Journal of Audit and Accountability
Volume 1 · Number 1 · 2026

Choosing More Bits for Some Layers Does Not Guarantee a Better Equal-Budget Model

Michael Sanders[1] · Daniel Gardner[2] · Jason Reed[3]

[1] United Kingdom · [2] Germany · [3] France

Original publication: November 7, 2026 | Article 595

Abstract

We assigned five bits to three model layers and three bits to the other three, matching the selected-weight code budget of uniform four-bit rounding. Twenty mixed allocations were compared on a calibration text. The chosen allocation improved held-text loss over a fixed alternating pattern by 0.5494 nats per token, but remained worse than uniform four-bit rounding by 0.9400. Calibration helped within a restricted candidate family without making that family preferable to a simpler baseline. The experiment used actual forward passes of a small public model and establishes a numerical loss comparison, not a hardware speed or packed-model-size result.

Actual CPU next-token scoring of reconstructed Pythia-14M weights. The controlled budget covers selected weight codes and group scales, while execution remains dense floating point.

Choosing More Bits for Some Layers Does Not Guarantee a Better Equal-Budget Model

> Actual CPU next-token scoring of reconstructed Pythia-14M weights. The controlled budget covers selected weight codes and group scales, while execution remains dense floating point.

Abstract

We assigned five bits to three model layers and three bits to the other three, matching the selected-weight code budget of uniform four-bit rounding. Twenty mixed allocations were compared on a calibration text. The chosen allocation improved held-text loss over a fixed alternating pattern by 0.5494 nats per token, but remained worse than uniform four-bit rounding by 0.9400. Calibration helped within a restricted candidate family without making that family preferable to a simpler baseline. The experiment used actual forward passes of a small public model and establishes a numerical loss comparison, not a hardware speed or packed-model-size result.

The budget comparison behind a mixed-precision claim

Assigning different numerical precision to different layers is attractive because layers can differ in their sensitivity to weight reconstruction. Yet moving bits toward one part of a model takes them away from another when the total code budget is fixed. A mixed allocation should therefore be compared with a uniform allocation at a declared common budget, rather than only with a weak or arbitrary mixed pattern.

We ask whether selecting the high-bit layers using calibration-text loss improves prediction on another text. The experiment compares a uniform four-bit rule, a fixed mixed rule and a calibration-selected mixed rule. Each mixed rule gives five bits to exactly three of six layers and three bits to the other three. Equal parameter counts per layer make their selected-weight code totals directly comparable.

Wang et al., 2026 address dynamic fusion for language-model compression. We do not implement its fusion mechanism. The connection is the need to separate a useful allocation decision from a stronger compression-performance claim. Our study evaluates a transparent allocation family using actual next-token losses, without treating the result as evidence for or against a different compression algorithm.

Mixed-precision approaches such as HAWQ explicitly study layer-sensitive allocation (Dong et al., 2019). HMQ addresses mixed quantization under hardware-oriented constraints in another modeling setting (Habi et al., 2020). Neither procedure is reproduced here. We use a much smaller exhaustive search whose candidates and evaluation criterion can be inspected completely, accepting that such enumeration does not provide a scalable method for large networks.

One public checkpoint and two local text roles

The model is the final public Pythia-14M standard-Pile checkpoint (Biderman et al., 2023). We load its captured tensor file into the corresponding GPT-NeoX architecture, promote source values to float32 and use evaluation mode with eager attention. The complete original state is restored before every allocation is applied. We do not fine-tune the checkpoint, update embeddings or introduce a learned calibration module.

The first 2,048 next-token targets of Tiny Shakespeare supply the selection criterion (Karpathy, 2026). Evaluation uses the first 16,384 next-token targets of the captured WikiText-2 test text (Merity et al., 2016). WikiText rows are joined with newline separators, yielding 287,151 tokens under the captured model tokenizer. The tokenizer roundtrip reproduces that joined text exactly.

The two texts have distinct roles and local file boundaries. Evaluation losses are not used to choose the three high-bit layers. However, neither text is certified absent from the checkpoint's original pretraining corpus. The study concerns locally separated calibration and evaluation information, not a guarantee that the public language model encounters entirely unseen material for the first time.

Both texts are scored with 256-input contexts reset between blocks. Each block predicts the following 256 observed tokens, so the calibration criterion averages eight blocks and evaluation averages sixty-four. Boundary inputs can be shared across neighboring blocks, but target positions are counted once. There is no generated continuation or feedback from a sampled output into a later context.

Every forward pass retains all 50,304 output logits in the softmax. The outcome is mean observed-target negative log likelihood in nats. It is not a code-generation pass rate, a factuality score or an estimate of conversational utility. Proper scoring considerations explain why the reported predictive quantity should remain distinct from a task-specific success event (Gneiting and Raftery, 2007).

What receives bits and what the budget includes

We reconstruct twenty-four internal attention and feed-forward weight matrices, four in each of six layers. Each layer contributes 196,608 affected weights, for 1,179,648 in total. Input and output embeddings, biases and normalization parameters retain their original promoted values. Unaffected state entries are checked for equality after each allocation is applied.

Within a selected matrix, values are grouped in row-major blocks of 128. Each block's scale is its largest absolute source value divided by the positive integer limit for the selected bit width. Values are rounded to the nearest code, bounded symmetrically and multiplied by that scale. No per-group clipping factor, activation quantization or error-compensation update is fitted.

The positive limits are three, seven and fifteen for three-, four- and five-bit rules. These symmetric ranges leave one representable signed code unused. Bit labels describe the space needed for the specified integer codes; reconstructed tensors still execute in float32. Finite-precision analysis motivates stating the code convention and rounding calculation explicitly (Higham, 2002; Goldberg, 1991).

Uniform four-bit coding uses 4,718,592 theoretical weight-code bits. Three five-bit layers and three three-bit layers use the same total because the layer weight counts are equal. Every policy also has 9,216 groups and one float32 scale per group, totaling 36,864 scale bytes. Thus the controlled quantities are code payload and scale inventory, not merely the arithmetic average of unrelated matrix bit labels.

The comparison does not include a serialized layer-allocation descriptor or format headers. Such information would be needed to decode a real mixed-bit file, and an actual implementation could add alignment or packing overhead. We therefore make no claim of byte-identical compressed files. The equal-budget statement refers specifically to the selected codes and scales, with the remaining source parameters retained.

The group-scale inventory adds one quarter of a theoretical bit per affected weight, since a thirty-two-bit scale is shared by 128 weights. Both uniform and mixed policies thus have 4.25 code-plus-scale bits per affected weight under this accounting. This is an inventory calculation before format overhead, not the memory footprint of the floating-point tensors used to execute the experiment.

Selecting among twenty mixed allocations

There are twenty ways to choose three high-bit layers from six. We evaluate all of them on the same calibration prefix. Each candidate starts from the original weights and applies its own three-/five-bit reconstruction. The selection criterion is actual mean next-token loss, using observed text targets without gradient-based training. Ties are resolved by the lexicographic order of the layer-index triples.

The selected high-bit layers are indices 1, 2 and 3. The fixed comparison assigns five bits to alternating indices 0, 2 and 4. That fixed pattern belongs to the searched candidate family, so the chosen allocation cannot have greater calibration loss than it under the implemented criterion. This inequality follows from selection; the held-text comparison remains an empirical result.

Calibration loss is 6.161669 for the selected pattern and 7.012789 for the fixed alternating pattern. Uniform four-bit coding is not a member of the twenty mixed candidates and is not used in their selection. Consequently, finding a calibration winner does not establish that the candidate family includes the best available policy at this code budget. The evaluation retains uniform coding as an external fixed baseline for that question.

This distinction separates parameter search from a guarantee about compression quality. Model-selection research explains why assessment data must be held outside the choice criterion (Varma and Simon, 2006), and leakage analysis makes the information boundary explicit (Kapoor and Narayanan, 2023). We protect that local boundary but do not infer that the selected layer triple will remain preferable on every corpus or checkpoint.

The search scores the same 2,048 calibration targets twenty times. Those are repeated evaluations of one text prefix, not 40,960 independent calibration observations. Enumerating the candidates is feasible for this small six-layer model. It would grow combinatorially for a larger fixed fraction of high-bit layers, so no large-model efficiency claim follows from its feasibility here.

The selected triple is global. Every held block uses the same high-bit layers, with no input-dependent switch or later adjustment. This matters when interpreting the allocation as a calibration product: it is one frozen model state rather than a controller that observes a query and chooses its precision. A dynamic policy would require a separate decision rule, an account of its information and computation, and an evaluation of the resulting conditional choices.

Selection also does not identify an intrinsic sensitivity ranking of individual layers. The chosen indices minimize loss jointly among triples while the complementary layers receive lower precision. Changing one member changes the allocation's interaction with every subsequent computation. We therefore report the winning triple as a property of this candidate family and calibration text, rather than declare its three members universally the most important layers.

Actual held-text losses

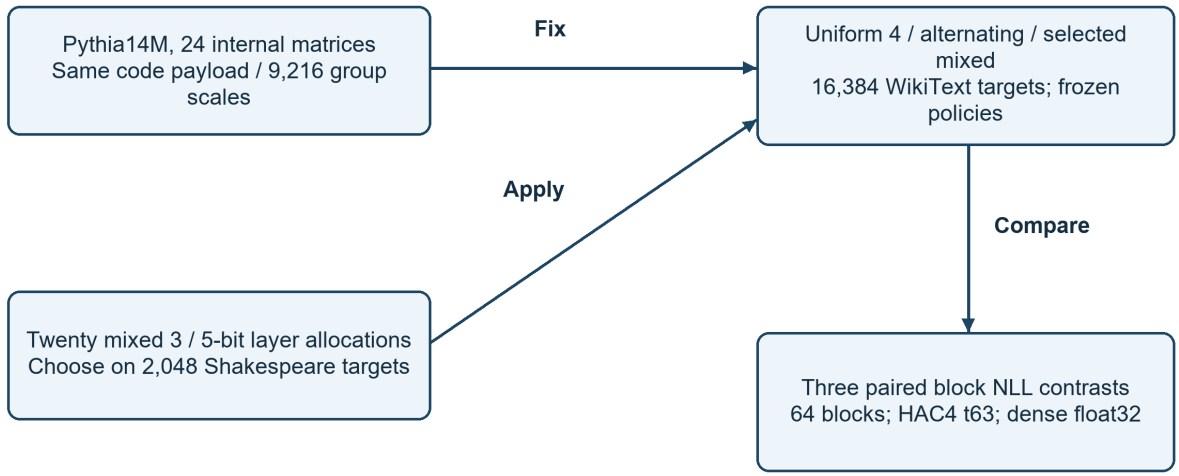

Figure 1. *Twenty equal-code-budget mixed allocations are scored on the calibration prefix. A frozen selected pattern, a fixed alternating pattern and uniform four-bit coding are compared on separate WikiText target positions using actual forward losses.*

The unmodified checkpoint's evaluation loss is 5.291043 nats per token. Uniform four-bit coding gives 6.189282, the fixed mixed pattern 7.678659 and the selected mixed pattern 7.129243. All three reconstructed policies lose predictive quality relative to the source on this segment. The selected mixed policy improves its family comparison while retaining a substantial disadvantage against uniform coding.

We form three block-paired contrasts: selected mixed minus uniform, fixed mixed minus uniform, and selected mixed minus fixed mixed. Each regression has an intercept, sixty-four block rows and no other covariates. Standard errors use Bartlett HAC covariance with four block lags and a finite correction, and intervals use t63. The HAC framework addresses a declared local serial-dependence structure (Newey and West, 1987).

Table 1. *Held next-token NLL differences in nats, N=64 paired block means, HAC4 covariance and t63 intervals. Intercept-only R-squared is numerically zero. The covariance conditions on the fitted allocation and does not propagate calibration-prefix, checkpoint-training or independent-corpus uncertainty.*

Model	Term	Estimate (SE)	95% CI	N
M1	Mean outcome/contrast	0.939962 (0.048378)	[0.843286, 1.036637]	64
M1: fit	R-squared = 0.0000	-	-	64
M2	Mean outcome/contrast	1.489377 (0.044759)	[1.399933, 1.578820]	64
M2: fit	R-squared = 0.0000	-	-	64
M3	Mean outcome/contrast	-0.549415 (0.047859)	[-0.645055, -0.453776]	64
M3: fit	R-squared = 0.0000	-	-	64

Model labels: M1 = Calibration-selected mixed minus uniform four-bit NLL; M2 = Fixed alternating mixed minus uniform four-bit NLL; M3 = Calibration-selected minus fixed alternating mixed NLL.

Selected mixed coding exceeds uniform loss by 0.939962, standard error 0.048378 and interval [0.843286, 1.036637]. Fixed alternating mixed coding exceeds uniform by 1.489377, standard error

0.044759 and interval [1.399933, 1.578820]. Selected minus fixed mixed loss is -0.549415, with standard error 0.047859 and interval [-0.645055, -0.453776].

These contrasts preserve the shared target positions. Subtracting predictions evaluated on unrelated text blocks would introduce a composition change into the allocation comparison. The paired estimates instead isolate numerical policy differences on the same observed contexts, while leaving unmodeled broader corpus variation visible. Validation principles require that any claim of transport beyond this text be tested on the intended population (Roberts et al., 2017).

Loss differences are in nats per token, not percentages of accuracy retained. Converting a mean loss to perplexity applies an exponential transformation and changes the interpretation of a numerical gap. Measurement-scale distinctions therefore matter when presenting compression results. We retain the directly scored NLL unit rather than describing the result as an unsupported percentage decrease in model capability.

Why the negative mixed-versus-uniform result matters

The allocation family forces half the layers down to three bits. Giving five bits to the others does not mathematically compensate for that change in predictive loss. Effects can interact through subsequent layers, and the actual forward criterion is not a sum of independently measured weight distortions. This is why equal code totals do not imply equal output quality or a conserved amount of useful model information.

The search improves which layers receive the higher precision within that family. It does not relax the family restriction or add uniform four-bit coding to the selection set. Including a broader baseline family in a future calibration search could be sensible, but it would define another prespecified experiment. We do not revise the current selection using the evaluation outcome and then present the revision as the original result.

Distribution-shift studies motivate caution when a selected rule moves between text sources (Ovadia et al., 2019). Our observed result does not identify whether the remaining disadvantage comes mainly from the three-bit constraint, the calibration content, or their interaction. Distinguishing those explanations would require additional controlled calibration cohorts and allocation families. The current comparison establishes their combined effect under the declared design.

Existing quantization methods tackle more sophisticated problems. GPTQ, AWQ, SmoothQuant and OmniQuant use different approximation or calibration strategies (Frantar et al., 2022; Lin et al., 2023; Xiao et al., 2022; Shao et al., 2023). LLM.int8 and ZeroQuant supply further numerical-compression contexts (Dettmers et al., 2022; Yao et al., 2022). None is executed, and their published claims cannot be evaluated from our simple group-maximum rounding experiment.

Compression also includes changes other than weight-code precision. Wanda and LLM-Pruner examine pruning-related routes (Sun et al., 2023; Ma et al., 2023), while LoRA concerns low-rank adaptation (Hu et al., 2022). Their budgets and objectives differ from the fixed reconstruction inventory here. We do not equate fewer nonzero weights, fewer adaptation parameters and fewer weight-code bits as one interchangeable compression quantity.

Numerical quality and hardware performance need separate evidence

The PyTorch runtime performs actual dense floating-point forward passes for the reconstructed tensors (Paszke et al., 2019). It does not execute packed three-, four- or five-bit kernels. A hardware roofline perspective explains why arithmetic, memory movement and implementation limits all affect performance (Williams et al., 2009). Matching a theoretical code payload alone does not supply those measurements.

Serving and execution methods introduce additional considerations. PagedAttention concerns memory management in language-model serving (Kwon et al., 2023), FlashAttention addresses attention execution (Dao et al., 2022), and speculative decoding changes the decoding procedure (Leviathan et al., 2022). Those systems are not benchmarked here. Their references clarify the separate evidence needed before converting a numerical loss comparison into a latency, throughput or memory-efficiency claim.

The main empirical finding is therefore conditional and useful: loss-based selection substantially improves a fixed mixed pattern on this held text, yet uniform four-bit reconstruction performs better at the stated code-and-scale budget. Retaining both comparisons prevents a favorable within-family result from becoming a claim that mixed precision must dominate uniform precision.

The local artifacts include all twenty calibration losses, the selected layer triple, code and scale counts, and per-token losses for every held policy. Dataset documentation preserves their scope (Gebru et al., 2021), and reusable provenance supports tracing the result to the captured checkpoint and texts (Wilkinson et al., 2016). Wider model, corpus and hardware studies remain necessary for a deployment recommendation, while this calculation supplies a transparent counterexample to an unconditional allocation advantage.

References

1. Biderman, Stella; Schoelkopf, Hailey; Anthony, Quentin; Bradley, Herbie; O'Brien, Kyle; Hallahan, Eric; Khan, Mohammad Aflah; Purohit, Shivanshu; Prashanth, USVSN Sai; Raff, Edward; Skowron, Aviya; Sutawika, Lintang; van der Wal, Oskar. (2023). Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. https://arxiv.org/abs/2304.01373.
2. Ma, Xinyin; Fang, Gongfan; Wang, Xinchao. (2023). LLM-Pruner: On the Structural Pruning of Large Language Models. https://arxiv.org/abs/2305.11627.
3. Frantar, Elias; Ashkboos, Saleh; Hoefler, Torsten; Alistarh, Dan. (2022). GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. https://arxiv.org/abs/2210.17323.
4. Wang, Junhao; Xu, Guoxi; Tong, Huimin; Xu, Gaochao; Chen, Yiwei. (2026). Dynamic Fusion for Large Language Models Compression. ICASSP 2026, 4711-4715. https://doi.org/10.1109/icassp55912.2026.11464073.
5. Ovadia, Yaniv, et al. “Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift.” *Advances in Neural Information Processing Systems*, vol. 32, 2019. https://arxiv.org/abs/1906.02530.
6. Williams, Samuel; Waterman, Andrew; Patterson, David. (2009). Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM. https://doi.org/10.1145/1498765.1498785.
7. Kwon, Woosuk; Li, Zhuohan; Zhuang, Siyuan; Sheng, Ying; Zheng, Lianmin; Yu, Cody Hao; Gonzalez, Joseph E.; Zhang, Hao; Stoica, Ion. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. https://arxiv.org/abs/2309.06180.
8. Dao, Tri; Fu, Daniel Y.; Ermon, Stefano; Rudra, Atri; Ré, Christopher. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. https://arxiv.org/abs/2205.14135.
9. Varma, Sudhir and Richard Simon. “Bias in error estimation when using cross-validation for model selection.” *BMC Bioinformatics*, vol. 7, no. 1, 2006. https://doi.org/10.1186/1471-2105-7-91.
10. Goldberg, David. (1991). What every computer scientist should know about floating-point arithmetic. ACM Computing Surveys. https://doi.org/10.1145/103162.103163.
11. Kapoor, Sayash; Narayanan, Arvind. (2023). Leakage and the reproducibility crisis in machine-learning-based science. Patterns. https://doi.org/10.1016/j.patter.2023.100804.

12. Karpathy, Andrej. (2026). Tiny Shakespeare text, char-rnn repository [Data file; snapshot retrieved 7 November 2026]. https://github.com/karpathy/char-rnn/blob/master/data/tinyshakespeare/input.txt.

13. Gneiting, Tilmann; Raftery, Adrian E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. Journal of the American Statistical Association. https://doi.org/10.1198/016214506000001437.

14. Sun, Mingjie; Liu, Zhuang; Bair, Anna; Kolter, J. Zico. (2023). A Simple and Effective Pruning Approach for Large Language Models. https://arxiv.org/abs/2306.11695.

15. Roberts, David R., et al. “Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure.” *Ecography*, vol. 40, no. 8, 2017, pp. 913-929. https://doi.org/10.1111/ecog.02881.

16. Wilkinson, Mark D.; Dumontier, Michel; Aalbersberg, IJsbrand Jan; Appleton, Gabrielle; Axton, Myles; Baak, Arie; Blomberg, Niklas; Boiten, Jan-Willem; da Silva Santos, Luiz Bonino; Bourne, Philip E.; Bouwman, Jildau; Brookes, Anthony J.; Clark, Tim; Crosas, Mercè; Dillo, Ingrid; Dumon, Olivier; Edmunds, Scott; Evelo, Chris T.; Finkers, Richard; Gonzalez-Beltran, Alejandra; Gray, Alasdair J.G.; Groth, Paul; Goble, Carole; Grethe, Jeffrey S.; Heringa, Jaap; ’t Hoen, Peter A.C; Hooft, Rob; Kuhn, Tobias; Kok, Ruben; Kok, Joost; Lusher, Scott J.; Martone, Maryann E.; Mons, Albert; Packer, Abel L.; Persson, Bengt; Rocca-Serra, Philippe; Roos, Marco; van Schaik, Rene; Sansone, Susanna-Assunta; Schultes, Erik; Sengstag, Thierry; Slater, Ted; Strawn, George; Swertz, Morris A.; Thompson, Mark; van der Lei, Johan; van Mulligen, Erik; Velterop, Jan; Waagmeester, Andra; Wittenburg, Peter; Wolstencroft, Katherine; Zhao, Jun; Mons, Barend. (2016). The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. https://doi.org/10.1038/sdata.2016.18.

17. Leviathan, Yaniv; Kalman, Matan; Matias, Yossi. (2022). Fast Inference from Transformers via Speculative Decoding. https://arxiv.org/abs/2211.17192.

18. Newey, Whitney K., and Kenneth D. West. “A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix.” *Econometrica*, vol. 55, no. 3, 1987, pp. 703-708. https://doi.org/10.2307/1913610.

19. Hu, Edward J., et al. “LoRA: Low-Rank Adaptation of Large Language Models.” *International Conference on Learning Representations*, 2022. https://openreview.net/forum?id=nZeVKeeFYf9.

20. Dong, Zhen; Yao, Zhewei; Gholami, Amir; Mahoney, Michael W.; Keutzer, Kurt. (2019). HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision. arXiv:1905.03696. https://arxiv.org/abs/1905.03696.

21. Lin, Ji; Tang, Jiaming; Tang, Haotian; Yang, Shang; Chen, Wei-Ming; Wang, Wei-Chen; Xiao, Guangxuan; Dang, Xingyu; Gan, Chuang; Han, Song. (2023). AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. https://arxiv.org/abs/2306.00978.

22. Yao, Zhewei; Aminabadi, Reza Yazdani; Zhang, Minjia; Wu, Xiaoxia; Li, Conglong; He, Yuxiong. (2022). ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers. https://arxiv.org/abs/2206.01861.

23. Paszke, Adam; Gross, Sam; Massa, Francisco; Lerer, Adam; Bradbury, James; Chanan, Gregory; Killeen, Trevor; Lin, Zeming; Gimelshein, Natalia; Antiga, Luca; Desmaison, Alban; Köpf, Andreas; Yang, Edward; DeVito, Zach; Raison, Martin; Tejani, Alykhan; Chilamkurthy, Sasank; Steiner, Benoit; Fang, Lu; Bai, Junjie; Chintala, Soumith. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. https://arxiv.org/abs/1912.01703.

24. Higham, Nicholas J.. (2002). Accuracy and Stability of Numerical Algorithms: Second Edition. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718027.

25. Xiao, Guangxuan; Lin, Ji; Seznec, Mickael; Wu, Hao; Demouth, Julien; Han, Song. (2022). SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. https://arxiv.org/abs/2211.10438.

26. Habi, Hai Victor; Jennings, Roy H.; Netzer, Arnon. (2020). HMQ: Hardware Friendly Mixed Precision Quantization Block for CNNs. arXiv:2007.09952. https://arxiv.org/abs/2007.09952.

27. Shao, Wenqi; Chen, Mengzhao; Zhang, Zhaoyang; Xu, Peng; Zhao, Lirui; Li, Zhiqian; Zhang, Kaipeng; Gao, Peng; Qiao, Yu; Luo, Ping. (2023). OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models. https://arxiv.org/abs/2308.13137.

28. Merity, Stephen; Xiong, Caiming; Bradbury, James; Socher, Richard. (2016). Pointer Sentinel Mixture Models. https://arxiv.org/abs/1609.07843.

29. Dettmers, Tim; Lewis, Mike; Belkada, Younes; Zettlemoyer, Luke. (2022). LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. https://arxiv.org/abs/2208.07339.

30. Gebru, Timnit, et al. "Datasheets for datasets" *Communications of the ACM*, 2021. https://doi.org/10.1145/3458723.