OptRot: Mitigating Weight Outliers via Data-Free Rotations for Post-Training Quantization

Advait Gadhikar[*]
CISPA
Saarbrücken, Germany

Riccardo Grazzi
Microsoft Research
Cambridge, UK

James Hensman
Microsoft Research
Cambridge, UK

Abstract

We introduce OptRot, a data-free preprocessing method to learn fusible rotations for post-training quantization of language models. OptRot reduces weight outliers by finding rotations which minimize the element-wise fourth power of the rotated weights. We show how reducing weight outliers can provably improve weight quantization performance and how OptRot rotations can outperform both Hadamard rotations and rotations learned by the data-dependent method SpinQuant.

1 Introduction

Increasing model size has enabled LLMs to perform a range of tasks [1, 2], encouraging practitioners to design huge models with billions of parameters. Post Training Quantization (PTQ) is a widely adapted strategy for compressing these models by lowering their precision, while limiting the drop in performance [3] to enable efficient inference. Scalar quantization is the most popular PTQ approach and relies on mapping each parameter value to a point on a finite precision grid, determined by the bit-width. Outliers prevent the finite grid from uniformly covering all values and can lead to large quantization errors. Hence, outlier reduction has increasingly received attention as a crucial pre-processing step [4, 5] for PTQ which uses a uniform quantization grid with algorithms like simple Round-to-Nearest (RTN) or the more powerful GPTQ [6].

Recent work has applied rotations to weight matrices to mitigate outliers in weights and activations while keeping the network functionally equivalent. Hadamard rotations have shown to significantly improve the performance of GPTQ [4, 7]. To minimize overhead, these rotations can be materialized online efficiently with the Walsh-Hadamard transform, and in some cases they can also be fused with model weights. Fused rotations can be learned without increasing inference cost. SpinQuant [8] and Kurtail [9] learn fused rotations to improve mainly activation quantization, while FlatQuant [10] learns invertible linear maps which cannot be fused, instead of rotations, to reduce outliers. ButterflyQuant [11] learns orthogonal transforms parameterized by Givens rotations. SpinQuant and FlatQuant are tailored for quantization via RTN, while learning transformations that improve quantization with GPTQ when compared to Hadamard rotations remains challenging.

In this work we introduce OptRot, a method that can learn rotations which can provably improve weight quantization performance for both RTN and GPTQ. We leverage the theoretical framework on incoherence processing introduced by QuIP [5] to highlight the connection between rotations and the weight quantization error. The quantization error is bounded by the incoherence of the weights and that of the covariance or Hessian of the input activations. The presence of outliers increases incoherence, worsening the bound. Hadamard rotations improve this incoherence as shown by [7], but this bound can be further improved by explicitly optimizing the incoherence objective.

OptRot learns rotations by minimizing a smooth proxy for the weight incoherence of the model: the element-wise fourth power of the rotated weights. We show that OptRot achieves better incoherence

[*]Work done during an internship at Microsoft.

ML For Systems workshop at Neural Information Processing Systems (NeurIPS 2025).

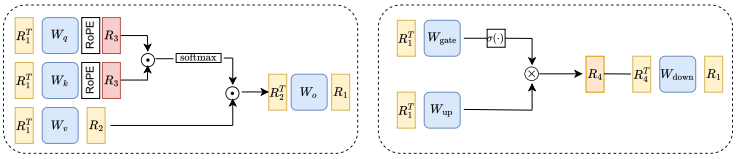

Figure 1: Rotations applied by QuaRot, SpinQuant and OptRot to improve PTQ. $R_1, R_1^\top, R_2, R_2^\top$ and $R_4^\top$ (yellow) are fusible rotations. R_3 (red) and R_4 (orange) are online rotations. R_1 is shared across layers. In OptRot and SpinQuant R_1 and R_2 are learned without increasing inference cost.

and weight quantization performance than QuaRot, SpinQuant and QuIP#. Differently from prior work, OptRot is data-free and (except for Kurtail) quantization-agnostic: it does not require a calibration set or the quantization scheme to learn rotations. Additionally, OptRot can learn rotations by optimizing a subset of the weights, without loading the entire model on GPU. We find that OptRot is also competitive with SpinQuant with activation quantization in the W4A8 regime. Yet, for the more extreme W4A4 case, we find that optimizing weight incoherence with OptRot introduces a trade-off which lowers activation quantization performance in comparison to SpinQuant.

2 OptRot

Let $W \in \mathbb{R}^{m\times n}$ be one original weight matrix of the LLM receiving input $x \in \mathbb{R}^n$ from the previous layer, and $H = \mathbb{E}[xx^\top] \in \mathbb{R}^{n\times n}$ be the uncentered input covariance or Hessian with SVD $H = Q\Lambda Q^T$. Let also $\mathrm{tr}(A)$ and $||A||$, $||A||_F$ be the trace, spectral norm and Frobenius norm of the matrix A respectively. Following [5], we define the incoherence of W and H as

$$\mu_W := w_{\max}\sqrt{mn}/||W||_F, \quad \mu_H := q_{\max}\sqrt{n}, \quad w_{\max} = \max_{ij}|W_{ij}|, \quad q_{\max} = \max_{i,j}|Q_{ij}|\,. \tag{1}$$

The weight incoherence, μ_W, measures how heavy-tailed the distribution of weights is. It has minimum $\mu_W = 1$, which is achieved when all weights are constant ($W_{i,j} = c \in \mathbb{R}$), while outlier weights can greatly increase this value. The Hessian incoherence μ_H has a similar role and also has minimum 1.

QuIP# [7] exploits incoherence processing, which decreases μ_W and μ_H by transforming $\tilde{W} \leftarrow UWV^T$ and $\tilde{H} \leftarrow VHV^T$ before quantization, where U, V are random Hadamard matrices. This approach introduces overhead during inference since the activations must be rotated prior to matrix multiply. QuaRot also uses Hadamard rotations, but exploits the rotational invariance of LLMs to reduce the overhead by fusing some of the rotations with the weights. These rotations can, in part, be learned without additional overhead during inference, as done in SpinQuant. We provide a schematic of the rotations used in Figure 1. Our method, OptRot, learns two fusible rotations (R_1 and R_2 in Figure 1) to minimize the weight incoherence. This offers a simple data-free method to learn rotations, while minimizing also the Hessian incoherence would require a calibration dataset. Instead of directly minimizing $\sum_{\tilde{W}} \mu_{\tilde{W}}$ where $\tilde{W}$ is a rotated weight matrix, we replace the non-smooth incoherence $\mu_{\tilde{W}}$ with $\|\mathrm{Vec}(\tilde{W})\|_p^p = \sum_{i,j}|\tilde{W}_{i,j}|^p$ where Vec flattens matrices into vectors, since high p-norm provides a smooth approximation to the infinity norm. Concretely, for an LLM of the Llama family with L layers we solve the following problem over our smooth incoherence loss:

$$\min_{R_1, R_{2,1},\ldots,R_{2,L}} \sum_{i=1}^{L}\sum_{s} \|\mathrm{Vec}(\tilde{W}^{(l,s)})\|_p^p\,, \tag{2}$$

where $\tilde{W}^{(l,s)} = R_1^\top W^{(l,s)}$ for $s \in \{\mathrm{q}, \mathrm{k}, \mathrm{gate}, \mathrm{up}\}$ and $\tilde{W}^{(l,\mathrm{v})} = R_1^\top W^{(l,\mathrm{v})} R_{2,l}$, $\tilde{W}^{(l,\mathrm{o})} = R_{2,l}^\top W^{(l,\mathrm{o})} R_{2,l}$, $\tilde{W}^{(l,\mathrm{down})} = R_4^\top W^{(l,\mathrm{down})} R_1$. To minimize the objective over the space of rotations, we use Cayley Gradient Descent [12] (same as SpinQuant). We set $p = 4$ in our experiments. Preliminary results showed no improvement with larger p.

2.1 Improving the Quantization Error by Reducing Incoherence

The quality of a quantized weight $\hat{W}$ in isolation can be measured by the following objective, which measures how well we can recover the outputs of the original layer.

$$\mathcal{L}(\hat{W}, H) = \mathrm{tr}\left((\hat{W} - W)H(\hat{W} - W)^T\right) = \mathbb{E}\left[||(\hat{W} - W)x||^2\right], \tag{3}$$

RTN. A simple strategy to quantize a weight matrix is b-bits round-to-nearest (RTN). First, each element of the weight matrix is scaled by applying the function $g(x;s) = \frac{2^b-1}{2}\left(\frac{x}{s}+1\right)$ (assuming symmetric quantization), where $s > 0$ is a scale parameter. Second, it is mapped to the nearest value of the quantization grid via the function $f(x) = \arg\min_{c\in\{0,\dots,2^b-1\}} |x-c|$. The resulting values are then stored and during inference g^{-1} is applied to retrieve the approximate weights $\hat{W}$. Combining all steps we obtain $\hat{W}_{\text{RTN}} = Q(W;s) = g^{-1}(f(g(W;s));s)$, where we extended g, g^{-1}, f to element-wise functions applied to the entire matrix. The scale s can be set to $w_{\max}$ so that all weights are inside the quantization grid, but the max is usually computed and applied over a small contiguous group of weights to mitigate outliers with a small increase in memory (as scales have to be stored). We show (Theorem 1) that if $s = w_{\max}$ the RTN error can be bounded as

$$\mathcal{L}(\hat{W}_{\text{RTN}}, H) \leq \frac{\mu_W^2}{(2^b-1)^2}\lambda_{\max}(H)||W||_F^2\,. \tag{4}$$

GPTQ. The GPTQ algorithm finds the quantized weight which approximately minimizes $\mathcal{L}$ by exploiting the LDL decomposition of the matrix H. In particular, as shown by [7], it finds $\hat{W}_{\text{GPTQ}} = Q(W+(W-\hat{W}_{\text{GPTQ}})U)$ where U is the strictly upper triangular matrix from the LDL decomposition of the hessian $H = (U+I)D(U+I)^\top$. Obtaining a proper worst-case bound for GPTQ is more challenging than for RTN since the corrections applied before discretization via Q can push some values far outside the quantization grid. Despite this, with some changes to the GPTQ algorithm (not used in practice) which include the use of stochastic rounding, the worst-case error can be bounded with probability at least $1-\delta$ as shown in Theorem 14 in [5]:

$$\mathcal{L}(\hat{W}_{\text{GPTQ}}, H) \leq \frac{\mu_H^2\mu_W^2}{n^2(2^b-3)^2}\,\text{tr}\left(H^{1/2}\right)^2||W||_F^2\log\left(\frac{4mn}{\delta}\right)^2. \tag{5}$$

Improving error bounds with rotations. Applying rotations to W or H only affects the error upper bounds via two terms: the weight incoherence μ_W, affecting both GPTQ and RTN, and the Hessian incoherence μ_H, affecting only GPTQ. Rotations can modify $w_{\max}, q_{\max}$, while $\|W\|_F, \text{tr}(H)$ and $\text{tr}(H^{1/2})$ are rotation-invariant. OptRot learns rotations that aim at minimizing μ_W.

Figure 2: Weight incoherence μ_W optimized by OptRot (top row), Hessian incoherence μ_H (middle row) and their product $\mu_H\mu_W$ (bottom row) for Llama-3.1-8B. Note that μ_H is identical for QuaRot and QuIP#, and QuIP# has a higher inference cost due to the additional online Hadamard rotations.

3 Experiments

We conduct experiments on the Llama-3 series of models, for weight and activation quantization. Weights are quantized with GPTQ, calibrated on the C4 [13] dataset, and activations are quantized with RTN. We report results on Wikitext [14] and six zero-shot commonsense reasoning benchmarks: Piqa [15], Hellaswag [16], Arc-E and Arc-C [17], Lambada [18] and Winogrande [19]. We also report the KL divergence, on the C4 dataset, between the quantized and original model. We present two versions of our method, OptRot (all) learns rotations by optimizing incoherence over all weights. OptRot (top-50) is a cheaper variant which chooses the top 50 weights with the largest incoherence loss to learn rotations. Due to space constraints we defer some results to the appendix.

Table 1: Results for weight-only quantization at 4-bits with GPTQ.

Method	Llama-3.2-1B			Llama-3.2-3B			Llama-3.1-8B		
	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓
FP16	56.77	9.76	0	64.72	7.81	0	70.29	6.24	0
No Rotation	49.43	13.86	0.36	56.28	11.78	0.35	67.39	7.47	0.23
QuaRot	55.26	10.79	0.136	**63.48**	8.36	0.097	69.17	6.72	0.0926
SpinQuant	54.9	10.73	0.137	63.05	8.37	0.096	69.46	6.71	0.0925
OptRot (top-50)	55.53	10.68	**0.123**	**63.48**	8.32	**0.093**	**69.5**	6.65	**0.0849**
OptRot (all)	**55.65**	**10.62**	0.125	63.41	**8.28**	**0.093**	**69.5**	**6.64**	0.0866

OptRot improves the RTN and GPTQ error bounds. Figure 2 (top row) plots the weight incoherence μ_W. SpinQuant and QuaRot achieve similar incoherences upon rotation as seen by the almost overlapping curves. QuIP# can further improve the incoherence on key, out and up-projection layers but it employs additional online rotations and hence is costlier to run. OptRot generally finds the smallest weight incoherence and hence the lowest RTN error bound (see Equation 4), with the most notable improvements in out, key and down-projection layers. The down-projection layer in particular is known to be affected by outliers [20], here OptRot consistently achieves lower weight incoherence.

Minimizing weight incoherence also affects the Hessian incoherence (Figure 2 middle row), as both the weights and their inputs are rotated. In comparison to Hadamard rotations with QuaRot or QuIP#, OptRot increases μ_H in key and out-projection layers. This suggests a trade-off between μ_W and μ_H: decreasing μ_W seems to worsen μ_H. Interestingly, SpinQuant, which learns rotations that minimize the activation quantization error, does not improve μ_H in comparison to Hadamard rotations.

The GPTQ quantization error bound in Equation 5 is controlled by the product $\mu_H \mu_W$ plotted in Figure 2 (bottom row). OptRot obtains a smaller product in all layers, except a few up-projection layers. This suggests that optimizing the weight incoherence with OptRot should improve the error in majority of the layers, leading to better downstream performance.

OptRot improves weight-only quantization. Results with GPTQ in Table 1 (4-bits) show that OptRot consistently outperforms other methods, even with the cheaper top-50 version. This result also holds for in the absence of online rotations (see Table 4) and at 3-bits (see Table 7). With the underperforming RTN quantization, SpinQuant still outperforms OptRot (see Appendix G).

Activation Quantization. OptRot matches or outperforms SpinQuant in the A8W4 setting (see Table 5), which can efficiently serve large models without a significant drop in performance [21]. In contrast, in the A4W4 setting, which sees a significant drop in performance compared to the full precision model, OptRot performs worse than SpinQuant and often even QuaRot. We hypothesize that this can be related to the increased Hessian incoherence (Figure 2 middle row).

Conclusion. We introduce OptRot, a method to learn rotations in a data-free manner for post-training weight quantization. Building on [7], we leverage the fact that rotations control the weight and Hessian incoherence which in turn bound the weight quantization error for RTN and GPTQ. OptRot optimizes the weight incoherence to improve these bounds, which translates to improved downstream performance for weight-only quantization. When including activation quantization, while OptRot is competitive with SpinQuant in the A8W4 regime, it underperforms in the A4W4 one.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, , et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[3] Zifei Xu, Alexander Lan, Wanzin Yazar, Tristan Webb, Sayeh Sharify, and Xin Wang. Scaling laws for post-training quantized large language models. *CoRR*, 2024.

[4] Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

[5] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36:4396–4429, 2023.

[6] 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.

[7] Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024.

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

[9] Mohammad Sadegh Akhondzadeh, Aleksandar Bojchevski, Evangelos Eleftheriou, and Martino Dazzi. Kurtail: Kurtosis-based llm quantization. *arXiv preprint arXiv:2503.01483*, 2025.

[10] Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024.

[11] Bingxin Xu, Zhen Dong, Oussama Elachqar, and Yuzhang Shang. Butterflyquant: Ultra-low-bit llm quantization through learnable orthogonal butterfly transforms. *arXiv preprint arXiv:2509.09679*, 2025.

[12] Jun Li, Li Fuxin, and Sinisa Todorovic. Efficient riemannian optimization on the stiefel manifold via the cayley transform. *arXiv preprint arXiv:2002.01113*, 2020.

[13] 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. *arXiv e-prints*, 2019.

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

[15] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439, 2020.

[16] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[17] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[18] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 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)*, pages 1525–1534, 2016.

[19] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740, 2020.

[20] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332, 2022.

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

A Round to Nearest Error Bound

Round-to-Nearest (RTN) is a simple quantization method which simply rounds every weight element to its nearest discrete value on the quantization grid. For a bit-width b, the weight values must be scaled such that $0 \leq W_{ij} \leq 2^b - 1$ assuming symmetric quantization.

Each element is rescaled as $W_{ij} \rightarrow \frac{2^b-1}{2}\left(\frac{W_{ij}}{w_{\max}} + 1\right)$ with $w_{\max} = \max_{i,j}|W_{i,j}|$ followed by rounding to the nearest integer. The error achieved by the RTN procedure can be upper bounded by the extreme case where each element is exactly in the middle of a quantization interval and incurs an error of $\frac{\Delta}{2}$ where $\Delta = \frac{2w_{\max}}{(2^b-1)}$. The following theorem derives this bound.

Theorem 1 (Worst Case Error Bound for RTN)**.** *Let $W \in \mathbb{R}^{m\times n}$ be a weight matrix and $H \in \mathbb{R}^{n\times n}$ be a symmetric positive semi-definite (PSD) matrix. Let $\hat{W}$ be the matrix obtained by applying uniform b-bit Round-to-Nearest (RTN) quantization to each element of W. The quadratic quantization error is bounded as follows:*

$$\operatorname{tr}((\hat{W} - W)H(\hat{W} - W)^\top) \leq \frac{\mu_W^2}{(2^b-1)^2}\lambda_{\max}(H)||W||_F^2$$

where μ_W is the weight incoherence defined in Equation 1 and $\lambda_{\max}(H)$ is the maximum eigenvalue of the matrix H.

Proof. Let $\eta := \hat{W} - W$ be the quantization error matrix. The objective function is $\operatorname{tr}(\eta H \eta^\top)$. By the cyclic property of the trace, we can rearrange the terms:

$$\operatorname{tr}(\eta H \eta^\top) = \operatorname{tr}(H\eta^\top\eta)\,.$$

We apply the Von Neumann's trace inequality $\operatorname{tr}(AB) \leq \sum_{i=1}^n \lambda_i(A)\lambda_i(B)$, where $(\lambda_i(A))_{i=1}^n$, $(\lambda_i(B))_{i=1}^n$ are the eigenvalues of A and B, which holds if A, B are PSD. In our case $A = H$, $B = \eta^\top\eta$

$$\operatorname{tr}(H\eta^\top\eta) \leq \sum_{i=1}^n \lambda_i(H)\lambda_i(\eta^\top\eta) \leq \lambda_{\max}(H)\operatorname{tr}(\eta^\top\eta) = \lambda_{\max}(H)\|\eta\|_F^2\,.$$

Next, we bound the Frobenius norm of the error matrix η. For a uniform b-bit quantization scheme, the width of each quantization interval is $\Delta = \frac{2w_{\max}}{(2^b-1)}$. For Round-to-Nearest, the maximum error for any single element is half of this interval width:

$$|\eta_{ij}| \leq \frac{\Delta}{2} = \frac{w_{\max}}{2^b - 1}$$

Using this per-element bound, we can bound the squared Frobenius norm, which is the sum of the squared magnitudes of all elements:

$$\|\eta\|_F^2 = \sum_{i=1}^m\sum_{j=1}^n |\eta_{ij}|^2 \leq \sum_{i=1}^m\sum_{j=1}^n \left(\frac{w_{\max}}{2^b-1}\right)^2 = mn\left(\frac{w_{\max}}{2^b-1}\right)^2$$

Finally, substituting the bound for η yields

$$\operatorname{tr}(\eta H \eta^\top) \leq mn\left(\frac{w_{\max}}{2^b-1}\right)^2 \lambda_{\max}(H)\,.$$

We conclude the proof by using the definition of μ_W. □

B Experimental Setup

OptRot. We learn the rotations by optimizing the weight incoherence objective described in Equation 2 with Cayley SGD [12] (as in SpinQuant) with a learning rate of 1 for 1000 steps. For the cheaper top-50 version of OptRot, we learn the rotations by optimizing only over the 50 weight matrices with the largest loss.

SpinQuant/QuaRot. We use the implementation at https://github.com/facebookresearch/SpinQuant with default parameters. In the case of SpinQuant, to learn rotations, we use 800 samples of length 2048 from C4, and we only use activation quantization to 8-bit (as recommended when using GPTQ) if not otherwise specified. Activation quantization to 4-bits did not improve the result of SpinQuant. Rotations are optimized with 800 steps of Cayley SGD with learning rate 1.5 and batch size 1. To implement QuaRot, we use the same implementation (as SpinQuant) with initial rotations. Hence, we do not include the online Hadamard rotations before the out-projection, which would add inference cost and are less relevant for weight-only quantization.

Quantization. After rotating the model, we quantize it using GPTQ if not otherwise specified. For GPTQ, we use 512 samples of length 256 from the C4 dataset as the calibration set and we set the group-size parameter to 256 and the damping parameter to 0.01.

C Learning SpinQuant Rotations

By default, SpinQuant learns rotations by quantizing only the activations to 8-bits using RTN and a straight-through estimator to backpropagate the gradients to minimize the KL divergence. Since our focus is on weight quantization, while SpinQuant focuses more on activation quantizaton, we measure the effect of learning the SpinQuant rotations by quantizing only the weights **(SpinQuant W4)** or activations to 4-bits **(SpinQuant A4)** on Wikitext perplexity.

As we can see in Table 2, there are not significant improvements for both the A4 and W4 variants when quantizing weights with GPTQ.

Since SpinQuant uses RTN to learn the rotations, choosing weight or activation quantization has an effect on the downstream performance when quantizing weights with RTN (Table 3), i.e., when the quantization method is aligned for both rotation learning and quantization. We observe that when the rotations are learnt with weight quantization, weight-only quantization with SpinQuant performs significantly better than the baseline which learns rotations with activation quantization. This version of SpinQuant (W4) also outperforms OptRot (Table 8). See Appendix G for an in-depth discussion.

Table 2: WikiText perplexity for learning SpinQuant rotations with weight or activation quantization followed by weight-only quantization to 4-bits with GPTQ.

Method	Llama-3.2-1B	Llama-3.2-3B	Llama-3.1-8B
SpinQuant (Baseline)	10.73	8.37	6.71
SpinQuant (A4)	10.73	8.36	6.7
SpinQuant (W4)	10.79	8.35	6.69

Table 3: WikiText perplexity for learning SpinQuant rotations with weight or activation quantization followed by weight-only quantization to 4-bits with RTN.

Method	Llama-3.2-1B	Llama-3.2-3B	Llama-3.1-8B
SpinQuant (Baseline)	13.56	10.03	7.57
SpinQuant (W4)	11.75	8.97	7.14

D Complete Incoherence Plots

We report weight incoherence μ_W and Hessian incoherence μ_H plots for the 1B, 3B and 8B models across all layers for completeness in Figures 3 4 and 5. OptRot almost always finds the lowest weight incoherence, corresponding to the best RTN error bound in Equation 4. It also obtains the lowest product $\mu_H \mu_W$, which yields the best GPTQ error bound in Equation 5, in a majority of the layers for all three models.

E Weight Quantization

In addtion to Table 2 in the main paper, we report additional results for weight-only quantization.

Without online rotations. Table 4 reports the performance of OptRot in the absence of the online rotations R_3, R_4. Even without these online rotations, OptRot outperforms SpinQuant and QuaRot.

3-bit quantization. Table 7 reports performance of OptRot for 3-bit weight-only quantization. OptRot achieves a larger gain over SpinQuant in this regime, further reducing the gap between the 3-bit model and the full-precision baseline.

F Activation Quantization

We show that OptRot can provably improve the error for weight-only quantization. While OptRot does not directly optimize for activation quantization, we empirically validate its performance. We quantize the weights with GPTQ and the activations with Round-to-Nearest (RTN). We hypothesize that a lower Hessian incoherence benefits activation quantization. As shown in Figure 2, OptRot improves the weight incoherence at the cost of increasing the Hessian incoherence, yet still improving the overall weight quantization bound. Hence, we would expect OptRot to not perform as well as SpinQuant for activation quantization.

A8W4. In this regime, OptRot is still competetive with SpinQuant with lower perplexity across all three models, although the gain is smaller. Quantizing the activations to 8-bits still achieves an acceptable overall performance for serving models [21]. Results are reported in Table 5.

A4W4. Quantizing activations to 4-bits is a hard problem [21] and the overall model performance drops significantly in this regime, making the model unusable in practice. Here, we observe that OptRot performs worse than SpinQuant. This is likely due to the higher Hessian incoherence achieved by OptRot. Results are reported in Table 6.

G RTN Results

Learning rotations with OptRot improves the error bound with Round-to-Nearest shown in Equation 4. In case of RTN, Equation (4) shows that the error is bounded only by the weight incoherence. Figure 7 shows that OptRot achieves the lowest incoherence. This translates to lower error (higher SNR) as shown in Figure 8. Table 8 reports results for weight-only quantization with RTN, where OptRot outperforms QuaRot (Hadamard rotations).

SpinQuant W4. Table 8 also shows that SpinQuant (W4) performs significantly better than both SpinQuant and OptRot on downstream tasks. However, the weight incoherence (Figure 7) and more importantly the SNR (Figure 8) plots show that OptRot usually achieves a lower error bound (Equation 4) and layerwise objective (Equation 3) than SpinQuant. This difference can be explained by the fact that SpinQuant (W4) optimizes an end-to-end quantization-aware objective: the KL divergence between the original and quantized model. Thus, SpinQuant, differently from OptRot, can exploit interactions between weights: the rotations can be learnt in a manner where certain weights compensate for quantization errors in other weights, both within and between layers.

This effect benefits downstream quantization when the quantization method used for rotation learning and quantization are the same (RTN in this case). In contrast, while using RTN for rotation learning and quantizing with GPTQ, these improvements are not observed (see Table 2). These results suggest that for SpinQuant, aligning the quantization method for both rotation learning and quantization yields the best downstream results. However, performing this alignment for GPTQ is not straightforward: for each step of rotation learning we would have to quantize each row of each weight matrix sequentially and update the Hessian matrix, making rotation learning more costly. Our approach, provides a way to more efficiently improve the performance of GPTQ by minimizing an upper bound to the layerwise objective. We note that GPTQ consistently outperforms RTN, and OptRot + GPTQ achieves the best overall results for weight-only quantization.

H SNR

We also report the Signal-to-Noise Ratio (SNR) in dB to measure the quality of the quantization with GPTQ. The SNR is computed for GPTQ as:

$$\text{SNR} := 10 * \log_{10}\left(\frac{\mathbb{E}\|Wx\|^2}{\mathbb{E}\|(\hat{W}-W)x\|^2}\right) = 10 * \log_{10}\left(\frac{\text{tr}(WHW^\top)}{\text{tr}((W-\hat{W})H(W-\hat{W})^\top)}\right), \quad (6)$$

where $H = \mathbb{E}[xx^\top]$ and the Expectation is computed over the calibration set. Note that any upper bound on the incoherence can be used to lower bound the SNR.

Figure 6 shows the SNR for each method across models. A high SNR means small quantization error. According to the bound in Equation 5 lowering the weight incoherence should improve the error and increase the SNR. Our results confirm this observation, the lower weight incoherence achieved by OptRot translates to a higher SNR for weight quantization.

Table 4: Results without online (R_3, R_4) rotations for weight-only quantization at 4-bits with GPTQ.

Method	Llama-3.2-1B			Llama-3.2-3B			Llama-3.1-8B		
	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓
FP16	56.77	9.76	0	64.72	7.81	0	70.29	6.24	0
No Rotation	49.43	13.86	0.362	56.28	11.78	0.355	67.39	7.47	0.233
Quarot	**54.6**	11.27	0.208	62.52	8.64	0.183	68.7	6.81	0.154
SpinQuant	54.19	11.21	0.211	62.3	8.69	0.181	68.66	6.79	0.152
OptRot (top-50)	54.4	11.06	0.189	**63.3**	8.51	**0.153**	**69.28**	6.74	**0.133**
OptRot (all)	54.42	**10.98**	**0.185**	62.29	**8.48**	0.163	69.1	**6.71**	**0.133**

Table 5: Results for A8W4 quantization, where activations are quantized with RTN and weights with GPTQ.

Method	Llama-3.2-1B			Llama-3.2-3B			Llama-3.1-8B		
	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓
FP16	56.77	9.76	0	64.72	7.81	0	70.29	6.24	0
No Rotation	48.9	14.49	0.367	54.93	13.03	0.367	66.38	7.65	0.249
Quarot	55.2	10.8	0.137	63.23	8.37	0.099	69.17	6.72	0.094
SpinQuant	55.21	10.75	0.133	**63.95**	8.34	0.098	**69.5**	6.72	0.096
OptRot (top-50)	55.1	10.68	**0.124**	63.28	8.32	**0.092**	69.4	6.66	**0.082**
OptRot (all)	**55.55**	**10.63**	0.126	63.33	**8.29**	0.094	69.32	**6.65**	0.087

Table 6: Results for A4W4 quantization, where activations are quantized with RTN and weights with GPTQ.

Method	Llama-3.2-1B			Llama-3.2-3B			Llama-3.1-8B		
	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓
FP16	56.77	9.76	0	64.72	7.81	0	70.29	6.24	0
No Rotation	32.22	204	2.88	34.67	367	2.01	37.7	75.68	2.07
Quarot	49.85	13.67	**0.392**	57.64	9.96	**0.297**	65.05	7.81	0.275
SpinQuant	49.63	**13.56**	0.393	**58.69**	**9.88**	0.308	**65.7**	**7.8**	**0.273**
OptRot (top-50)	**50.47**	14.05	0.429	56.92	10.97	0.417	64.05	8.29	0.325
OptRot (all)	50.11	13.97	0.43	57.67	10.34	0.362	65.27	8.38	0.348

Table 7: Results for 3-bit weight-only quantization with GPTQ.

Method	Llama-3.2-1B			Llama-3.2-3B			Llama-3.1-8B		
	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓
FP16	56.77	9.76	0	64.72	7.81	0	70.29	6.24	0
No Rotation	40.23	77.32	1.62	50.55	15.58	0.75	58.13	11.82	0.64
Quarot	51.25	14.47	0.427	**60.68**	10.24	0.322	66.36	8	0.275
SpinQuant	50.8	14.38	0.415	59.83	10.28	0.327	66.81	7.99	0.275
OptRot (top-50)	**51.77**	14	0.407	**60.68**	**10.1**	0.32	66.8	7.86	0.263
OptRot (all)	51.37	**13.78**	**0.384**	60.36	10.17	**0.313**	**67.31**	**7.85**	**0.257**

Table 8: Results for 4-bit weight-only quantization with RTN.

Method	Llama-3.2-1B			Llama-3.2-3B			Llama-3.1-8B		
	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓	Acc ↑	Wiki ↓	KL ↓
FP16	56.77	9.76	0	64.72	7.81	0	70.29	6.24	0
No Rotation	48.14	15.13	0.451	54.84	16.2	0.416	64.38	8.23	0.279
Quarot	51.27	13.7	0.4	59.41	10.1	0.327	67.2	7.57	0.244
SpinQuant	51.18	13.56	0.399	59.54	10.03	0.324	67.14	7.57	0.242
OptRot (all)	52.26	12.81	0.331	60.46	9.64	0.273	67.65	7.4	0.224
SpinQuant (W4)	**54.65**	**11.75**	**0.241**	**61.75**	**8.97**	**0.198**	**67.79**	**7.14**	**0.171**

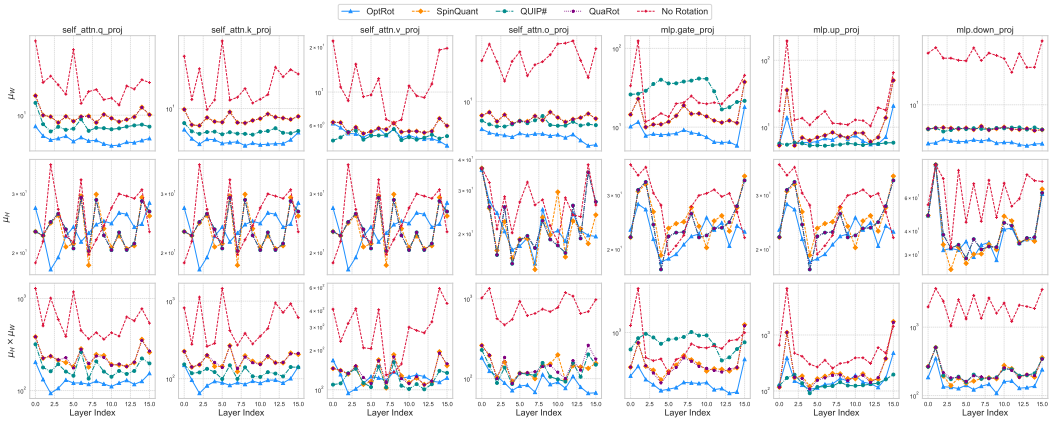

Figure 3: Weight Incoherence μ_W (top row), Hessian Incoherence μ_H (middle row) and $\mu_H \mu_W$ (bottom row) on Llama-3.2-1B.

Figure 4: Weight Incoherence μ_W (top row), Hessian Incoherence μ_H (middle row) and $\mu_H \mu_W$ (bottom row) on Llama-3.2-3B.

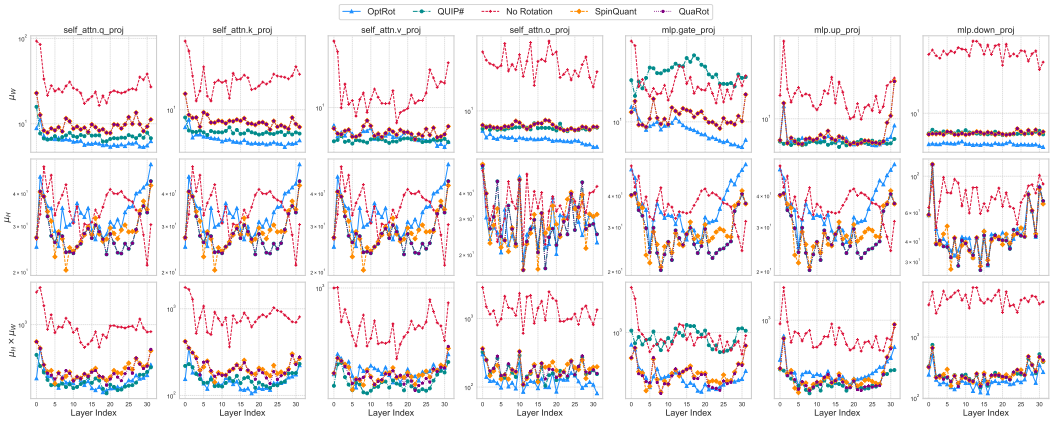

Figure 5: Weight Incoherence μ_W (top row), Hessian Incoherence μ_H (middle row) and $\mu_H \mu_W$ (bottom row) on Llama-3.1-8B.

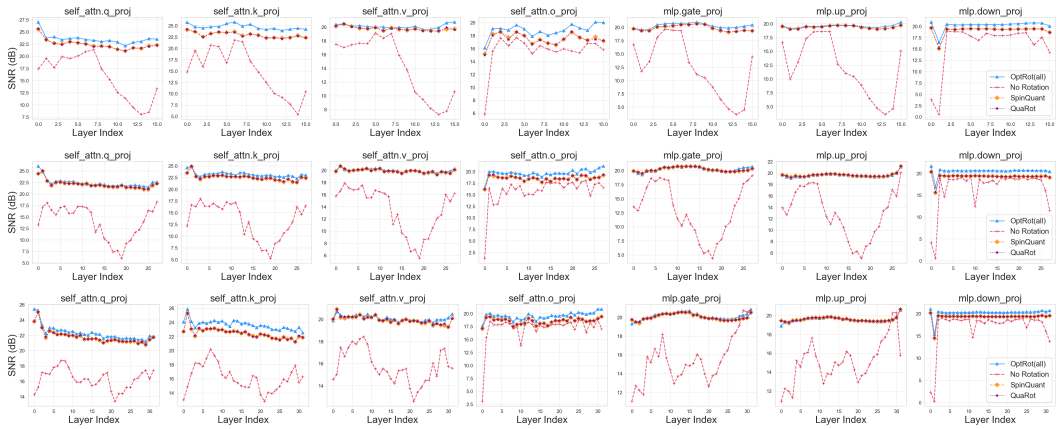

Figure 6: SNR for weight-only quantization with GPTQ at 4-bits for Llama-3.2-1B (top row), Llama-3.2-3B (middle row) and Llama-3.1-8B (bottom row).

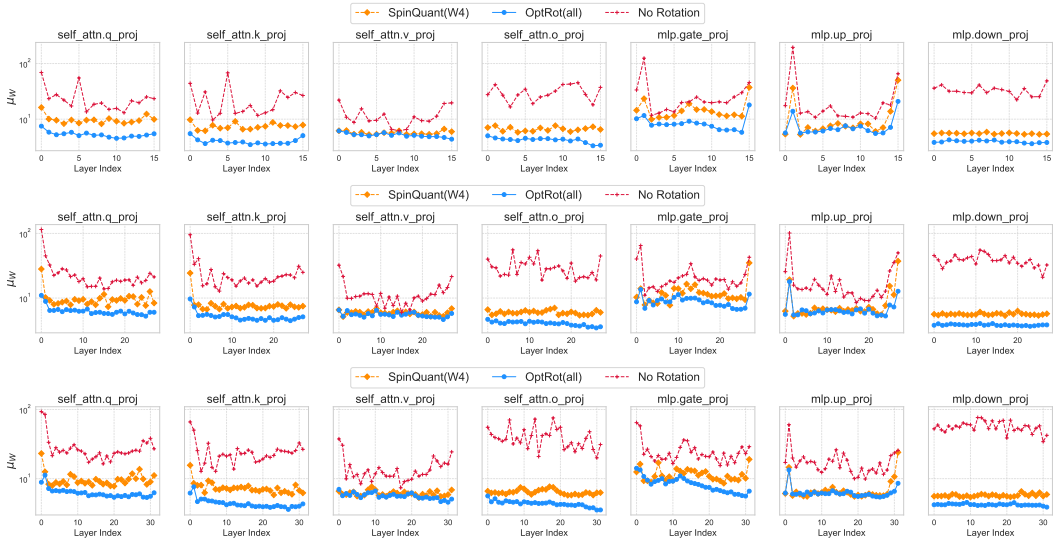

Figure 7: Weight incoherence μ_W comparison for SpinQuant (W4) and OptRot on Llama-3.2-1B (top row), Llama-3.2-3B (middle row) and Llama-3.1-8B (bottom row).

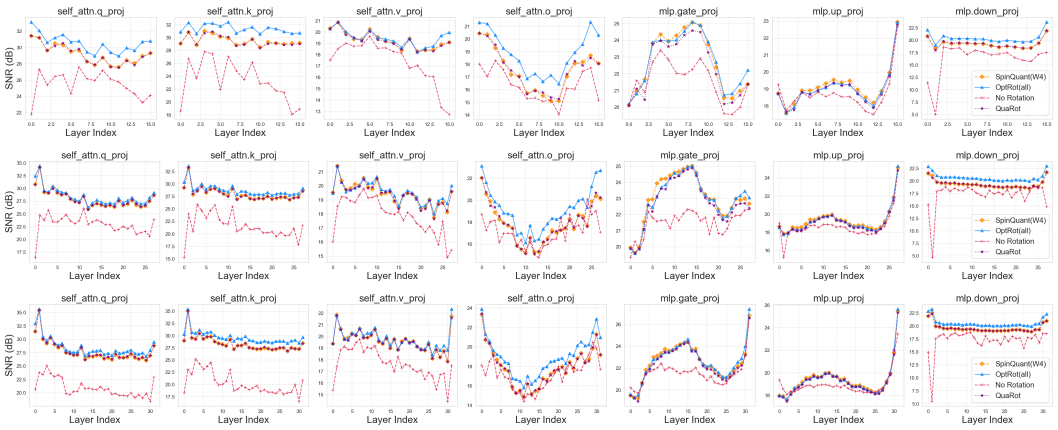

Figure 8: SNR for weight-only quantization with RTN at 4-bits for Llama-3.2-1B (top row), Llama-3.2-3B (middle row) and Llama-3.1-8B (bottom row).