TAGQuant: Token-Aware Clustering for Group-Wise Quantization

Jaeseong Lee[*], **Seung-won Hwang**[*],
Aurick Qiao, Zhewei Yao, Yuxiong He
Snowflake AI Research, Seoul National University[*]

Abstract

Grouping, e.g., grouping channels, which is widely used in current integer-based quantization, has become essential for the emerging MXFP4 format. Ideally, each group should contain channels with similar quantization scales. To guide such groups, existing work clusters the channels using scalar proxy, ignoring the token dimension, which we find suboptimal. In this paper, we propose TAGQuant, a simple yet powerful enhancement for such "group-wise" quantization. By strategically shuffling channels to group those with similar token-wise activation distributions, TAGQuant ensures better clustering of large- and small-range values. This shuffle operation is hardware-efficient, and seamlessly integrated into the quantization process with only 0.01× latency overhead. TAGQuant reduces relative GSM8K error in both INT4 and MXFP4 formats, by up to 86% in Llama-3.1-8B-Instruct compared to baselines, validating the effectiveness of our channel shuffling approach for group-wise quantization. Code is publicly available.

1 Introduction

A common challenge across both algorithmic and hardware perspectives in large language model (LLM) quantization is supporting "group-wise" quantization– quantizing consecutive channels. After its introduction (Shen et al., 2020; Yao et al., 2022), it has been widely used in integer-based quantization (Frantar et al., 2023; Ashkboos et al., 2024b), though optional. Moreover, group-wise quantization with a small group size of 32 is officially adopted in the recently proposed MXFP4 format (Rouhani et al., 2023). Therefore optimizing group-wise quantization for small group sizes has become essential for quantization efficiency.

In this paper, we investigate how to optimize group-wise quantization. The key problem is outliers, which significantly expand the quantization scale, which controls the range, or granularity, of the quantization. These outliers increase granularity, making the other values in the same group to be undistinguishable after quantization.

Ideally, each group should contain channels with similar quantization scales. To guide such groups, a straightforward approach would be clustering channels with similar quantization scales, and use those clustered channels in the same group.

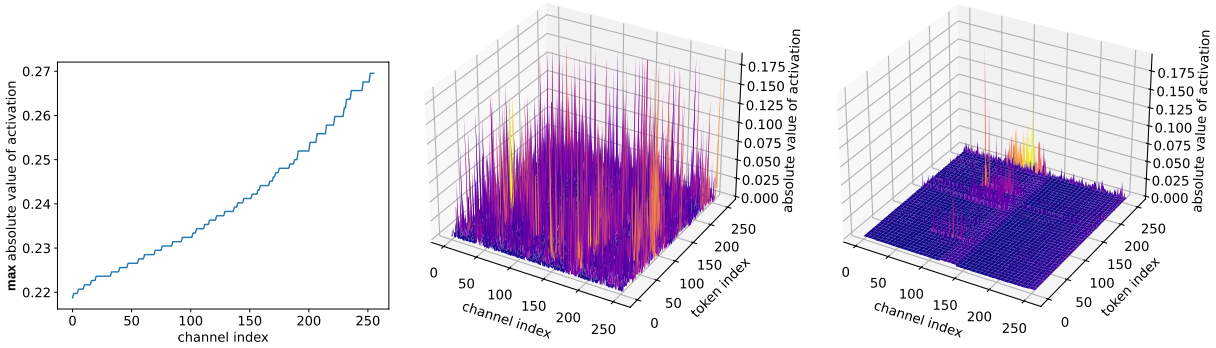
Existing work to cluster the channels, RPTQ (Yuan et al., 2023), identifies channels with similar quantization scales, with a scalar proxy $\max_j |a_{j,c}|$ per channel c, where $a_{j,c}$ is the activation of channel c for jth token. We argue it simplifies the activation values of each channel too much, ignoring the **token** dimension. Figure 1b with token dimension describes this challenge– channel index reordered by a scalar proxy alone (y-score in Figure 1a) does not guarantee consecutive channels have similar quantization scales, leading to high quantization error within each quantization group.

To address this limitation, we introduce TAGQuant, which captures finer-grained activation dynamics, considering the token dimension. Instead of relying on the scalar values, we propose to cluster vectors of token-wise activation distributions, and then reorder them. We then apply a dendrogram-based optimal leaf ordering algorithm to reorganize channels, ensuring adjacent channels exhibit similar token-wise distribution patterns. After obtaining an ordering based on calibration data, the ordering is then fixed for hardware efficiency. In this new channel index, the consecutive channels exhibits similar token-wise distribution (Figure 1c), making it easier to quantize in groups.

- We propose TAGQuant, a method that substantially enhances group-wise quantization.
- Unlike the existing grouping optimization

[*]Work done while visiting Snowflake. Correspond to seungwonh@snu.ac.kr

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track), pages 253–262
March 25-27, 2026



(a) 2D plot after RPTQ (b) 3D plot after RPTQ (c) 3D plot after TAGQuant

Figure 1: (a) RPTQ (Yuan et al., 2023) reorders by maximum value across token indices. However, when the token index (y-axis) is also plotted as in (b), it is highly irregular– consecutive channels with similar proxy may not have similar token-wise distribution, leading to high group-wise quantization error. (c) In contrast, TAGQuant clusters channels well so that consecutive channels have similar token-wise distribution.

technique, we introduce a finer-grained strategy using token-wise activation distributions to optimize the grouping.

- Our newly introduced operation to materialize TAGQuant incurs minimal overhead, adding only 0.01× latency on A10 GPUs.
- Our approach significantly improves W4A4 quantization efficiency in both INT4 and recently proposed MXFP4, demonstrating superior accuracy retention on LLaMA-3 and Phi-4.
- Code is publicly available.[1]

2 Related Works

2.1 Post-Training Quantization (PTQ)

Quantization techniques for neural networks generally fall into two broad categories: Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ). QAT fine-tunes the model while emulating low-precision arithmetic for weights and activations. However, QAT requires additional computational resources and data, which can be prohibitively expensive for very large models. In contrast, we focus on PTQ, with the benefit of pre-trained models to lower precision after full training, without updating its weights. PTQ is appealing because it requires no additional gradient-based training– becoming more practical in the LLM era.

Round-to-Nearest (RTN) RTN is the most naïve PTQ technique. After scaling numbers into the allowed range of k-bit format, it simply rounds the given scaled number to the nearest k-bit number.

[1] https://github.com/thnkinbtfly/TAGQuant

2.2 Activation Outliers in PTQ

Efficient low-bit quantization of large models has prompted many techniques to deal with the effect of *activation outliers*– the large-magnitude values that distort quantization. The following solutions have been proposed:

Mixed-Precision LLM.int8 (Dettmers et al., 2022) splits out the most extreme activation channels to 16-bit while quantizing the remaining values to 8-bit. QUIK (Ashkboos et al., 2024a) similarly extracts the outlier channels to use 16-bit while quantizing the remaining channels to 4-bit. Unlike LLM.int8, they fix channel indices for hardware acceleration.

Channel Scaling SmoothQuant (Xiao et al., 2023) migrates the quantization difficulty from activations to weights via a rescaling transformation. They smooth out activation outliers, making their distributions more uniform, by offloading each activation's scale into its corresponding weight. Similarly, AWQ (Lin et al., 2024) identifies a small subset of particularly sensitive weight channels by analyzing activation statistics. They amplify those weight values before the weight-quantization.

Rotation-Based QuaRot (Ashkboos et al., 2024b) applies the Hadamard transformation, whose matrix is a rotation matrix, to remove outliers from hidden representations. This redistributes the variance of extremely large activation components across many dimensions. This rotation is computationally invariant.

Distinction We focus on optimizing the grouping, which can be orthogonally applied with each

of these techniques. Moreover, while these algorithms were mainly evaluated on integer-based format only, we evaluate the algorithms on MXFP4 with weight 4bits and activation 4bits (W4A4) format as well.

2.3 Optimizing the Grouping for Outliers

RPTQ (Yuan et al., 2023) identifies outlier channels based on scalar proxies. Based on this information, they reorder the channels to cluster channels with similar quantization scales.

However, their simplified optimization ignores the token-wise distribution patterns (Figure 1). In contrast, we leverage token-wise activation distributions to optimize the grouping. This results in a better grouping strategy, boosting the benchmark performances (Table 3).

3 Proposed Method

3.1 Preliminaries

3.1.1 Integer-based Group-wise Quantization

Integer-based quantization has been popular with various algorithmic support (Shen et al., 2020; Yao et al., 2022; Frantar et al., 2023; Xiao et al., 2023; Lin et al., 2024; Ashkboos et al., 2024a) and hardware support such as Ampere GPUs (NVIDIA, 2020). Among symmetric quantization and asymmetric quantization, we use asymmetric quantization as default, following Gong et al. (2024).

Integer-based group-wise quantization groups every consecutive k elements, and each group shares a scale, and zero-point value. Formally, let $\{x_1, \ldots, x_k\}$ be the set of real numbers in one group. INT4, for example, encodes this group as $(S, z, \{q_i\})$, where S is the scale, z is the zero-point value, and each q_i is the 4-bit value for x_i. The real value is reconstructed as $x_i \approx S \times (q_i + z)$. The scale S and zero-point z can be typically obtained as:

$$z = \min x_i \tag{1}$$

$$S = \frac{\max x_i - \min x_i}{2^4 - 1} \tag{2}$$

3.1.2 MXFP4 Quantization

MXFP4 is a 4-bit format defined by the recent OCP standard for low-precision deep learning (Rouhani et al., 2023). In an MXFP4 representation, a tensor is divided into groups of $k = 32$ elements each, and each group shares a single 8-bit scale of E8M0 format while storing individual values in a 4-bit of E2M1 format (Rouhani et al., 2023). Formally, let $\{x_1, \ldots, x_{32}\}$ be the set of real numbers in one group. MXFP4 encodes this group as $(S, \{q_i\})$, where S is the 'shared scale' and each q_i is the 4-bit value for x_i. The real value is reconstructed as $x_i \approx S \times q_i$. The shared scale S can be typically obtained as:

$$S = \lfloor \log_2(\max |x_i|) \rfloor \tag{3}$$

3.1.3 QUIK

QUIK (Ashkboos et al., 2024a) compresses the majority of weight parameters and activation values to 4-bit, but keeps a small subset of outlier elements in higher precision (e.g. 16-bit) for accuracy. With calibration data, they first identify which channels tend to produce extreme values, and use higher precision for those channels afterwards, which is orthogonal with group optimization such as RPTQ or our work. They also implement GPU kernels to efficiently support this mixed-precision format, getting up to 3.4x throughput improvement over FP16.

3.2 TAGQuant

We highlight three key contributions of TAGQuant in each subsections.

1. Capturing finer-grained activation dynamics of token-wise activation distributions,
2. Tailoring the algorithm for INT4 or MXFP4 format.
3. Proposing channel shuffling to mitigate the discrepancy in a hardware-efficient manner,

3.2.1 Capturing Finer-Grained Activation Dynamics: Token-Wise Distribution

The activation outliers of each channel are typically estimated by the maximum absolute values of the activations (Yuan et al., 2023; Xiao et al., 2023). We observe that this coarse-grained analysis leads to suboptimal grouping for group-wise quantization (Table 3) To find out why, we investigate the maximum absolute value of the activations (Figure 1a), and depict the absolute values by preserving the token index (y-axis in Figure 1b), on RPTQ-sorted Llama-3.1-8B-Instruct. As already discussed, the existing group optimization technique, RPTQ (Yuan et al., 2023), ordering by the maximum absolute values (Figure 1a) fails to capture token-wise distribution (Figure 1b).

Llama-3.1-8B-Instruct	bits	GSM8K
Original	16	81.7
RTN	4	60.5
QuaRot (Ashkboos et al., 2024b)	4	57.6
SmoothQuant (Xiao et al., 2023)	4	60.7
QUIK (Ashkboos et al., 2024a)	4	**77.6**

Table 1: Accuracy(%) of downstream tasks of various quantization algorithm on W4A4 MXFP4 with Llama-3.1-8B-Instruct.

		GSM8K	GPQA	DROP	MGSM
$k = 128$	RTN	70.2	22.5	**47.7**	**47.6**
	QuaRot	**72.2**	**28.6**	43.0	42.7
$k = 32$	RTN	**79.1**	**27.9**	**54.6**	**59.0**
	QuaRot	71.6	22.5	53.6	49.4

Table 2: Accuracy(%) of downstream tasks of quantization algorithms on W4A4 INT4 format with Llama-3.1-8B-Instruct, varying the group size (k). QuaRot suffers when group size becomes smaller.

Therefore, we propose to compare the distributions in a finer-grained manner– using a calibration dataset, we compare the token-wise distributions d_c per channel c, to determine channels with similar scales per token.

3.2.2 Tailoring Token-Wise Distribution (d_c)

We now materialize the mapping f for channel shuffling. Formally, consider the input activation $a_{i,j,c}$, where i is the batch index, j is the token index, and c is the channel index.

Tailoring d_c for MXFP4 To find channels with similar token-wise distributions, we aim to model d_c as the histogram of the scale values across the batch. We first get the scale value extending Eq. 3:

$$S_{i,j,c} = \lfloor \log_2 |a_{i,j,c}| \rfloor \tag{4}$$

Considering the scale value $S_{i,j,c}$ uses 8-bit in MXFP4, we can cheaply obtain the token-wise scale histogram $d_c = (f_{0,0,c}, \cdots, f_{255,N,c})$ per channel c as follows:

$$f_{m,j,c} = \sum_i \mathbb{1}(S_{i,j,c} = m) \tag{5}$$

where N is the sequence length, and $\mathbb{1}$ is the indicator function.

Tailoring d_c for INT4 Unlike MXFP4, the scale value and the zero-point value are not restricted to 8-bit in INT4. Therefore, we concatenate all the activation values across the batch to model the token-wise distribution per channel. We simply model token-wise distribution d_c per channel c as follows:

$$d_c = (a_{0,0,c}, a_{0,1,c}, \cdots, a_{B,N,c}) \tag{6}$$

where B is the batch size.

Obtaining Shuffle Ordering f From d_c Now we aim to derive the shuffle ordering so that the channels in the same group tend to have similar scales, lowering the quantization error per each group.

First, to cluster channels with similar vectors of d_c, we draw a dendrogram of d_c using the UPGMA algorithm (Sokal and Michener, 1958).

We use the L^2 distance as the distance metric between two vectors of d_c. The distance between two channels c_1 and c_2, or two clusters of channels C_1 and C_2, are defined as:

$$D(c_1, c_2) = \|d_{c_1} - d_{c_2}\|_2 \tag{7}$$

$$D(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{c_1 \in C_1} \sum_{c_2 \in C_2} D(c_1, c_2) \tag{8}$$

At each step, the two closest clusters are merged until all channels are merged into a single cluster.

Appendix A reports our empirical selection of this algorithm and use of L^2 distance.

Now, we want to derive the mapping f from the ordering of the leaf nodes of the obtained dendrogram. Among diverse equivalent dendrograms, we choose one by applying a leaf ordering algorithm optimized for hierarchical clustering (Bar-Joseph et al., 2001) to ensure the adjacent channel indices have similar token-wise scale distribution. Finally, we obtain the index mapping for shuffling $f(c_l) = l$ where c_l is the original channel index of the lth leaf in the dendrogram. Figure 1c shows that TAGQuant clusters channels well to make token-wise distribution of consecutive channels similar.

3.2.3 Hardware-Efficient Channel Shuffling

Shuffling channels may seem to incur additional overhead, but we restrict mapping f to be fixed, then we can fuse the shuffling into the quantization kernel provided in QUIK (Ashkboos et al., 2024a) implementation. Our experiments show that there is no noticeable latency overhead (Table 5).

4 Experiments

In this section, we aim to address the following research questions:

	bits	Llama-3.1-8B-Instruct				Phi-4-14B			
		GSM8K	GPQA	DROP	MGSM	GSM8K	GPQA	DROP	MGSM
Original	16	81.7	32.8	59.7	67.1	93.4	51.6	69.0	82.2
QUIK (Ashkboos et al., 2024a)	4	77.6	25.9	**57.1**	58.0	92.6	44.6	68.0	80.7
QUIK+RPTQ (Gong et al., 2024)	4	77.8	28.1	56.5	58.1	92.9	44.2	68.3	80.8
QUIK+TAGQuant	4	**80.0**	**29.7**	56.8	**58.9**	**93.6**	**46.4**	**69.0**	**81.4**

Table 3: Accuracy(%) of downstream tasks of various quantization algorithm on W4A4 MXFP4 format (group size $k = 32$).

- RQ1: Does TAGQuant improve the performance?
- RQ2: Does TAGQuant overcome the failure cases of existing INT4 or MXFP4 W4A4 quantizaiton?
- RQ3: Is TAGQuant optimized for hardware efficiency?
- RQ4: Does TAGQuant reduce the activation variance within each group?

We employ LLMs over diverse families and scales: Llama-3.1-8B-Instruct (Dubey et al., 2024), and Phi-4-14B (Abdin et al., 2024).

Tasks and Datasets We evaluate with GSM8K 8-shot CoT (Cobbe et al., 2021), a math reasoning dataset; GPQA 0-shot CoT (Rein et al., 2024), a graduate-level QA dataset; DROP 3-shot (Dua et al., 2019), reading comprehension dataset; and Multilingual GSM 0-shot CoT (Shi et al., 2023), which is a multilingual math reasoning dataset. For calibration data, we follow the setting of Gong et al. (2024).

Implementation Details To evaluate, we extend LM-EVALUATION-HARNESS[2] (Gao et al., 2021) to use the prompts used by Llama-3.1 series.[3] To simulate MXFP4 quantization, we extend LLMC (Gong et al., 2024) framework to support MXFP4 format. Implementations of all baselines are adopted from LLMC. Following Ashkboos et al. (2024a), for QUIK, we use higher bits for the last linear layer in each MLP layer (e.g. 16-bit in our implementation), and use 256 channels as 16-bit for each linear layer. For INT4 quantization, we use group size $k = 32$ as default following MXFP4 format, while we will also investigate $k = 128$ for RQ2. All evaluations are done on one H100-80GB, requiring less than 6 hours.

[2] https://github.com/neuralmagic/lm-evaluation-harness/tree/llama_3.1_instruct

[3] https://huggingface.co/datasets/meta-llama/Llama-3.1-8B-Instruct-evals/

	GSM8K	GPQA	DROP	MGSM
Original (16bits)	81.7	32.8	59.7	67.1
QUIK	79.9	28.8	55.1	62.6
QUIK+ShuffleQ	**81.6**	**31.9**	**58.0**	**64.9**

Table 4: Accuracy(%) of downstream tasks of various quantization algorithm on W4A4 INT4 format with Llama-3.1-8B-Instruct (group size $k = 32$).

	$N = 4096$	$N = 8192$
TAGQuant	0.801	2.021
- channel shuffling	0.792	2.009

Table 5: Latency (ms) comparison of $N \times N$ mixed-precision matrix multiplication with and without channel shuffling.

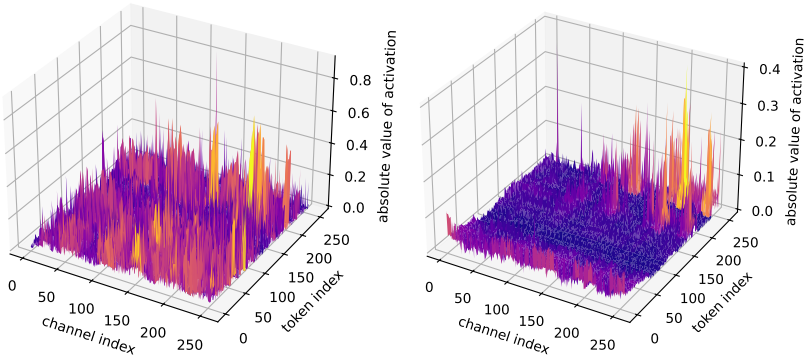
Comparisons We compare the following methods:

- **Round-To-Nearest** (RTN) directly quantize without outlier mitigation.
- **QuaRot** (Ashkboos et al., 2024b) mitigates outliers by rotating with hadamard matrix.
- **SmoothQuant** (Xiao et al., 2023) mitigates outliers by channel scaling.
- **QUIK** (Ashkboos et al., 2024a) extracts outlier channels and use high bits for them.
- **TAGQuant** groups channels with similar activation distributions upon QUIK.

4.1 Experimental Results

RQ1: TAGQuant Improves the Performance Based on Table 1, we mainly compare upon QUIK as the most competitive baseline.

Table 3 shows that TAGQuant outperforms all the baselines in MXFP4 quantization– For example,



(a) 3D plot after RPTQ (b) 3D plot after TAGQuant

Figure 2: Activation distribution after applying RPTQ and TAGQuant on a test dataset, GSM8K.

	Average std
QUIK	0.1605
QUIK+RPTQ	0.1603
QUIK+TAGQuant	**0.1572**

Table 6: Average standard deviation of the errors within quantization groups.

TAGQuant lowers the error in GSM8K by 56% (-3.9%p to -1.7%p) and GPQA by 34% (-4.7%p to -3.1%p) compared with the toughest baseline, QUIK+RPTQ, on Llama-3.1-8B-Instruct.

Table 4 shows that TAGQuant outperforms all the baselines in INT4 quantization as well– For example, TAGQuant lowers the error in GSM8K by 86% (-0.7%p to -0.1%p) and GPQA by 82% (-5.1%p to -0.9%p) compared with the toughest baseline, QUIK, on Llama-3.1-8B-Instruct.

4.1.1 RQ2: Failure Cases of Existing Algorithms on W4A4

Coarse-grained Clustering (RPTQ) The second and third rows of Table 3 show that the clustering of RPTQ provides only marginal improvement. In contrast, TAGQuant provides stark improvements, as described in RQ1.

Small Groups Table 1 shows the weakness of QuaRot (Ashkboos et al., 2024b). To investigate, we compared QuaRot performance varying the group size in Table 2. We find QuaRot suffers when a small group size is used, which is consistent with the recently reported result by Lee et al. (2024). This points out the unique challenge of quantizing LLMs into emerging hardware, such as MXFP4, which constrains group size to be as small as 32. In contrast, TAGQuant works well for the small group size as well (Table 3,4), such as required in MXFP4.

4.1.2 RQ3: Hardware-Efficiency

To compare the latency, we compile the kernels of QUIK and QUIK+TAGQuant on an Ampere GPU (A10), where QUIK kernel is originally designed for. We measure the latency of $N \times N$ mixed-precision matrix multiplication, following the setting of QUIK (Ashkboos et al., 2024a).

Table 5 shows only the negligible overhead of the fused shuffling operation.

4.1.3 RQ4: Variance Reduction

To validate that TAGQuant reduces the activation variance within each group, we compare the standard deviation of the activation values within each group before and after applying TAGQuant. Table 6 shows that TAGQuant reduces the standard deviation of the activation values within each group, compared to RPTQ (Yuan et al., 2023) and QUIK (Ashkboos et al., 2024a).

On a test dataset, GSM8K, we also plot the activation distribution before and after applying RPTQ and TAGQuant in Figure 2. Figure 2 shows that TAGQuant indeed clusters the token-wise distribution better, reducing the variance within each group.

5 Conclusion

In this work, we introduced TAGQuant, a quantization strategy to optimize the grouping for group-wise quantizations. By strategically shuffling channels based on predetermined index pairs, TAGQuant groups channels with similar token-wise distributions. This ensures that large- and small-range values are effectively clustered. Our experiments demonstrated improvement on Llama-3.1 and Phi-4.

Limitations

While TAGQuant is promising, we observe noticable degradation in the multilingual benchmarks, such as MGSM. Also, considering the MXFP4 format is not widely adopted yet, the real impact of TAGQuant on production systems remains to be seen.

Despite these promising results, its impact on different model architectures and activation patterns warrants further investigation. Future directions include extending TAGQuant to mixed-precision settings and exploring alternative grouping strategies that further optimize activation distributions for group-wise quantization.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 Technical Report. *Preprint*, arXiv:2412.08905.

Saleh Ashkboos, Ilia Markov, Elias Frantar, Tingxuan Zhong, Xincheng Wang, Jie Ren, Torsten Hoefler, and Dan Alistarh. 2024a. QUIK: Towards End-to-end 4-Bit Inference on Generative Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3355–3371, Miami, Florida, USA. Association for Computational Linguistics.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024b. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*.

Ziv Bar-Joseph, David K. Gifford, and Tommi S. Jaakkola. 2001. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl_1):S22–S29.

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. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Nips '22, Red Hook, NY, USA. Curran Associates Inc.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 514 others. 2024. The Llama 3 Herd of Models. *Preprint*, arXiv:2407.21783.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation. Zenodo.

Ruihao Gong, Yang Yong, Shiqiao Gu, Yushi Huang, Chengtao Lv, Yunchen Zhang, Dacheng Tao, and Xianglong Liu. 2024. LLMC: Benchmarking Large Language Model Quantization with a Versatile Compression Toolkit. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 132–152, Miami, Florida, US. Association for Computational Linguistics.

Janghwan Lee, Jiwoong Park, Jinseok Kim, Yongjik Kim, Jungju Oh, Jinwook Oh, and Jungwook Choi. 2024. AMXFP4: Taming Activation Outliers with Asymmetric Microscaling Floating-Point for 4-bit LLM Inference. *Preprint*, arXiv:2411.09909.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *MLSys*.

NVIDIA. 2020. NVIDIA A100 GPUs Power the Modern Data Center. https://www.nvidia.com/en-us/data-center/a100/.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof Q&a benchmark. In *First Conference on Language Modeling*.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Stosic Dusan, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, and 14 others. 2023. Microscaling Data Formats for Deep Learning. *Preprint*, arXiv:2310.10537.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8815–8821.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*.

Robert R. Sokal and Charles Duncan Michener. 1958. A statistical method for evaluating systematic relationships. *University of Kansas science bulletin*, 38:1409–1438.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.

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.

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.

Clustering Algorithm	dist	GSM8K
KMeans	L^2	78.5
UPGMA	L^2	**80.0**
UPGMA	L^1	78.8
WPGMA	L^1	77.1
UPGMC	L^1	76.1
WPGMC	L^1	78.0
Nearest point	L^1	77.8
Farthest point	L^1	78.2
Ward variance minimization	L^1	78.1

Table 7: Comparison of various clustering algorithms and distance metrics.

A Validation of Our Clustering Algorithm

Table 7 validates our selection of L^2 distance and UPGMA Clustering Algorithm.

B Hyperparameter Selection

To compare the hyperparameter selection, we vary the sample size and sequence length in Table 8. As expected, when sequence length gets longer and longer, the performance tends to improve. Varying the number of the samples leverages different quality of samples, leading to varying performance numbers.

We also compare the group size in Table 9. Our focus was using small group size k, where 32 is popularly used in the emerging architectures. As expected, larger k value diminishes the effectiveness of TAGQuant.

Sample size	Sequence length	GSM8K	GPQA	DROP	MGSM	avg
128	2048	0.816	0.319	0.580	0.649	**0.5910**
128	1024	0.823	0.317	0.581	0.642	0.5908
128	512	0.813	0.275	0.583	0.640	0.5778
256	2048	0.792	0.288	0.582	0.629	0.5728
64	2048	0.786	0.261	0.588	0.649	0.5710

Table 8: Effect of sample size and sequence length on Llama-3.1-8B-Instruct with INT4 quantization.

	GSM8K	GPQA	DROP	MGSM
QUIK+TAGQuant	76.8	30.6	55.2	61.7
QUIK	77.2	30.1	56.2	62.0

Table 9: Comparison of QUIK and QUIK+TAGQuant on larger group size, such as $k = 128$.