. Our comprehensive investigation reveals that for LLMs, FP8 activation consistently outshines its integer (INT8) equivalent, with the performance edge becoming more noticeable in models possessing parameters beyond one billion. For weight quantization, our findings indicate that FP4 exhibits comparable, if not superior, performance to INT4, simplifying deployment on FP-supported hardware like H100. To mitigate the overhead from precision alignment caused by the disparity between weights and activations, we propose two scaling constraints for weight quantization that negligibly impact the performance compared to the standard W4A8 model. We additionally enhance our quantization methods by integrating the Low Rank Compensation (LoRC) strategy, yielding improvements especially in smaller models.

1 Introduction

As Natural Language Processing (NLP) evolves, Large Language Models (LLMs) like Codex [6] and ChatGPT [14] have become essential, transforming our interaction with technology and daily communication. However, their complexity and computational intensity present deployment challenges [15, 5, 18], particularly in resource-limited settings. One solution is quantization, which represents data in lower-precision formats such as 8-bit integers or floating-point numbers, reducing memory needs and potentially enhancing inference latency through better GEMM computation throughput on compatible GPUs. Post-Training Quantization (PTQ), which directly reduces the precision of a fully trained model's parameters, is often preferred for LLMs due to its simplicity and lower computational overhead. Recent studies indicate that PTQ on 8-bit integer (INT8) weight-only quantization does not compromise the quality of LLMs [25, 2, 24, 21], and only a minor accuracy drop is observed with INT4 weight quantization when advanced algorithm such as GPTQ applied [4, 26, 7, 9]. The exploration of activation quantization, in addition to weight-only quantization, has also gained interest. This approach expedites inference times by taking advantage of unified precision leading to more efficient execution on hardware. The primary challenge in implementing activation quantization lies in the trade-off between efficiency and performance. As evidenced in studies such as ZeroQuants [25, 26], SmoothQuant [24] and others, reducing the precision of activation from FP16 to INT8 inevitably results in a decrease in model quality. This degradation is partially due to the presence of extreme values or outliers in the activation of LLMs [3, 24, 9, 7], which is partly attributed to the pretraining effect [23]. In the presence of outliers, uniform quantization like INT8 or INT4, fail to accurately

*Equal Contribution. Code will be released as a part of https://github.com/microsoft/DeepSpeed

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

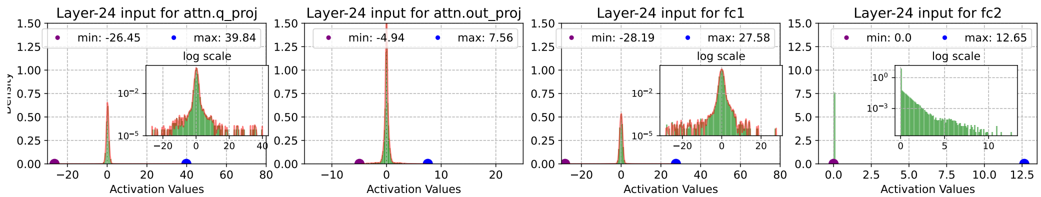

Figure 1: Distribution of Activation values of the final layer of the pretrained OPT-1.3b model. From the left to right columns, they are respectively for the linear modules attn.q_proj (same as attn.k_proj and attn.v_proj), attn.out_proj, fc1, and fc2. The histogram's x-axis ranges from the smallest to largest activation values, while the y-axis denotes their frequency in the dataset. See legend for their minimum and maximum values. Density functions illustrate the probability of different activation values. For more details, please see Section 2.

represent the main body of the data as they become skewed towards the outlier. This issue stems from the inherent assumption in these techniques of a uniform data distribution [22], an assumption that might not correspond to the actual data points distribution.

Considering the drawbacks of integer quantization delineated previously, floating-point (FP) methods like FP8 or FP4, employing ExMy notation, arise as more potent alternatives [12, 1, 8, 20, 28]. Unlike the fixed range of integer types, floating-point methods allow for adjusting the decimal point position, enabling dynamic scaling across activation maps and preserving important features. While there is debate about the quality of models between integer and floating-point quantization [20], recent research on PTQ LLMs using FP8/FP4 in [28] reveals FP8 to be substantially better than INT8 activation quantization. In terms of hardware support and performance, while INT8 computations are broadly supported by most modern CPUs and GPUs [13, 23], lower-bit floating-point operations are also increasingly recognized in the industry. An example of this is the newly release of NVIDIA's H100 GPU, specifically engineered for FP8 computations [12]. Hence, despite the potentially higher computation cost of FP8 compared to INT8 and in light of hardware support, the improved model quality could make this trade-off worthwhile and merits further exploration.

While a few studies such as the one by [28] have ventured into the realm of post-training FP quantization in LLMs, they have unveiled considerable drawbacks in terms of model quality. Specifically, when implementing GPTQ [4] for FP8 quantization on both weights and activation for models such as LLaMA-7B or LLaMA-30b [19], there is an observed perplexity degradation surpassing 1.0 on Wiki-text2 dataset [11]. This level of model degradation presents significant practicality issues, hindering the optimal utilization of these models. In response to these findings, our paper undertakes an in-depth exploration into FP quantization. We primarily focus on activation — an integral element that could potentially enhance the system performance of these quantization techniques.

Contribution. Our main contributions include:

(1) Demonstrating minimal degradation with FP8 activation and weight quantization: Particularly in larger models, FP8 activation and weight quantization result in negligible degradation, performing comparably to the original FP16 models.

(2) Identifying potential in FP8 activation and FP4 weights, and the impact of Low Rank Compensation (LoRC): We highlight the potential in FP8 activation and FP4 weights. The LoRC method, proposed in [26], significantly reduces quantization errors in the W4A8 scheme for FP quantization, especially in smaller models, thereby enhancing performance.

(3) Illustrating the maintenance of quality in the W4A8 floating-point model even when constraints are imposed on the scaling factors: For true efficiency in the W4A8 model, a conversion from FP4 to FP8 for weight is crucial. To alleviate this converting overhead, we here suggest two possible scaling constraints for weight quantization: (a) restricting all scaling factors to be a power of 2 and (b) requiring the scaling factors in one compute group (e.g., several rows of the weight matrix [25] to be transferable by simple bit-shifting). Our analysis indicates that these two restrictions negligibly affect the model's performance in comparison to the conventional W4A8 configuration.

2 Motivation and Methodology

The impact of 8-bit activation quantization, especially potential accuracy loss, is comprehensively outlined in ZeroQuant-V2 [26]. They present a direct comparison between the W16A16 and W16A8 (INT8) quantization schemes across a variety of models. To provide an easier understanding, we quoted their results (Table 1 in [26]) for both OPT[27] and BLOOM [17] models, which indicates that the quality of models, especially the OPT family, is significantly influenced by the activation quantization.

Distribution of Activations. We sought to understand the cause of the aforementioned degradation from FP16 activation and INT8, prompting us to scrutinize the distribution of activation values illustrated in Figure 1. We selected a random sentence from the C4 dataset and processed it through a pre-trained OPT-1.3B model. The statistical activation inputs for the 2nd, middle, and final layer were subsequently chosen for a detailed examination. The four histograms correspondingly represent the activations for the Multi-head Attention (MHA) and Multi-Layer Perceptron (MLP):[2] (1) attn.q_proj, the input for the query, key, or value in the MHA mechanism; (2) attn.out_proj, the input for the MHA's projection matrices; (3)fc1, the initial input for the fully-connected (fc1) projection in MLP; (4)fc2, the subsequent input for the fully-connected (fc2) projection in the MLP.

The activation distribution outlined in Figure 1 reveals the skewness for all modules. In this particular module, a large portion of the values cluster around zero, with only a handful surpassing this range. This phenomenon is due to the inputs being processed by the "ReLU" (Rectified Linear Unit) operator. This operator, purposefully, voids any negative input values, resulting in a skewed distribution focused around zero. Only positive activation values persist unmodified, giving rise to the outliers observed. This extreme skewness is most noticeable at the final layer. These observations offer a deeper understanding of how activation quantization impacts various modules, even within the same layer. Consequently, this signifies that we must exercise caution when selecting quantization methods. Quantization techniques that employ integers, such as INT8 or INT4, and rely on uniform quantization, may not be ideally suited to manage distributions that are skewed. This is due to the inherent assumption of uniform distribution within these methods, which may not align with the actual distribution of data points.

Motivation of 8-bits and 4-bits floating-points. The integer quantization such as in INT8 or INT4 states $Q(x) = \text{INT}\big((x - Z)/S\big) - Z$, where Q is the quantization function, x is a floating point input vector/tensor, S is a real valued scaling factor, and Z is an integer zero point. Based on different settings, the quantization method can be viewed as symmetric ($Z = 0$) or asymmetric ($Z \neq 0$) quantization. In scenarios where outliers exist, uniform quantization techniques like INT8 and INT4, regardless of their symmetric or asymmetric variants, frequently fail to accurately approximate the values of clustered data. Consequently, this makes the quantization error larger for those clustered values, as these methods attempt to adjust their fit to accommodate the outlier. Essentially, these techniques become skewed towards the outlier, leading to a reduced accuracy in representing the main body of the data. Given the limitations of integer quantization, floating-point methods such as FP8 or FP4, utilizing ExMy notation, emerge as superior alternatives. In these methods, the 'x' and 'y' values represent the bits allocated for the exponent and mantissa, respectively, totaling to 7 in FP8 or 3 in FP4. The flexibility of FP8 lies in its ability to adjust the decimal point position, unlike integer types with a fixed range. We present in Section B on how to efficiently

Considering the advantages of ExMy, which allows for dynamic scaling across activation maps, quantization error is reduced and essential features are preserved. In this paper, we investigate the performance of FP8 or FP4 techniques for handling the variability in activation or weight values. This could potentially lead to an enhancement in the model's performance on post-training quantization.

In light of ZeroQuant-V2 [26], we applied fine-grained quantization (FGQ) for weight and token-wise quantization for activation. In addition, we will also investigate the add-on feature LoRC (Low Rank Compensation) proposed in [26], which aims to reduce quantization errors in weights by employing low-rank matrix factorization. LoRC involves two main steps: first, it performs Singular Value Decomposition (SVD) on the error matrix, which is the difference between the original weight and the quantized weight. The error matrix is thus decomposed into two unitary matrices and a diagonal matrix. Second, the method formulates a new error approximation using two low-rank matrices that are derived from the matrices in the first step. This approximation is then added to the quantized weight to yield a more accurate estimate of the original weight, thereby reducing quantization errors.

Based on GPTQ (without or with LoRC), we perform comprehensive comparisons between the use of FP8 or INT8 activation quantization, coupled with adjusting the weight quantization to FP8 and FP4. Particularly we explore the potential of FP4 weight and FP8 activation quantization.

[2]The hidden dimension is 2048 for 'attn.q_proj', 'attn.out_proj' and 'fc1', and 8196 for 'fc2'. We pick 20 tokens (position 8 to 28) and vectorize this 20×2048 or 20×8196 matrices to plot their distributions.

Table 1: The evaluation outcomes for LLaMA (top) and OPT (bottom) using different Integer (INT) and Floating-point (FP) quantization methods applied to weight and activation. The performance is measured in terms of perplexity (lower scores are better) and spans across three datasets: WikiText-2 (WIKI), PTB, and C4.

Q-type	Weight- -Activation	LLaMA-3b Mean	LLaMA-3b WIKI/PTB/C4	LLaMA-7b Mean	LLaMA-7b WIKI/PTB/C4	LLaMA-13b Mean	LLaMA-13b WIKI/PTB/C4	LLaMA-30b Mean	LLaMA-30b WIKI/PTB/C4
W16A16	N/A	11.93	7.35/19.1/9.34	13.37	5.68/27.35/7.78	10.31	5.09/19.22/6.61	5.79	4.10/7.30/5.98
W8A8	INT – INT	12.00	7.41/19.16/9.41	13.58	5.72/27.89/7.13	10.63	5.16/20.07/6.67	5.90	4.21/7.42/6.06
	INT – FP	11.96	7.37/19.16/9.35	13.45	5.69/27.57/7.09	10.38	5.11/19.42/6.62	5.80	4.11/7.31/5.99
	FP – FP	11.99	7.37/19.23/9.37	13.46	5.70/27.58/7.10	10.38	5.11/19.41/6.62	5.81	4.12/7.31/5.99
W4A8	INT – INT	12.55	7.67/20.23/9.74	16.23	6.44/34.45/7.79	11.48	5.32/22.35/6.78	6.02	4.36/7.54/6.16
	INT – FP	12.39	7.62/19.87/9.68	16.09	6.75/33.80/7.72	11.31	5.28/21.91/6.73	5.94	4.27/7.45/6.11
	FP – FP	12.45	7.62/20.05/9.67	15.14	6.32/31.61/7.51	11.08	5.26/21.27/6.73	5.92	4.26/7.42/6.09
W4A8 +LoRC	INT – INT	12.52	7.65/20.18/9.72	14.14	5.88/29.26/7.27	10.81	5.28/20.38/6.76	6.00	4.34/7.51/6.14
	INT – FP	12.38	7.58/19.89/9.65	14.01	5.84/28.95/7.24	10.56	5.22/19.75/6.71	5.90	4.24/7.39/6.07
	FP – FP	12.42	7.61/19.98/9.66	13.95	5.87/28.75/7.24	10.80	5.24/20.46/6.72	5.91	4.26/7.40/6.07

Q-type	Weight – – Activation	OPT-3b Mean	OPT-3b WIKI/PTB/C4	OPT-7b Mean	OPT-7b WIKI/PTB/C4	OPT-13b Mean	OPT-13b WIKI/PTB/C4	OPT-30b Mean	OPT-30b WIKI/PTB/C4
W16A16	N/A	15.44	14.62/16.97/14.72	11.90	10.86/13.09/11.74	11.22	10.13/12.34/11.20	10.70	9.56/11.84/10.69
W8A8	INT – INT	15.94	14.98/17.49/15.36	12.66	11.20/14.29/12.48	15.94	12.13/19.82/15.86	25.76	14.63/32.90/29.74
	INT – FP	15.85	14.93/17.56/15.05	11.99	10.92/13.24/11.80	11.27	10.16/12.42/11.23	10.69	9.51/11.87/10.71
	FP – FP	15.86	14.97/17.55/15.05	11.99	10.91/13.24/11.81	11.27	10.16/12.42/11.23	10.69	9.51/11.87/10.71
W4A8	INT – INT	16.41	15.39/18.22/15.62	13.18	11.61/15.00/12.92	16.70	12.32/21.21/16.56	24.42	14.80/30.38/28.09
	INT – FP	16.40	15.46/18.23/15.51	12.20	11.13/13.49/11.99	11.34	10.20/12.53/11.30	10.73	9.54/11.91/10.75
	FP – FP	16.29	15.32/18.19/15.35	12.09	10.89/13.44/11.95	11.34	10.16/12.55/11.30	10.72	9.52/11.90/10.75
W4A8 +LoRC	INT – INT	16.38	15.50/18.05/15.59	12.75	11.37/14.33/12.53	15.89	12.06/19.76/15.85	27.20	15.94/34.50/31.16
	INT – FP	16.23	15.40/17.97/15.32	12.13	11.07/13.43/11.90	11.34	10.23/12.49/11.29	10.71	9.48/11.91/10.74
	FP – FP	16.23	15.50/17.92/15.28	12.09	10.96/13.40/11.90	11.33	10.15/12.55/11.29	10.71	9.48/11.90/10.75

3 Main Results and Conclusion

In the experiments, we include two model-type families: LLaMA [19] and OPT [27], with sizes ranging from 1 billion to 30 billion parameters. The evaluation spans across three datasets: Wikitext-2 (WIKI) [11], PTB [10], and C4 [16]. For more experiment details, please see Appendix A. The primary results in Table 1 reveal the impact of various quantization types which are applied to weight and activation specified in the 2nd column; for instance, W4A8 precision, INT – FP means INT4 is used for weight and FP8 for activation.

FP8 Activation is much better than INT8. The high-level summary of the results in Table 1 indicates that for both LLaMA and OPT model families, FP8 activation generally outperforms INT8 activation. This observation corroborates the motivation discussed in Section 2, emphasizing FP8's superior capacity to capture more nuanced information.

FP8 weights rival INT8, while FP4 weights potentially outperform INT4. From Table 1, we observe comparable performances between INT8 and FP8 weight quantization across various models and datasets, when keeping activation at FP8. This probably due to we used FGQ on weight quantization. Interestingly, when weight quantization is lowered, FP4 exhibits certain advantages over INT4, particularly evident in LLaMA-7b (15.14 to 16.09) and LLaMA-13b models (11.08 to 11.31). Specifically, under the W4A8 configuration for LLaMA-7b, we see 0.95 improvement of FP4 over INT4, a significant gain. The preferable performance of FP4 over INT4 is particularly advantageous for hardware designs like H100, where FP8 is already supported. Thus, a simple modification to accommodate FP4 would be easier than implementing a system supporting INT4 weight and FP8 activation.

LoRC improves W4A8. Table 1 shows that the Low Rank Compensation (LoRC) method enhanced the W4A8 quantization scheme, reducing quantization errors. This improvement is particularly pronounced in smaller models, underlining the effectiveness of LoRC in optimizing the performance of these computing processes while impacting little on the model-size.

Conclusion. In this study, we demonstrate that floating-point (FP) quantization significantly surpasses integer (INT) quantization in the context of large language models (LLMs) during post-training quantization. Notably, FP8 activation exceeds INT8, especially in larger models. Moreover, FP8 and FP4 weight quantization are either competitive with or surpass their INT equivalents. The Low Rank Compensation (LoRC) approach greatly enhances the W4A8 quantization scheme, particularly in smaller models.

References

[1] Léopold Cambier, Anahita Bhiwandiwalla, Ting Gong, Mehran Nekuii, Oguz H Elibol, and Hanlin Tang. Shifted and squeezed 8-bit floating point format for low-precision training of deep neural networks. *arXiv preprint arXiv:2001.05674*, 2020.

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

[3] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. *arXiv preprint arXiv:2212.09720*, 2022.

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

[5] Amir Gholami, Zhewei Yao, Sehoon Kim, Michael W Mahoney, and Kurt Keutzer. Ai and memory wall. *RiseLab Medium Post*, 2021.

[6] GitHub. Github copilot. https://github.com/features/copilot/, 2021.

[7] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

[8] Andrey Kuzmin, Mart Van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, and Tijmen Blankevoort. Fp8 quantization: The power of the exponent. *arXiv preprint arXiv:2208.09225*, 2022.

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

[10] Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, page 273, 1994.

[11] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[12] Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

[13] NVIDIA. FasterTransformer. https://github.com/NVIDIA/FasterTransformer, January 2023.

[14] OpenAI. Openai chatgpt. https://openai.com/blog/chatgpt/, 2022.

[15] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *arXiv preprint arXiv:2211.05102*, 2022.

[16] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019.

[17] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[18] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

[19] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[20] Mart van Baalen, Andrey Kuzmin, Suparna S Nair, Yuwei Ren, Eric Mahurin, Chirag Patel, Sundar Subramanian, Sanghyuk Lee, Markus Nagel, Joseph Soriaga, et al. Fp8 versus int8 for efficient deep learning inference. *arXiv preprint arXiv:2303.17951*, 2023.

[21] Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*, 2023.

[22] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

[23] Xiaoxia Wu, Cheng Li, Reza Yazdani Aminabadi, Zhewei Yao, and Yuxiong He. Understanding int4 quantization for transformer models: Latency speedup, composability, and failure cases. *arXiv preprint arXiv:2301.12017*, 2023.

[24] Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*, 2022.

[25] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *arXiv preprint arXiv:2206.01861*, 2022.

[26] Zhewei Yao, Cheng Li, Xiaoxia Wu, Stephen Youn, and Yuxiong He. A comprehensive study on post-training quantization for large language models. *arXiv preprint arXiv:2303.08302*, 2023.

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

[28] Yijia Zhang, Lingran Zhao, Shijie Cao, Wenqiang Wang, Ting Cao, Fan Yang, Mao Yang, Shanghang Zhang, and Ningyi Xu. Integer or floating point? new outlooks for low-bit quantization on large language models. *arXiv preprint arXiv:2305.12356*, 2023.

A Experiment Details

As we used GPTQ method [4], we use C4 dataset to randomly select 128 sentences for the light-weight PTQ and each of them has 2048 tokens. We run them on a single GPU (i.e, V100-32GB) thanks for the two open-source github repositories.[3] To accommodate the real computation efficiency, the group-size for weight quantization is 256 for both model family (OPT and LLaMA) except the LLaMA-3b with 320 as its hidden-dimension is 3200. All the checkpoints we used are from huggingface.[4] As for activation, we perform token-wise quantization in order to accommodate the latency requirements. For LoRC method, the dimension for the two low-rank matrix we used for LLaMA is 8. While for OPT, the dimension is 16, 32, 40 and 56 respectively for 1.3b, 6.7b, 13b and 30b. We did not try others dimension as indicated by [26] that dimension of the low-rank matrix does not play too much impact on the quantization error as long as it larger than 8.

Table A.1: Comparisons between E2M1 and E3M0. The quantization is FP4 for weight and FP8 for activation.

Activation (FP8)	OPT-1.3b	OPT-6.7b	OPT-13b	OPT-30b
Weight-FP4 (E3M0)	16.96	12.41	11.53	10.86
Weight-FP4 (E2M1)	16.23	12.09	11.33	10.71

B Casting FP4 to FP8

A unique challenge arises due to the use of different precision levels for weights (W) and activations (A). The actual software implementation of W4A8 in H100 NVIDIA hardware is that one needs to cast W's FP4 to match the FP8 precision used in A. The direct method of dequantization followed by quantization again could potentially have a detrimental effect on inference efficiency, hence it is not a viable solution. To address this, we propose the bit-shifting method.

As described above that in order to achieve better real latency speedup in the NVIDIA H100 hardware, it is suggested that the scale S for the weight quantization needs to be represented as a power of 2. We thus repeat the set of experiments for FP4 weight and FP8 activation quantization, with and without LoRC, shown in Table B.1. We see that without LoRC, the degradation of restricting scales $S = 2^n$ could make the models degrades considerable large (> 1.5PPL), except for the lager models LLaMA-30b and OPT-30b. However, with LoRC, such degradation has been relatively reduced including for the mode OPT-3b, OPT-7b,LLaMA-7b and LLaMA-13b.

Results. As detailed in Section B, to maximize real latency speedup on NVIDIA H100 hardware, we suggest the scale factor S for weight quantization to be represented as a power of 2. In pursuit of this, we executed a series of experiments using FP4 for weight and FP8 for activation quantization. The results of these experiments, conducted both with and without LoRC, are presented in Table B.1. Our data shows that while constraining the scaling factors occasionally results in unexpected improvements in models like LLaMA-7b and LLaMA-13b, we generally observe a minor degradation of quality in the W4A8 floating-point model, regardless of whether we used method M1 or M2. M2 generally outperforms M1. When we implement LoRC, this decline in quality can be mitigated, particularly in the OPT-1.3b, LLaMA-7b, and LLaMA-13b models. Hence, our results advocate for the use of LoRC, especially when considering scale restrictions for weight quantization in deep learning models.

[3]`https://github.com/IST-DASLab/gptq` and `https://github.com/qwopqwop200/GPTQ-for-LLaMa.git`

[4]LLaMA-3b is `openlm-research/open_llama_3b` and all other LLaMA are from `decapoda-research/llama-#b-hf` where `#` can be 7b, 13b and 30b. As for OPT, they are from `facebook/opt-#b` where `#` can be 1.3b, 7b, 13b and 30b.

Table B.1: Scale values (S) are evaluated both without and with restrictions of being a power of 2, as shown in the second column. The quantization type employed is FP4 for weight and FP8 for activation.

Q-type	Scale $S = 2^n$	LLaMA-3b Mean	LLaMA-3b WIKI/PTB/C4	LLaMA-7b Mean	LLaMA-7b WIKI/PTB/C4	LLaMA-13b Mean	LLaMA-13b WIKI/PTB/C4	LLaMA-30b Mean	LLaMA-30b WIKI/PTB/C4
W4A8	✗	12.45	7.62/20.05/9.67	15.14	6.32/31.61/7.51	11.08	5.26/21.27/6.73	5.92	4.26/7.42/6.09
	✓(M1)	12.66	7.76/20.41/9.81	16.33	6.34/34.82/7.82	10.90	5.31/20.63/6.78	6.00	4.38/7.48/6.15
	✓(M2)	12.55	7.68/20.21/9.77	14.49	6.37/29.32/7.79	10.95	5.26/20.81/6.77	5.97	4.31/7.48/6.13
W4A8 LoRC	✗	12.42	7.61/19.98/9.66	13.95	5.87/28.75/7.24	10.80	5.24/20.46/6.72	5.91	4.26/7.40/6.07
	✓(M1)	12.61	7.69/20.37/9.76	14.23	5.94/29.47/7.30	10.74	5.28/20.19/6.75	5.98	4.33/7.48/6.13
	✓(M2)	12.42	7.63/19.89/9.74	13.68	5.90/27.83/7.32	10.40	5.23/19.22/6.76	5.94	4.28/7.44/6.11

Q-type	Scale $S = 2^n$	OPT-1.3b Mean	OPT-1.3b WIKI/PTB/C4	OPT-6.7b Mean	OPT-6.7b WIKI/PTB/C4	OPT-13b Mean	OPT-13b WIKI/PTB/C4	OPT-30b Mean	OPT-30b WIKI/PTB/C4
W4A8	✗	16.29	15.32/18.19/15.35	12.09	10.89/13.44/11.95	11.34	10.16/12.55/11.30	10.72	9.52/11.90/10.75
	✓(M1)	16.66	15.65/18.66/15.65	12.29	11.12/13.69/12.05	11.36	10.22/12.54/11.32	10.77	9.58/11.96/10.76
	✓(M2)	16.47	15.23/18.55/15.62	12.25	11.11/13.61/12.03	11.40	10.22/12.61/11.36	10.74	9.47/11.96/10.78
W4A8 LoRC	✗	16.23	15.50/17.92/15.28	12.09	10.96/13.40/11.90	11.33	10.15/12.55/11.29	10.71	9.48/11.90/10.75
	✓(M1)	16.47	15.59/18.37/15.45	12.17	11.10/13.47/11.95	11.36	10.21/12.54/11.32	10.74	9.49/11.96/10.76
	✓(M2)	16.30	15.39/18.10/15.42	12.19	11.11/13.49/11.97	11.41	10.34/12.54/11.34	10.75	9.49/11.96/10.78