FPTQuant: Function-Preserving Transforms for LLM Quantization

Boris van Breugel [1] **Yelysei Bondarenko** [1] **Paul Whatmough** [1] **Markus Nagel** [1]

Abstract

Large language models (LLMs) are compute- and energy-intensive at inference time. While quantization improves efficiency, naive approaches often degrade performance due to outliers. We introduce FPTQuant, a method that enables effective transformer quantization through four novel, lightweight function-preserving transforms (FPTs): (1) a pre-RoPE transform for queries/keys, (2) a value transform, (3) an MLP scaling transform, and (4) a dynamic residual scaling. These FPTs exploit transformer equivariances to reshape activations without altering model function, require no custom kernels, and add negligible inference overhead. FPTQuant enables static INT4 quantization with minimal overhead and shows SOTA speed-up of up to $3.9\times$ over FP. Empirically, FPTQuant has an excellent accuracy-speed trade-off—it is performing on par or exceeding most prior work and only shows slightly lower accuracy compared to a method that is up to 29% slower.

1. Introduction

Motivation. Inference on large language models (LLMs) incurs a significant compute toll for every token generated, which ultimately costs money and consumes environmental resources. These costs limit the proliferation of LLM use cases, especially on resource constrained edge devices. They are also a significant barrier to furthering AI research and democratization. Therefore, improving LLM inference efficiency is a critical goal. Of the numerous LLM efficiency techniques proposed to date, quantization is by far the most successful; significantly reducing the inference cost by reducing the data bit width across the model. However, the transformer architecture used in LLMs is significantly more challenging to quantize.

Transforms for aggressive quantization. Outliers in transformer weights and activations hinder quantization by forcing a trade-off: either increase range and lose precision near zero, or clip outliers and risk accuracy loss (Bondarenko et al., 2021; Kovaleva et al., 2021; Dettmers et al., 2022; Bondarenko et al., 2023; Sun et al., 2024a). Prior work mitigates this using operations like scaling or rotation that smooth outliers without altering model behaviour pre-quantization. For instance, Xiao et al. (2024) apply per-channel scaling $\mathbf{T} = \mathrm{diag}(\mathbf{s})$ to inputs $\mathbf{X}$ and inverse scaling to weights $\mathbf{W}$, preserving function in full precision: $(\mathbf{XT})(\mathbf{T}^{-1}\mathbf{W}) = \mathbf{XW}$, though not under quantization. We refer to such operations as *function-preserving transforms (FPTs)*, for which we desire the following properties:

- P1 **Function-preservation.** Without any quantization, inserting transform pairs should not change the output (up to computational errors). In practice, this means each FPT typically has an inverse operation.
- P2 **Expressivity.** Transforms with a continuous parametrization and more degrees of freedom are desirable. Continuity means transforms can be optimized directly, e.g. using gradient descent. Extra degrees of freedom offer more flexibility for reducing the quantization error.
- P3 **Compute overhead.** Depending on the FPT type and location, it may be possible to merge (or 'fuse') it into an existing operation in a pretrained model. Non-mergeable FPTs represent a new op in the computational graph, and incur additional overhead, as well as requiring software and/or hardware support.

Contributions. Our contributions are threefold:

1. We introduce FPTQuant: Function-Preserving Transforms for Quantization (Figure 1). FPTQuant includes four novel FPTs that are designed to be both expressive and cheap.
2. We show FPTQuant enables INT4 quantization with minimal overhead. This provides a SOTA speed-up of up to $3.9\times$ over FP.
3. We show FPTQuant has an excellent accuracy-speed trade-off – it is performing on par or exceeding most prior work and only shows slightly lower accuracy compared to a method that is up to 29% slower.

[1]Qualcomm AI Research. Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. Correspondence to: {bvanbreu, ybond, pwhatmou, markusn}@qti.qualcomm.com.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

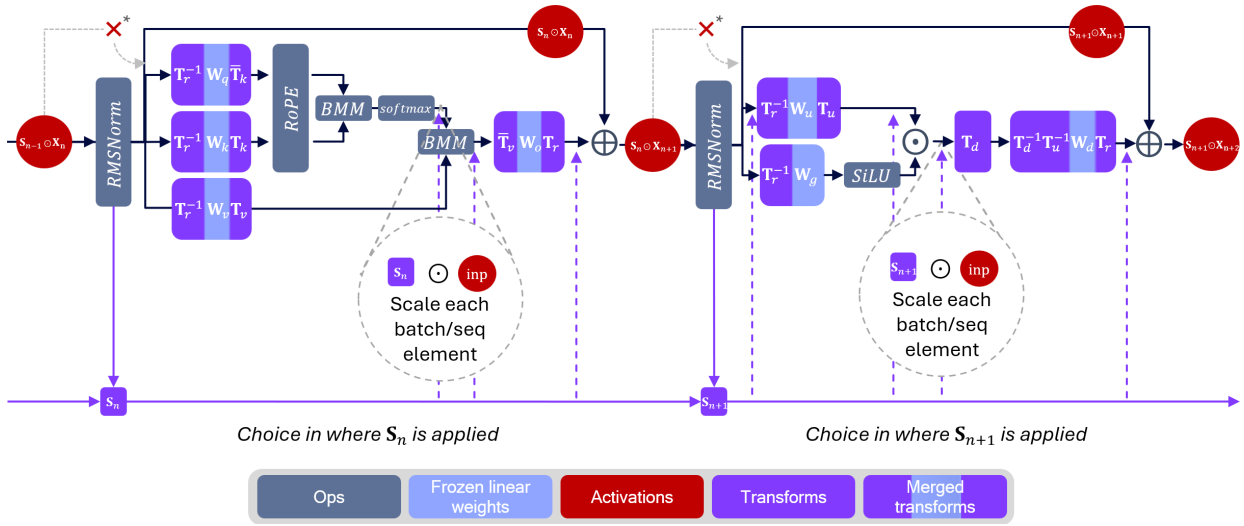

Figure 1. **FPTQuant**. FPTQuant consists of 6 transform types. $(\mathbf{T}_k, \bar{\mathbf{T}}_k)$ is a scale-and-rotate transform merged into the query and key weights; $(\mathbf{T}_v, \bar{\mathbf{T}}_v)$ consists of invertible matrices per head merged into value and output weights; $(\mathbf{T}_u, \mathbf{T}_u^{-1})$ is a per-channel scaler merged into up and down projection weights; transforms $\{\mathbf{S}_n\}_{n=1}^{N}$ ($N = 2\times$number of transformer blocks for typical LLMs) are per-token scalers applied to the residual and within the attention and MLP blocks. The scales $\mathbf{S}_n$ are computed by existing RMSNorms, and in practice means now the RMSNorm is also applied to the residual (versus the original network, see $\times^*$). We also use partly online Hadamard transform $\mathbf{T}_d$ (Ashkboos et al., 2024b) and mergeable rotation matrix $\mathbf{T}_r$ (Liu et al., 2024b).

2. Related work

Nagel et al. (2019) introduced FPTs for CNN quantization, noting that ReLU and per-channel scaling commute, enabling cross-layer weight scaling. In LLMs, Xiao et al. (2024) address activation quantization challenges by shifting outliers to weights via online per-channel scaling. Wei et al. (2023) refine this with a shift and grid search, while Shao et al. (2024) extend it using gradient descent and scaling vectors for queries and keys.

Chee et al. (2024) first explored channel-mixing transforms for weight quantization, later advancing to vector quantization (Tseng et al., 2024). QuaRot (Ashkboos et al., 2024b) uses Hadamard transforms to reduce outliers, while SpinQuant (Liu et al., 2024b) adds trainable rotations to improve performance without inference cost. DuQuant (Lin et al., 2024) applies fixed permutations and block rotations, and FlatQuant (Sun et al., 2024b) introduces efficient Kronecker-product-based transforms. Appendix A compares the cost and effectiveness of these methods, and Appendix B reviews related work on quantization.

3. Method

3.1. Transforms

Equivariances and independencies in pretrained models should be exploited. When an FPT is equivariant with respect to a model operation, it can be applied before or after that operation—potentially enabling mergeability. For instance, Ashkboos et al. (2024a) leverage the equivariance $\mathrm{RMSNorm}(\mathbf{XM}) = \mathrm{RMSNorm}(\mathbf{X})\mathbf{M}$ (for orthogonal $\mathbf{M}$) to insert rotations into residuals and merge them into linear layers, achieving expressivity without added compute. Understanding such properties is key to balancing expressivity (P2) and inference cost (P3). We now introduce four equivariances that motivate four novel transforms.

3.1.1. Pre-RoPE transform (mergeable)

Reducing the bit width of KV cache and queries can significantly reduce memory footprint and computational cost of attention, especially with longer context windows. Unfortunately, we cannot naively merge transforms into the query and key projection weights, because modern LLMs use RoPE positional encodings (Su et al., 2024) (see Appendix C). We introduce a pair of pre-RoPE transforms $(\mathbf{T}_k, \bar{\mathbf{T}}_k)$, where $\mathbf{T}_k$ is applied to keys and $\bar{\mathbf{T}}_k$ can be interpreted as an inverse of $\mathbf{T}_k$, applied to the queries. The transforms consist of scaled 2×2 rotation matrices, and applying these to the query and key weights Pre-RoPE, the attention output remains unchanged. For simplicity we first assume a single attention head. Denoting $i, j \in \mathbb{N}$ as the token indices and RoPE applied to queries and keys as function $f : \mathbb{R}^d \times \mathbb{N} \to \mathbb{R}^d$ with $f(\mathbf{x}, i) = \mathbf{x}\mathbf{R}_{\Theta,i}^{d_{head}}$ (see details Appendix C), the following holds:

Theorem 1. *Let $N = d_{head}/2$, and $\mathbf{R}_n \in O(2)$ and $s_n \in \mathbb{R}$, for $n = 1, ..., N$. Define $\mathbf{T}_k = \mathrm{diag}(\mathbf{s})\,\mathrm{diag}(\{\mathbf{R}_n\}_{n=1}^{N})$ and $\bar{T}_k = \mathrm{diag}(\mathbf{s}^{-1})\,\mathrm{diag}(\{\mathbf{R}_n\}_{n=1}^{N})$. Given query and key weights $(\mathbf{W}_q, \mathbf{W}_k) \in \mathbb{R}^{d_{in} \times d_{head}}$, define $\tilde{\mathbf{W}}_q = \mathbf{W}_q\bar{\mathbf{T}}_k$*

and $\tilde{\mathbf{W}}_k = \mathbf{W}_k\mathbf{T}_k$. *Now it holds:*

$$\langle f(\mathbf{x}_i\tilde{\mathbf{W}}_q, i), f(\mathbf{x}_j\tilde{\mathbf{W}}_k, j)\rangle = \langle f(\mathbf{x}_i\mathbf{W}_q, i), f(\mathbf{x}_j\mathbf{W}_k, j)\rangle$$

In practice, for multi-head attention and grouped-query attention, we can choose an independent transform for each key head. Assuming there are H key heads and mH query heads for some $m, H \in \mathbb{N}$ ($m = 1$ for standard multihead attention), this means we have H independent transforms as above. For the more typical grouped query-attention ($m > 1$), each key head is attended to by multiple query heads, hence we need to repeat the corresponding $\mathbf{T}_k$ transform across these heads. Generally, we can thus write:

$$\mathbf{s}^{(h)} \in \mathbb{R}^d, \mathbf{R}_n^{(h)} \in O(2), \quad \forall h, n \tag{1}$$

$$\mathbf{T}_k^{(h)} = \mathrm{diag}(\mathbf{s}^{(h)})\,\mathrm{diag}(\{\mathbf{R}_n^{(h)}\}_{n=1}^N), \tag{2}$$

$$\mathbf{T}_k = \mathrm{diag}(\{\mathbf{T}_k^{(h)}\}_{h=1}^H) \tag{3}$$

$$\bar{\mathbf{T}}_k = \mathrm{diag}(\underbrace{\bar{\mathbf{T}}_k^{(1)}, ..., \bar{\mathbf{T}}_k^{(1)}}_{m\times}, \bar{\mathbf{T}}_k^{(2)}, ..., \bar{\mathbf{T}}_k^{(H)}), \tag{4}$$

3.1.2. Multihead value transform (mergeable)

Note that the attention probabilities are shape (B, mH, l^1, l^2) and the values (B, mH, l^2, d). The batched matmul (BMM) multiplies these per sample, head, and token, and sum this over l^2. Note d plays no role in this BMM, consequently we are free to apply any invertible transform to the d axis. Note that the different heads in the values are independent, hence we can apply a different transform to each attention head. Newer models use grouped-query attention, which requires a bit of bookkeeping: we need to repeat the inverses per key head, across the corresponding softmax heads. Assuming there are again H value heads (repeated to mH heads) and mH query heads, we can choose any invertible $\mathbf{T}_v^{(h)} \in \mathbb{R}^{d\times d}$, and set:

$$\mathbf{T}_v = \mathrm{diag}(\{\mathbf{T}_v^{(h)}\}_{h=1}^H), \tag{5}$$

$$\bar{\mathbf{T}}_v = \mathrm{diag}(\underbrace{(\mathbf{T}_v^{(1)})^{-1}, ..., (\mathbf{T}_v^{(1)})^{-1}}_{m\times}, (\mathbf{T}_v^{(2)})^{-1}, ..., (\mathbf{T}_v^{(H)})^{-1}), \tag{6}$$

which are merged into respectively $\mathbf{W}_v$ and $\mathbf{W}_o$ weights.

3.1.3. Pseudodynamic residual scaling

In modern transformers, residual connections are typically unnormalized—LayerNorm or RMSNorm is not applied to the residual. This leads to large per-token scale variation, making residuals difficult to quantize and contributing to outliers in downstream layers such as the FFN output (Bondarenko et al., 2023). We propose a lightweight, function-preserving method to normalize residuals dynamically, without altering model outputs or requiring custom kernels. This involves two steps:

Step 1: Normalize residuals. Let $\mathbf{X}_n$ be the residual input, $\mathbf{Y}_n$ the output of the n-th block, and $\mathbf{Z}_n = \mathbf{X}_n + \mathbf{Y}_n$. We compute a per-token scaling factor:

$$\mathbf{S}_n = \mathbf{1} \oslash ||\mathbf{X}_n||_R, \quad \text{where } ||\mathbf{x}||_R = \frac{1}{\sqrt{d}}||\mathbf{x}||_2,$$

and apply it to both residual and output: $\tilde{\mathbf{X}}_n = \mathbf{S}_n \odot \mathbf{X}_n$, $\tilde{\mathbf{Y}}_n = \mathbf{S}_n \odot \mathbf{Y}_n$, which gives a scaled output $\tilde{\mathbf{Z}}_n = \tilde{\mathbf{X}}_n + \tilde{\mathbf{Y}}_n = \mathbf{S}_n \odot \mathbf{Z}_n$.

Step 2: Recursive scaling. Earlier scaling affects later scaling. We can define a recursive relationship:

$$\mathbf{S}_0 = \mathbf{1}, \quad \mathbf{S}_n = \mathbf{S}_{n-1} \oslash ||\tilde{\mathbf{Z}}_{n-1}||_R. \tag{7}$$

This allows us to compute all scales from the normalized outputs alone. Because matrix multiplications, bias-free linear layers, and BMMs commute with per-token scaling, we can push the rescaling deep into the attention and FFN blocks (see Figure 1). This flexibility helps reduce quantization error within those blocks.

Final output. The final transformer output is $\tilde{\mathbf{Z}}_N = \mathbf{S}_N \odot \mathbf{Z}_N$. While we could divide by $\mathbf{S}_N$ to recover the original output, this is unnecessary in practice, as the LM head typically begins with an RMSNorm, which cancels the scale.

3.1.4. Scaler transform on up (mergeable)

The entry-wise product $\odot$ is commutative under matrix multiplication with a diagonal matrix: given $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times d}$ and diagonal $\mathbf{M} \in \mathbb{R}^{d\times d}$, $(\mathbf{A} \odot \mathbf{B})\mathbf{M} = \mathbf{A} \odot (\mathbf{B}\mathbf{M})$. A diagonal $\mathbf{M}$ is useful for us: it can be used to apply a scaling transform to up projections, and merge the inverse into the down projection layer. Note that a scaling vector is also used in (Xiao et al., 2024; Shao et al., 2024; Wei et al., 2023), however these works do not consider that scaling commutes with the entry-wise product, and instead apply a scale to all linear input activations online before quantizing.

3.1.5. Other transforms

In addition to these new transforms, FPTQuant uses a rotation matrix $\mathbf{T}_r$ for rotating the residuals, since this is completely mergeable and Liu et al. (2024b) showed this is effective at reducing activation quantization error. Additionally, the notoriously bad quantization error of activations at the down projection input (Table 3 in (Liu et al., 2024b)) warrants an online transform here; we use a Hadamard transform like in (Ashkboos et al., 2024b; Chee et al., 2024), because it is relatively cheap (Table 3). A schematic illustration of all our transforms applied to a typical transformer block is shown in Figure 1. We optimize transforms locally and globally, see Appendix D.

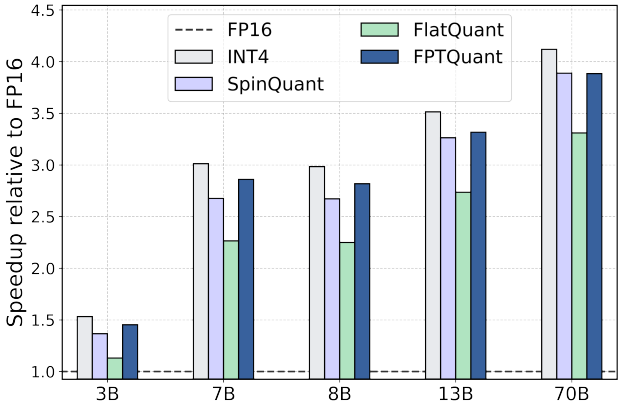

Figure 2. Prefill speedup of FPTQuant on a single transformer block of LLaMA models across different sizes: 3B, 7B, 8B, 13B, and 70B. We use batch size 1 and sequence length 1024.

4. Experiments

Set-up. We evaluate FPTQuant on Llama-2-7B (Touvron et al., 2023), Llama-3-8B (Grattafiori et al., 2024), and Llama-3.2-3B-instruct, covering both standard and edge-device models. We use Wikitext-2 (Merity et al., 2017) and six common-sense reasoning tasks from LM-Harness. We compare against FP, RTN-opt, QuaRot (Ashkboos et al., 2024b), SpinQuant (Liu et al., 2024b), and FlatQuant (Sun et al., 2024b). We use round-to-nearest (RTN) for all methods, since RTN performs competitive to more advanced quantization methods when using transforms (Sun et al., 2024b). See Appendix E for more experimental details.

Runtime Performance. We benchmark on an NVIDIA RTX 3080 Ti using static INT4 quantization and CUTLASS kernels. Figure 2 shows that FPTQuant achieves $2.8\times$ – $3.9\times$ speedup over FP16, outperforming FlatQuant (15-29% slower) and matching or exceeding SpinQuant.

Table 1. **FPTQuant excels for more activation quantizers.** Exploring different activations quantization settings with W4KV4A4 on Llama 3.2 3B instruct. *Linears+KV* is the setting used in (Ashkboos et al., 2024b; Liu et al., 2024b; Sun et al., 2024b). *+BMM input* also quantizes the inputs to the attention batched matmuls (queries and softmax output). *All except residuals* includes all activations except for the residual. We report Wikitext perplexity.

Quant	Method	W4A4KV4	W4A8KV8
FP16		10.48	
Linears+KV	SpinQuant	12.71	11.71
	FlatQuant	11.38	10.68
	FPTQuant	11.71	10.78
+BMM input	SpinQuant	13.16	10.88
	FlatQuant	12.30	10.68
	FPTQuant	13.99	10.56
All except residual	SpinQuant	20.13	11.73
	FlatQuant	18.60	11.49
	FPTQuant	17.17	10.99

Table 2. Comparison of the perplexity score on WikiText-2 (Merity et al., 2017) for Llama-3.2-3B-instruct (L3.2 3B-it), Llama-3-8B (L3 8B) and Llama-2-7B (L2 7B). See Appendix F for accuracy metrics.

#Bits	Method	L3.2 3B-it	L3 8B	L2 7B
16-16-16	FP16	10.48	5.75	5.47
4-8-8	RTN-opt	11.20	7.32	7.11
	QuaRot	10.89	7.04	6.22
	Spinquant	11.03	6.54	5.97
	Flatquant	10.67	6.20	6.46
	FPTQuant	10.65	6.27	5.85
4-8-4	RTN-opt	11.57	7.78	8.04
	QuaRot	11.09	7.29	11.91
	Spinquant	11.47	7.43	6.45
	Flatquant	10.88	6.51	5.91
	FPTQuant	11.12	6.78	6.05
4-4-4	RTN-opt	46.84	543	2220
	QuaRot	12.81	19.72	1218
	Spinquant	12.71	11.04	1461
	Flatquant	11.38	9.55	951
	FPTQuant	11.71	9.74	940

Accuracy and perplexity. In Table 1, we consider Wikitext-2 perplexity of different transforms at W4A4KV4 quantization, varying which activations are quantized. We observe that FPTQuant is better than baselines at the most challenging quantization setting (*All except residual*). Compared to *Linears+KV*, we see a slight drop for FPTQuant relative to baseline when going to *+BMM input*—this is a setting in which queries are quantized, and our mergeable Pre-RoPE transform is not as expressive as the online R_3 (Liu et al., 2024b) or P_h (Sun et al., 2024b) transforms. Table 2 shows results across different bit widths, with only linears and KV cache quantized. FPTQuant consistently outperforms QuaRot and SpinQuant. It is competitive with FlatQuant, despite FPTQuant being 15-29% faster. We observe similar results for reasoning tasks and dynamic quantization, see Appendix F.

5. Discussion

When choosing FPTs, there is a trade-off between expressivity (P2) and cost (P3): more expressive transforms can help reduce quantization error, but incur overhead. By understanding commutation properties of existing operations within the LLM, we have designed most of FPTQuant's transforms to be both expressive, yet mergeable into existing weights. In many settings, the FPTs used by FPTQuant provide a good trade-off between accuracy and speed. For some settings, one may prefer to combine FPTQuant with more expressive, non-mergeable transforms.

References

Ashkboos, S., Croci, M. L., Nascimento, M. G. d., Hoefler, T., and Hensman, J. SliceGPT: Compress Large Language Models by Deleting Rows and Columns, February 2024a. URL http://arxiv.org/abs/2401.15024. arXiv:2401.15024.

Ashkboos, S., Mohtashami, A., Croci, M. L., Li, B., Jaggi, M., Alistarh, D., Hoefler, T., and Hensman, J. QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs, March 2024b. URL https://arxiv.org/abs/2404.00456v1.

Banner, R., Nahshan, Y., Hoffer, E., and Soudry, D. Post-training 4-bit quantization of convolution networks for rapid-deployment. *arXiv preprint arXiv:1810.05723*, 2018.

Bhalgat, Y., Lee, J., Nagel, M., Blankevoort, T., and Kwak, N. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. PIQA: Reasoning about Physical Commonsense in Natural Language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239. Number: 05.

Bondarenko, Y., Nagel, M., and Blankevoort, T. Understanding and overcoming the challenges of efficient transformer quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7947–7969, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.627. URL https://aclanthology.org/2021.emnlp-main.627.

Bondarenko, Y., Nagel, M., and Blankevoort, T. Quantizable Transformers: Removing Outliers by Helping Attention Heads Do Nothing. *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.12929v2.

Bondarenko, Y., Chiaro, R. D., and Nagel, M. Low-Rank Quantization-Aware Training for LLMs, September 2024. URL http://arxiv.org/abs/2406.06385. arXiv:2406.06385.

Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13169–13178, 2020.

Chee, J., Cai, Y., Kuleshov, V., and De Sa, C. M. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36, 2024.

Chen, M., Shao, W., Xu, P., Wang, J., Gao, P., Zhang, K., and Luo, P. Efficientqat: Efficient quantization-aware training for large language models. *arXiv preprint arXiv:2407.11062*, 2024.

Choukroun, Y., Kravchik, E., Yang, F., and Kisilev, P. Low-bit quantization of neural networks for efficient inference. In *ICCV Workshops*, pp. 3009–3018, 2019.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge, March 2018. URL http://arxiv.org/abs/1803.05457. arXiv:1803.05457 [cs].

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.

Dettmers, T., Svirschevski, R., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., and Alistarh, D. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Du, D., Zhang, Y., Cao, S., Guo, J., Cao, T., Chu, X., and Xu, N. Bitdistiller: Unleashing the potential of sub-4-bit llms via self-distillation. *arXiv preprint arXiv:2402.10631*, 2024.

Egiazarian, V., Panferov, A., Kuznedelev, D., Frantar, E., Babenko, A., and Alistarh, D. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*, 2024.

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2020.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. In *International conference on machine learning*, pp. 1737–1746. PMLR, 2015.

Huang, W., Qin, H., Liu, Y., Li, Y., Liu, X., Benini, L., Magno, M., and Qi, X. Slim-llm: Salience-driven mixed-precision quantization for large language models. *arXiv preprint arXiv:2405.14917*, 2024.

Hubara, I., Nahshan, Y., Hanani, Y., Banner, R., and Soudry, D. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518*, 2020.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Jeon, Y., Lee, C., Park, K., and Kim, H.-y. A frustratingly easy post-training quantization scheme for llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14446–14461, 2023.

Kim, S., Hooper, C., Gholami, A., Dong, Z., Li, X., Shen, S., Mahoney, M. W., and Keutzer, K. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

Kovaleva, O., Kulshreshtha, S., Rogers, A., and Rumshisky, A. Bert busters: Outlier dimensions that disrupt transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3392–3405, 2021.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Lee, C., Jin, J., Kim, T., Kim, H., and Park, E. 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, pp. 13355–13364, 2024.

Lee, J. H., Kim, J., Kwon, S. J., and Lee, D. Flexround: Learnable rounding based on element-wise division for post-training quantization. In *International Conference on Machine Learning*, pp. 18913–18939. PMLR, 2023.

Li, Y., Gong, R., Tan, X., Yang, Y., Hu, P., Zhang, Q., Yu, F., Wang, W., and Gu, S. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.

Lin, H., Xu, H., Wu, Y., Cui, J., Zhang, Y., Mou, L., Song, L., Sun, Z., and Wei, Y. DuQuant: Distributing Outliers via Dual Transformation Makes Stronger Quantized LLMs. In *Advances in Neural Information Processing Systems*. arXiv, November 2024. doi: 10.48550/arXiv.2406.01721. URL http://arxiv.org/abs/2406.01721. arXiv:2406.01721.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Liu, Z., Oguz, B., Zhao, C., Chang, E., Stock, P., Mehdad, Y., Shi, Y., Krishnamoorthi, R., and Chandra, V. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 467–484, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.26. URL https://aclanthology.org/2024.findings-acl.26/.

Liu, Z., Zhao, C., Fedorov, I., Soran, B., Choudhary, D., Krishnamoorthi, R., Chandra, V., Tian, Y., and Blankevoort, T. SpinQuant: LLM quantization with learned rotations, May 2024b. URL https://arxiv.org/abs/2405.16406v2.

Liu, Z., Zhao, C., Huang, H., Chen, S., Zhang, J., Zhao, J., Roy, S., Jin, L., Xiong, Y., Shi, Y., Xiao, L., Tian, Y., Soran, B., Krishnamoorthi, R., Blankevoort, T., and Chandra, V. Paretoq: Scaling laws in extremely low-bit llm quantization, 2025. URL https://arxiv.org/abs/2502.02631.

Luo, Y., Gao, Y., Zhang, Z., Fan, J., Zhang, H., and Xu, M. Long-range zero-shot generative deep network quantization. *Neural Networks*, 166:683–691, 2023.

Meller, E., Finkelstein, A., Almog, U., and Grobman, M. Same, same but different: Recovering neural network quantization error through weight factorization. In *International Conference on Machine Learning*, pp. 4486–4495. PMLR, 2019.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Nagel, M., Baalen, M. v., Blankevoort, T., and Welling, M. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Nagel, M., Amjad, R. A., van Baalen, M., Louizos, C., and Blankevoort, T. Up or Down? Adaptive Rounding for Post-Training Quantization, April 2020. URL https://arxiv.org/abs/2004.10568v2.

Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., Van Baalen, M., and Blankevoort, T. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Nagel, M., Fournarakis, M., Bondarenko, Y., and Blankevoort, T. Overcoming oscillations in quantization-aware training. In *International Conference on Machine Learning*, pp. 16318–16330. PMLR, 2022.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, N.-Q., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, 2016.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. WinoGrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://dl.acm.org/doi/10.1145/3474381.

Shao, W., Chen, M., Zhang, Z., Xu, P., Zhao, L., Li, Z., Zhang, K., Gao, P., Qiao, Y., and Luo, P. OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models, March 2024. URL http://arxiv.org/abs/2308.13137. arXiv:2308.13137 [cs].

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Sun, M., Chen, X., Kolter, J. Z., and Liu, Z. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024a.

Sun, Y., Liu, R., Bai, H., Bao, H., Zhao, K., Li, Y., Hu, J., Yu, X., Hou, L., Yuan, C., Jiang, X., Liu, W., and Yao, J. FlatQuant: Flatness Matters for LLM Quantization, October 2024b. URL http://arxiv.org/abs/2410.09426. arXiv:2410.09426.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Tseng, A., Chee, J., Sun, Q., Kuleshov, V., and De Sa, C. QuIP#: Even Better LLM Quantization with Hadamard Incoherence and Lattice Codebooks, February 2024. URL https://arxiv.org/abs/2402.04396v2.

Wei, X., Zhang, Y., Li, Y., Zhang, X., Gong, R., Guo, J., and Liu, X. Outlier Suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling, October 2023. URL http://arxiv.org/abs/2304.09145. arXiv:2304.09145 [cs].

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models, March 2024. URL http://arxiv.org/abs/2211.10438. arXiv:2211.10438 [cs].

Xu, Y., Xie, L., Gu, X., Chen, X., Chang, H., Zhang, H., Chen, Z., Zhang, X., and Tian, Q. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a Machine Really Finish Your Sentence?, May 2019. URL http://arxiv.org/abs/1905.07830. arXiv:1905.07830 [cs].

Zhao, R., Hu, Y., Dotzel, J., De Sa, C., and Zhang, Z. Improving neural network quantization without retraining using outlier channel splitting. In *International conference on machine learning*, pp. 7543–7552. PMLR, 2019.

A. Detailed transform comparison

In Table 3 we include the representation and theoretical cost of existing transforms. In Table 4 we review existing works, the transforms they use, and their placements.

Table 3. **Comparing different transforms.** Cost is measured in terms of a single matrix vector multiplication, xM, where $M \in \mathbb{R}^{n\times n}$ and row vector $x \in \mathbb{R}^n$. Memory is total parameters.

Transform	Cost	Memory	Matrix representation
Scaler	$\mathcal{O}(n)$	n	$A = \text{diag}(\mathbf{s})$, with $\mathbf{s} \in \mathbb{R}^n$, $s_i \neq 0$
Full matrix	$\mathcal{O}(n^2)$	n^2	Any invertible matrix $A \in \mathbb{R}^{n\times n}$
Orthogonal	$\mathcal{O}(n^2)$	n^2	$A \in \mathbb{R}^{n\times n}$ s.t. $AA^T = I$
Rotation	$\mathcal{O}(n^2)$	n^2	$A \in \mathbb{R}^{n\times n}$ s.t. $AA^T = I$ and $\det(A) = 1$
Block diagonal (K blocks)	$\mathcal{O}(\frac{n^2}{K})$	$\frac{n^2}{K}$	$A = \text{diag}(B_1, ..., B_K)$, with invertible $B_k \in \mathbb{R}^{\frac{n}{K}\times\frac{n}{K}}$, $k = 1, ..., K$
Kronecker	$\mathcal{O}(n\sqrt{n})$	$\sim 2n$	$P = P_1 \otimes P_2$, with invertible $P_i \in \mathbb{R}^{n_i\times n_i}$ and $n_1 n_2 = n$ (usually $n_1 \approx n_2 \approx \sqrt{n}$)
Hadamard Transform (HT)	$\mathcal{O}(n \log n)$	0	$H_n = \frac{1}{\sqrt{n}} \bigotimes_{i=1}^{\log_2 n} \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$
Randomized HT (RHT)	$\mathcal{O}(n \log n)$	n	$\text{diag}(\mathbf{s})H_n$, with Bernoulli $\mathbf{s} \in \{-1, +1\}^n$
Block HT (K blocks)	$\mathcal{O}(n \log[n/K])$	0	$A = \text{diag}(\{H_{n/K}\}^K)$

B. Quantization related work

Quantization Neural network quantization has been demonstrated as an effective technique for reducing the model size and improving computational efficiency (Krishnamoorthi, 2018; Nagel et al., 2021). Quantization methods can generally be categorized into post-training quantization (PTQ) and quantization-aware training (QAT) families. PTQ algorithms take a pretrained high precision network and convert it directly into a fixed-point network without the need for the original training pipeline (Banner et al., 2018; Cai et al., 2020; Choukroun et al., 2019; Hubara et al., 2020; Meller et al., 2019; Zhao et al., 2019; Nagel et al., 2019; 2020; Li et al., 2021). These methods are data-free or only require a small calibration dataset, and are generally fast and easy to use. Quantization-aware training (QAT) methods (Gupta et al., 2015; Jacob et al., 2018; Esser et al., 2020; Bhalgat et al., 2020; Nagel et al., 2022) simulate quantization during training, allowing the model to find more optimal solutions compared to PTQ. However, they generally require longer training times, increased memory usage, need for labeled data and hyperparameter tuning.

LLM quantization The excessive training cost and memory usage of traditional QAT methods make them less suitable for quantizing modern LLMs. A few works focus on developing efficient variants of QAT for LLMs include (Liu et al., 2024a; Du et al., 2024; Chen et al., 2024; Dettmers et al., 2024; Xu et al., 2023; Bondarenko et al., 2024). Notably, ParetoQ (Liu et al., 2025) is the only work we are aware of that scale QAT to billions of tokens.

Post-training quantization of LLMs is a challenging task due to presence of strong numerical outliers in weights and activations (Bondarenko et al., 2021; Kovaleva et al., 2021; Dettmers et al., 2022; Bondarenko et al., 2023; Sun et al., 2024a). Various strategies have been explored at tackling these difficulties. These include employing second-order information to mitigate the quantization error (Frantar et al., 2022); emphasizing the importance of so-called "salient" weights that correspond to high-magnitude activations (Dettmers et al., 2023; Lin et al., 2023; Lee et al., 2024); separating outliers and use mixed-precision (Kim et al., 2023; Huang et al., 2024; Egiazarian et al., 2024). Some of the other LLM PTQ methods include (Jeon et al., 2023; Lee et al., 2023; Luo et al., 2023; Chee et al., 2024). Note that many of these PTQ techniques focus primarily on weight quantization and memory size reduction.

C. Proof Theorem 1

RoPE background. RoPE's (Su et al., 2024) aim is to modify the queries and keys, such that the output of the query-key multiplication is dependent on their relative positions. RoPE achieves this by multiplying queries and keys with a time-dependent rotation matrix, i.e. RoPE is a function $f : \mathbb{R}^d \times \mathbb{N} \to \mathbb{R}^d$ with $f(\mathbf{x}, i) = \mathbf{x}\mathbf{R}_{\Theta,i}^{d_{head}}$, where i denotes the token index, Θ the RoPE parameters, and d_{head} the head dimension. Matrix $\mathbf{R}_{\Theta,i}^{d_{head}}$ is a block-diagonal matrix with $N = d_{head}/2$ blocks. Each block n has size 2×2 and denotes a rotation of angle $i\theta_n$ of two dimensions. Denoting a 2-dimensional rotation of angle θ by $\mathbf{R}_\theta^{(2)}$, we can thus write $\mathbf{R}_{\Theta,i}^{d_{head}} = \text{diag}((\mathbf{R}_{i\theta_n})_{n=1}^N)$. As desired, the product between embedded

Table 4. **Transformations in LLM quantization literature.** (R)HT: (randomized) hadamard transform. CW: Channel-wise. E2E: end-to-end training, with either original [Label] or student-teacher [ST] loss.

Work	Transform style	Transform location	Mergeable	Optimization
Smoothquant (Xiao et al., 2024)	CW Scaler	after RMSNorm	True	Local L_∞
Outlier supp+(Wei et al., 2023)	CW Affine	after RMSNorm	True	Grid search
OmniQuant(Shao et al., 2024)	CW Affine	after RMSNorm and $\mathbf{W}_v$	True	Block-wise
	CW Scaler	on embedded queries and keys	False†	Block-wise
QuaRot (Ashkboos et al., 2024b)	HT	on embedded queries and keys, before $\mathbf{W}_o, \mathbf{W}_d$ ‡	False	-
	HT	on values	True	-
	RHT	residual	True	-
SpinQuant (Liu et al., 2024b)	RHT	on embedded queries and keys (R_3), before $\mathbf{W}_d$ (R_4)	False	-
	Rotation	residual (R_1)‡, values R_2	True	E2E[Label]
FlatQuant (Sun et al., 2024b)	Kronecker	after RMSNorm (P_a and P_{ug}), before $\mathbf{W}_o$ (P_o), before $\mathbf{W}_d$ (P_d),	False	E2E[Label]
	Full	embedded queries and keys (P_h)	False	E2E[Label]
	Full	values (P_v)	True	E2E[Label]
DuQuant (Lin et al., 2024)	Scaler-permute-rotate	linear weights/inputs	False	Iterative greedy
FPTQuant (us)‡	PreRoPE	merged into $\mathbf{W}_q, \mathbf{W}_k$	True	Local L_p+E2E[ST]
	Full per head	merged into $\mathbf{W}_v, \mathbf{W}_o$	True	Local L_p+E2E[ST]
	CW Scaler	merged into $\mathbf{W}_u, \mathbf{W}_d$	True	Local L_p+E2E[ST]
	Sequence Scaler	residual, on softmax output and before $\mathbf{W}_d$	False	-

† Authors claim channel-wise scaling of queries and keys can be merged, which does not hold for non-additive positional encodings (e.g. RoPE). ‡We also use SpinQuant's mergeable R1 rotation, and non-mergeable HT at mm.

keys and queries depends only on their relative, not absolute, position: $\langle f(\mathbf{q}_i, i), f(\mathbf{k}_j, j)\rangle = \mathbf{q}_i \mathbf{R}^d_{\Theta, i-j} \mathbf{k}_j^\intercal$. We develop transforms that we can apply to queries and keys, yet do not alter the output of the attention softmax. We design these to commute with RoPE's $\mathbf{R}^d_{\Theta,i}$ for all i, so that they can be applied before RoPE and merged into $\mathbf{W}_q$ and $\mathbf{W}_k$.

Theorem 1 Let $N = d_{head}/2$, and $\mathbf{R}_n \in O(2)$ and $s_n \in \mathbb{R}$, for $n = 1, ..., N$. Define $\mathbf{T}_k = \mathrm{diag}(\mathbf{s})\,\mathrm{diag}(\{\mathbf{R}_n\}_{n=1}^N)$ and $\bar{T}_k = \mathrm{diag}(\mathbf{s}^{-1})\,\mathrm{diag}(\{\mathbf{R}_n\}_{n=1}^N)$. Given query and key weights $(\mathbf{W}_q, \mathbf{W}_k) \in \mathbb{R}^{d_{in} \times d_{head}}$, define $\tilde{\mathbf{W}}_q = \mathbf{W}_q \bar{\mathbf{T}}_k$ and $\tilde{\mathbf{W}}_k = \mathbf{W}_k \mathbf{T}_k$. Now it holds:

$$\langle f(\mathbf{x}_i \tilde{\mathbf{W}}_q, i), f(\mathbf{x}_j \tilde{\mathbf{W}}_k, j)\rangle = \langle f(\mathbf{x}_i \mathbf{W}_q, i), f(\mathbf{x}_j \mathbf{W}_k, j)\rangle$$

Proof. First, let us prove that $\mathbf{T}_k$ commutes with $\mathbf{R}^{d_{head}}_{\Theta,i}$ for any i and Θ. Both are block diagonal (with blocks of size 2×2), so we can treat each block individually. For the individual blocks of $\mathbf{R}^d_{\Theta,i}$ and $\mathbf{T}_k$, write $\mathbf{R}_{i\theta_n}$ and $w_n \mathbf{R}_{\phi_n}$. Trivially, scalars commute with matrices, i.e. $w\mathbf{A} = \mathbf{A}w$ for any matrix $\mathbf{A}$ and $w \in \mathbb{R}$. Additionally, 2×2 rotations commute, hence $\mathbf{R}_{i\theta_n} w_n R_{\phi_n} = w_n R_{\phi_n} R_{i\theta_n}$. As this holds for all blocks, $\mathbf{R}^{d_{head}}_{\Theta,i} \mathbf{T}_k = \mathbf{T}_k \mathbf{R}^{d_{head}}_{\Theta,i}$.

Second, note that $\bar{\mathbf{T}}_k \mathbf{T}_k^\intercal = I$,[1] since weights and rotations cancel out. Replacing $\mathbf{W}_q, \mathbf{W}_k$ by respectively $\tilde{\mathbf{W}}_q$ and $\tilde{\mathbf{W}}_k$

[1]For single-headed attention, $\bar{\mathbf{T}}_k = \mathbf{T}_k^{-1}$, but this is not true for grouped query attention (Eq. 1which is typically used in LLMs.

thus gives attention values:

$$\begin{aligned}\langle f(\mathbf{x}_i\tilde{\mathbf{W}}_q, i), f(\mathbf{x}_j\tilde{\mathbf{W}}_k, j)\rangle &= \langle \mathbf{x}_i\mathbf{W}_q\bar{\mathbf{T}}_k\mathbf{R}^d_{\Theta,m}, \mathbf{x}_j\mathbf{W}_k\mathbf{T}_k\mathbf{R}^d_{\Theta,n}\rangle \\ &= \langle \mathbf{x}_i\mathbf{W}_q\mathbf{R}^d_{\Theta,i}\bar{\mathbf{T}}_k, \mathbf{x}_j\mathbf{W}_k\mathbf{R}^d_{\Theta,j}\mathbf{T}_k\rangle \\ &= \langle \mathbf{x}_i\mathbf{W}_q\mathbf{R}^d_{\Theta,i}\bar{\mathbf{T}}_k\mathbf{T}_k^\intercal, \mathbf{x}_j\mathbf{W}_k\mathbf{R}^d_{\Theta,j}\rangle \\ &= \langle f(\mathbf{x}_i\mathbf{W}_q, i), f(\mathbf{x}_j\mathbf{W}_k, j)\rangle,\end{aligned}$$

as desired. □

Remark 2. Note: $\mathbf{R}^d_{\Theta,i}$ overall is a rotation matrix, however rotation matrices generally do not commute unless they share the same axes of rotations. This motivates a transform that uses the same block structure. Note also that a block-wise orthogonal matrix would not suffice, since orthogonal matrices that are not rotations (i.e. that contain also a reflection) do not commute with rotations.

D. Transform optimization

D.1. Local optimization

To reduce the worst outliers, we optimize all transforms first locally and independently. We minimize the L_p norm of each transform's merged weights and use gradient descent. For example, for the rotation transform R1 (Liu et al., 2024b), we optimize:

$$\min_{\mathbf{T}_r} \sum_{i=1}^{\#layers} \Big[\sum_{\mathbf{W}\in\{\mathbf{W}^i_q,\mathbf{W}^i_k,\mathbf{W}^i_v,\mathbf{W}^i_u,\mathbf{W}^i_g\}} ||\mathbf{T}_r^{-1}\mathbf{W}||_p + \sum_{\mathbf{W}\in\{\mathbf{W}^i_o,\mathbf{W}^i_d,\mathbf{W}^i_g\}} ||\mathbf{W}\mathbf{T}_r||_p \Big], \tag{8}$$

whilst for the PreRoPE transforms $\mathbf{T}^i_q$ and $\mathbf{T}^i_k$ of layer i with shared parameters Φ, we just minimize:

$$\min_{\Phi} ||\mathbf{W}^i_q\mathbf{T}^i_q||_p + ||\mathbf{W}^i_k\mathbf{T}^i_k||_p.$$

Since $\mathbf{T}_r$ affects all linear layers, we optimize it first (Eq 8). Locally optimized transforms are merged into the weights, after which the next transform is optimized and so forth. We set $p = 4$, following LR-QAT (Bondarenko et al., 2024) who showed L_4 is good for determining the quantization grid.

D.2. End-to-end optimization

We follow (Liu et al., 2024a) and use student-teacher training for reducing the quantization error further. We train the student (the quantized model with transforms) to approximate the teacher (the unquantized FP model), with Jensen-Shannon Divergence loss:

$$\min_{\Phi} \mathbb{E}_X[JSD[f(\mathbf{X}), f_\Phi(\mathbf{X})], \tag{9}$$

where f denotes the original model, f_Φ the quantized model, and Φ includes both the transformation and the quantization grid parameters. It is essential we include the latter—the grid cannot adapt to the transformed input otherwise.

The end-to-end student-teacher approach deviates from SpinQuant (Liu et al., 2024b) and FlatQuant (Sun et al., 2024b). SpinQuant uses the LLM's original next-token prediction loss. Compared to next-token prediction, student-teacher training: 1) provides more signal (i.e., for each data point and sequence element, a full vector of probabilities, vs. a single label), and in turn this 2) decreases overfitting. This is an important reason to avoid next-token prediction loss: although we are working with transforms that in the absence of quantization do not change the model output, the combination of the large number of parameters $|\Phi|$ and the quantization non-linearities (i.e. rounding), actually provide the transformed and quantized model with enough capacity to overfit. FlatQuant optimizes the mean squared error (MSE) per transformer block. This is not directly applicable for transforms that may affect multiple blocks at once, for example a rotation applied to the residual and merged into all linears, as used here and by (Ashkboos et al., 2024b; Liu et al., 2024b).

E. Experimental details

E.1. Models and Tasks

We evaluate on Llama-2-7B, Llama-3-8B, and Llama-3.2-3B-instruct. We evaluate perplexity on Wikitext-2 and use LLM-harness to evaluate the same Common Sense Reasoning tasks used in FlatQuant (Sun et al., 2024b): Piqa (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), ARC-e and ARC-c (Clark et al., 2018), and LAMBADA (Paperno et al., 2016). .

E.2. Baselines

We compare against:

- **FP**: Full-precision baseline.
- **RTN-opt**: RTN with optimized ranges.
- **QuaRot**, **SpinQuant**, **FlatQuant**: State-of-the-art FPT-based methods.

All methods use RTN quantization, as it performs well with transforms (Liu et al., 2024b; Sun et al., 2024b).

E.3. Training Setup

To understand the value of the transforms we introduce, we use end-to-end student-teacher training for all methods, including baselines. This means we train the quantization grid for RTN and QuaRot, thereby getting much better results than in the original paper (Ashkboos et al., 2024b). Similarly, we get better results for SpinQuant (Liu et al., 2024b), since we found that student-teacher training does better than the original next-token prediction loss. All methods are trained for 1024 steps with batch size 16 and sequence length 2048 on Wikitext-2.

E.4. Runtime Setup

We implement all methods in PyTorch with CUDA 12.1, using INT4 CUTLASS kernels from the QuaRot repository[2]. Runtime is measured on a single transformer block (due to memory limits) on an NVIDIA RTX 3080 Ti. Each configuration is run 1000 times, and we report mean speed-up over FP16.

F. Additional experiments

F.1. Table 1: extended

Set-up. In Table 5 we extend Table 2 to include Common Sense Reasoning tasks using LM-Harness. We use the same tasks as used in FlatQuant (Sun et al., 2024b): Piqa (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), ARC-e and ARC-c (Clark et al., 2018), and LAMBADA (Paperno et al., 2016).

Results. Overall, we see FPTQuant performs competitive—outperforming baselines except FlatQuant, but being faster than FlatQuant. For the very challenging setup of W4A4KV4 and Llama-2-7B at W4A8KV4 the gap is sometimes larger, especially for zero-shot accuracy.

F.2. Dynamic quantization

Setup. We repeat the previous experiment with W4A4KV4 in a dynamic quantization setting. This is identical to the FlatQuant (Sun et al., 2024b) setup, from which we report baseline results.

Results. We observe (Table 6) that FPTQuant outperforms all baselines except FlatQuant. However, FPTQuant is up to 29% faster than FlatQuant.

[2] https://github.com/spcl/QuaRot

Table 5. **Table 2 extended.** Comparison of the perplexity score on WikiText-2 (Merity et al., 2017) and averaged accuracy on 6 Zero-shot Common Sense Reasoning tasks for Llama-3.2-3B-instruct (L3.2 3B-it), Llama-3-8B (L3 8B) and Llama-2-7B (L2 7B).

#Bits (W-A-KV)	Method	L3.2 3B-it Wiki (↓)	L3.2 3B-it 0-shot[6] Avg.(↑)	L3 8B Wiki (↓)	L3 8B 0-shot[6] Avg.(↑)	L2 7B Wiki (↓)	L2 7B 0-shot[6] Avg.(↑)
16-16-16	FP16	10.48	65.63	5.75	73.33	5.47	69.79
4-8-8	RTN-opt	11.20	61.09	7.32	67.35	7.11	56.93
	QuaRot	10.89	63.12	7.04	67.60	6.22	63.43
	Spinquant	11.03	63.28	6.54	71.60	5.97	66.01
	Flatquant	10.67	65.04	6.20	72.11	6.46	62.07
	FPTQuant	10.65	64.00	6.27	72.72	5.85	65.96
4-8-4	RTN-opt	11.57	58.92	7.78	64.73	8.04	48.09
	QuaRot	11.09	63.18	7.29	66.71	11.91	39.71
	Spinquant	11.47	59.04	7.43	65.56	6.45	59.28
	Flatquant	10.88	63.69	6.51	70.83	5.91	66.04
	FPTQuant	11.12	62.42	6.78	69.46	6.05	62.68
4-4-4	RTN-opt	46.84	31.16	543	30.04	2220	29.98
	QuaRot	12.81	54.38	19.72	42.76	1218	30.21
	Spinquant	12.71	54.88	11.04	54.58	1461	29.24
	Flatquant	11.38	61.00	9.55	61.43	951	29.70
	FPTQuant	11.71	59.27	9.74	52.96	940	29.65

Table 6. **Dynamic quantization.** We run the dynamic quantization experiment from FlatQuant (Table 1 and Table 2) (Sun et al., 2024b), reporting their results for baselines. FPTQuant outperforms baselines, except FlatQuant, yet FlatQuant is up to 29% slower.

Method	LLaMA-2 7B Wiki (↓)	LLaMA-2 7B 0-shot[6] Avg.(↑)	LLaMA-3 8B Wiki (↓)	LLaMA-3 8B 0-shot[6] Avg.(↑)
FP16	5.47	69.79	5.75	73.33
SmoothQuant	83.1	-	210.2	-
QuaRot	8.56	57.73	10.60	61.34
SpinQuant	6.14	63.52	7.96	66.98
FlatQuant	5.79	67.96	6.98	71.23
FPTQuant	5.97	66.06	7.17	68.09