Optimal Formats for Weight Quantisation

Douglas Orr [1] **Luka Ribar** [1] **Carlo Luschi** [1]

Abstract

We propose a framework for systematic design and analysis of quantisation formats. Our objective of minimising the KL divergence between the original and quantised model outputs aligns with minimising the squared quantisation error of the model parameters. Guided by classical quantisation theory, we therefore develop and evaluate squared-error-optimal formats for known distributions. Uniform quantisation followed by lossless compression with a variable-length code is shown to be optimal. However, we find that commonly used block formats and sparse outlier formats also outperform fixed-length codes, implying they also exploit variable-length encoding. Finally, we derive the optimal allocation of bit-widths to individual parameter tensors across the model's layers, saving up to 0.25 bits per parameter when tested with direct-cast quantisation of language models.

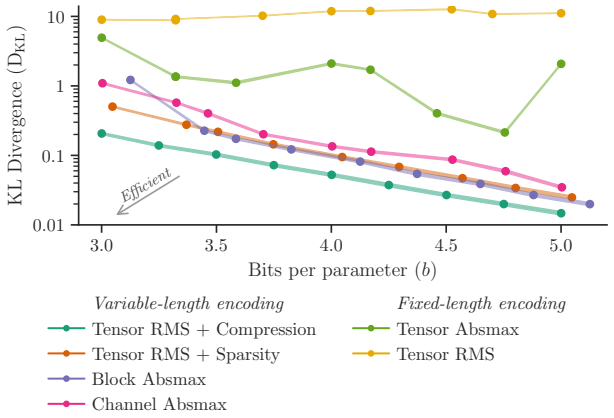

Figure 1. The trade-off between average bits per parameter and top-k KL divergence for Llama 3.1 8B. To approach optimal performance, some form of variable-length encoding is needed: lossless compression, block (or channel) absmax scaling or 0.1% sparse outlier removal. The shaded line width is ± 2 standard error over evaluation data. See Figure 6 for other models.

1. Introduction

Weight quantisation enables large deep learning models to run on low-resource hardware and edge devices by saving space and memory bandwidth usage. It can be seen as an optimisation problem, where the goal is to retain the behaviour of the high-precision reference model while reducing the total number of bits needed to store its parameters. This naturally splits into two sub-problems of format design and quantisation procedure, both of which are highly active areas of research. We focus on the format design question, i.e., how to choose a representation space for model parameters. This is somewhat independent from the quantisation procedure, which aims to find an optimal point in that space.

The space of possible formats for a model is rich, where element formats including integer, floating-point and non-uniform quantisers can be combined with tensor, channel or block scaling and augmented using sparse outlier storage or rotations (Dettmers et al., 2022a; Dettmers & Zettlemoyer, 2023; Dettmers et al., 2023; Dotzel et al., 2024; Rouhani et al., 2023; Tseng et al., 2024). Since this combinatorial space is too large to be explored directly, empirical studies typically narrow the search space.

This work considers the problem of optimal format design after direct-cast quantisation. Our main takeaway is that *the most effective quantisation formats all employ some form of variable-length encoding* (see Figure 1). Block formats outperform optimal elementwise formats by effectively allocating their scale bits to represent the block maximum. Tensor-scaled formats can be effective if sparse outliers are stored separately, again implying variable bits-per-element. Finally, quantisation followed by lossless compression employs variable length explicitly and doesn't benefit from block scaling or sparse outlier removal.

Contributions Our work explains the performance of various quantisation schemes through the lens of optimising KL divergence and weight statistics. Towards this, we propose $\sqrt[3]{\mathrm{p}}$ block-scaled Normal, Laplace and Student-t non-uniform quantisers. Additionally, we propose the signmax scaling scheme, as well as using Fisher information to determine optimal bit width per parameter. These are, to the best of our knowledge, novel contributions.

[1]Graphcore Research. Correspondence to: Douglas Orr <douglaso@graphcore.ai>, Luka Ribar <lukar@graphcore.ai>.

Presented at the 3^{rd} Efficient Systems for Foundation Models Workshop at the International Conference on Machine Learning, Vancouver, Canada.

2. Optimal Quantisation Formats

Our objective is to minimise the expected KL divergence between the predictions of a quantised model and those of the original model. This KL divergence between predictive distributions $\mathrm{p}_\theta(y \mid x)$ and $\mathrm{p}_{\tilde{\theta}}(y \mid x)$ of output y given input x can be approximated as

$$\mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right) \approx \frac{1}{2} \sum_t \bar{f}_t \cdot \sum_{i \in T_t} (\tilde{\theta}_i - \theta_i)^2, \qquad (1)$$

where θ are the original parameters, $\tilde{\theta}$ are quantised, $\tilde{\theta} = \mathtt{dequantise}(\mathtt{quantise}(\theta))$, $\bar{f}_t$ is the average of the diagonal of the Fisher information matrix for tensor t and T_t are the parameter indices corresponding to t. See Appendix A for further explanation and Figures 7 and 8 for an empirical evaluation of how well this equation predicts KL divergence between pretrained LLM outputs when a single tensor is perturbed with noise.

To minimise this approximation to the KL divergence, we must minimise the squared reconstruction error of each tensor, subject to a bit-width constraint. This problem is well-studied in classical quantisation literature.

Fixed-length-coding This formulation constrains the size of each encoded element $q_i := \mathtt{quantise}^{\mathrm{elem}}(\theta_i)$ where $q_i \in Q^{\mathrm{elem}}$ and $|Q^{\mathrm{elem}}| \leq b$ for a bit width target b. For samples from an unknown distribution, the Lloyd-Max algorithm (Lloyd, 1982; Max, 1960) i.e. 1D k-means can be used to choose the codepoints Q^{elem}. For a known distribution $\theta_i \sim \mathcal{D}$, the density of codepoints should be proportional to the cube root of the pdf of $\mathcal{D}$ (Panter & Dite, 1951). We outline such *cube root density* ($\sqrt[3]{\mathrm{p}}$) quantisers for Normal, Laplace and Student-t distributions in Appendix B.

Block-absmax scaling In order to avoid truncating extreme values, calculate $n := \max_{i \in B} |\theta_i|$ over a block of B elements and store $\mathtt{quantise}^{\mathrm{elem}}(\theta_i / n)$ for each element (Rouhani et al., 2023). Observing that the scaled distribution $\mathcal{D}^{\mathrm{elem}}$ is well-approximated by mixture of a truncated-and-scaled $\mathcal{D}$ and a two-point distribution at ± 1 (Figure 12), we derive $\sqrt[3]{\mathrm{p}}$ quantisers for block-scaled data. Example code for Student-t data is given in Listing 1 and other distributions in Appendix F.

Block-signmax scaling We also consider a variation which normalises a block according to the signed value which is the absolute maximum within the block, i.e. $n^{\mathrm{signmax}} := \theta_{\hat{i}}$ where $\hat{i} = \arg\max_{i \in B} |\theta_i|$. The benefit is that the block maximum is known to be at $+1$ (rather than ± 1 for block-absmax scaling), so an element format with an even number of codepoints can allocate a pair of special codepoints at 0 and $+1$, while being otherwise symmetric (see Figure 2).

RMS scaling In order to assume a distribution of θ_i, this scaling scheme sets $n^{\mathrm{RMS}} := \sqrt{\frac{1}{B} \sum_{i \in B} \theta_i^2}$. Either moment matching to the $\mathcal{D}$ assumed by the quantiser or search can then be used to select quantiser parameters.

Listing 1 $\sqrt[3]{\mathrm{p}}$ block-absmax Student-t quantiser

```
b, block_size, df = 4, 64, 7

df_ = (df - 2) / 3
scale = ((2 * log(block_size / pi)) ** ((3-df)/(2*df))
         * block_size ** (-1/df)
         * sqrt(3))

c0, c1 = scipy.stats.t.cdf([-1, 1], df_, scale=scale)
p = torch.linspace(c0, c1, 2**b)
Q = torch.tensor(scipy.stats.t.ppf(p, df_, scale=scale))

def quantise(x): return torch.bucketize(x, (Q[1:]+Q[:-1])/2)
def dequantise(i): return Q[i]
```

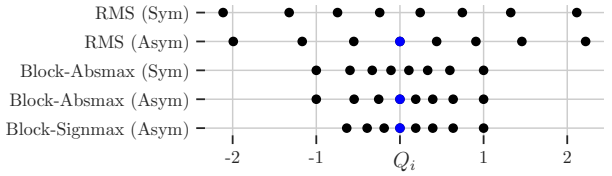

Figure 2. 3-bit $\sqrt[3]{\mathrm{p}}$ codepoint distributions for normally distributed data, illustrating RMS, absmax and signmax scaling methods and symmetric/asymmetric variants (with $B = 64$ for block formats). The principal benefit of asymmetric variants is that they have an encoding for **0**. INT formats are asymmetric, while most floating-point formats are symmetric but represent zero twice (± 0).

Variable-length-coding An alternative formulation applies an entropy constraint on q_i, assuming that the quantised values are encoded using a lossless compressor that approaches the Shannon (1948) limit. In this case, the optimal quantisation before compression is uniform (integer) quantisation, see Appendix A.4.

Variable bit-width allocation Equation (1) indicates that Fisher information can predict KL divergence due to quantisation. This suggests that there may be an optimal *variable* allocation of bits across parameter tensors, while respecting the average bit-width constraint at the model level. This scheme should allocate more bits to tensors that are more "sensitive", i.e., having higher Fisher information. We derive the following *variable bit allocation* scheme:

$$b_t^\star := b^0 + \log_2 \mathrm{RMS}(\theta_{T_t}) + \frac{1}{2} \log_2 \bar{f}_t, \qquad (2)$$

where $b_t^\star$ is the bit width of tensor t and b^0 is chosen to satisfy the overall size constraint. Intuitively, if tensor `a` has $4\times$ the Fisher information of tensor `b`, or $2\times$ the RMS, `a` uses 1 more bit than `b`. See Figure 10 for an example bit allocation and Appendix A.5 for a derivation.

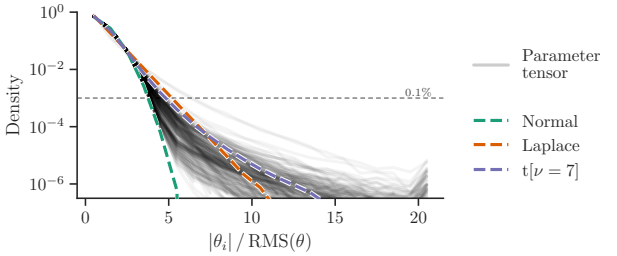

Figure 3. A histogram of absolute parameter values for Llama 3.1 8B. As we care about tails not scales (the overall scale of a tensor is easily absorbed into a scaling factor), we divide each parameter value by the RMS of the tensor. See Figure 23 for other models.

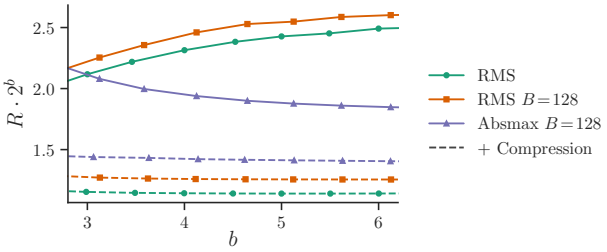

Figure 4. The error/size tradeoff for iid Student-t [$\nu = 5$] data with various optimal quantisers (hue). Surprisingly, block absmax quantisers can outperform tensor RMS formats for iid data, even though there is no inherent block structure. However this situation is reversed when adding optimal compression, implying that block absmax quantisers exploit some form of variable-length coding.

3. Analysis — Simulated Data

Motivated by the parameter statistics of Figure 3, we consider $\mathcal{D} \in \{\text{Normal}, \text{Laplace}, \text{Student-t}\}$ in turn. Using the methods described above, we compare optimal formats with block absmax or tensor RMS scaling on iid data from each distribution. Our aim is to establish whether there are benefits to block absmax formats for iid data, and where those benefits come from. Unless noted, all experiments in this section use `bfloat16` block scale and symmetric codepoints. We report R, the ratio of RMS error to parameter RMS and often plot $R \cdot 2^b$ to make diagonal error/bits trade-off lines horizontal. See Appendix D for details and more results.

Block formats exploit variable-length encoding. Our main result is the tradeoff between error and bit width for various scaling strategies with optimal quantisers, Figures 4 and 15. We were surprised to find that block absmax formats can outperform tensor RMS formats using optimal element quantisers, even for iid simulated data. An explanation is suggested when we add lossless compression, which uses a variable number of bits to encode each value. With compression, the advantage of block formats disappears and tensor RMS scaling performs best. This suggests a perspective on the benefit of block absmax formats — instead of viewing them as a way to avoid clipping, we can view them as a variable-length code, using additional (scale) bits to encode the block maximum. Since they outperform optimal fixed-length codes, they must somehow exploit variable-length coding. While the exact mechanism isn't clear, we provide a rough illustration of the effective code length in Figure 14.

Additional observations We compare standard formats against optimal block element formats in Figure 16. Here we see that NF4 (Dettmers et al., 2023) is not optimal for RMS error across different block sizes, and that `E2M1` is consistently better than `INT4` and `E3M0`. For floating-point formats, Figure 17 shows that the optimal number of exponent bits generally doesn't change as the total bit allocation grows. This is expected, since exponent bits govern the shape of the quantisation density function while mantissa bits govern the resolution, and the optimal shape should remain fixed. Figure 18 shows the benefit of using 4-10 scale mantissa bits over `E8M0`. In Figure 22, we see that an elementwise Huffman code approaches the theoretical compression performance.

4. Experiments

We evaluate a wide variety of weight formats described above for direct-cast quantisation of pretrained language models from the Llama 3, Qwen 2.5, Gemma 3 and Phi 4 families. Direct-cast quantisation, sometimes called round-to-nearest, is a simple quantisation technique that performs one-shot conversion, without using data or fine-tuning to recover task performance. The primary comparison is an efficiency trade-off between top-k KL-divergence of quantised and original model predictions, D_{KL}, against average bits per parameter, b. We also use $\rho := \mathrm{D}_{\mathrm{KL}} \cdot 2^{2b}$ as a measure of inefficiency of representation. See Appendix E for further details of our methodology.

Uniform quantisation with compression is efficient. Figures 1 and 6 show that a uniform grid quantiser followed by optimal lossless compression consistently outperforms other approaches. Here, tensor RMS scaling is sufficient and can be folded into the grid resolution. Huffman coding achieves near-optimal compression, as shown in Figure 6.

Variable-length encoding is necessary for good performance. Figures 1 and 6 also demonstrate the characteristics of near-optimal formats without lossless compression. All employ block or channel absmax scaling and/or separate storage of sparse outliers (Kim et al., 2024). Our search over a wide range of element formats was unable to find fixed-length schemes that can reach the same performance as block or sparse formats. Consistent with our observations of Section 3, this indicates that they exploit variable-length encoding. We also observe that there is no benefit in adding block absmax scaling or sparse outlier removal to lossless compression (see Figure 24), indicating that their benefit

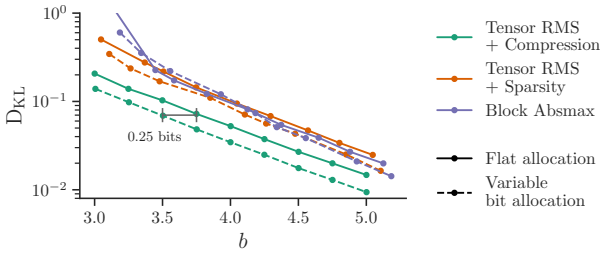

Figure 5. The performance of the Fisher-based variable bit allocation scheme of Equation (2) for Llama 3.1 8B. The tradeoff curve for shows a general shift to the left, although some settings for absmax scaling are degraded. See also Figures 26 and 27.

comes from the same source. Alternatively, random rotations can mitigate the poor performance of fixed-length codes Figure 25, but cannot match the performance of optimal variable-length codes.

Variable bit allocation improves efficiency. The variable bit allocation scheme promises to reach the same overall compression level for a model with less degradation by allocating more bits to parameters with higher Fisher information. Figure 26 shows that this is the case, with an improvement across 8 of 11 models and different formats. The exception is Gemma models, where our prediction of KL based on Fisher information also breaks down (Figure 8).

Additional observations In Figure 28, we compare element formats against a Student-t baseline over $b \in [3..5]$, indicating the Student-t format performs best in almost all cases. Figure 29 compares 4-bit $\sqrt[3]{p}$ formats against `NF4` and `SF4` baselines: $\sqrt[3]{p}$ Normal and `NF4` show similar performance, but $\sqrt[3]{p}$ Student-t outperforms `SF4`. For block absmax scaling, Figure 30 confirms the results from simulated data — a block size near 128 and 4-7 scale mantissa bits perform best. We compare symmetric, asymmetric and signmax variants for block scaling with integer or Student-t element formats in Figure 31, finding that signmax delivers a consistent improvement across models and particularly for small $b \approx 3$. The performance of symmetric vs asymmetric variants is inconsistent across models. In Figure 32 we evaluate different ways to choose the quantiser scale. Search to minimise R is better than moment matching when using RMS scaling, but can be harmful for absmax scaling unless weighted by the per-parameter Fisher information.

5. Discussion

Related work Integer and floating-point formats have been extensively evaluated for training and inference (Wang & Kanwar, 2019; Wang et al., 2018; Noune et al., 2022; Sun et al., 2020; Gernigon et al., 2023; Dettmers & Zettlemoyer, 2023; Liu et al., 2025). Non-linear formats allow greater flexibility and can be implemented using look-up tables to accelerate communication without requiring specific hardware support. Dettmers et al. (2023) introduce `NF4`, aimed to be the theoretically optimal 4-bit format under the assumption of normally-distributed weights; similarly, Dotzel et al. (2024) introduce `SF4`, assuming a Student-t distribution. In both approaches, the authors derive the codebook so that each quantisation bin is equally populated under the assumed distribution. However, this does not lead to optimal codes in terms of the RMS error, which we instead motivate and pursue in our current work. Yoshida (2023) identifies that block size can have a significant impact on the scaled distribution, and derives `AF4` assuming a normal distribution. `AF4` is similar to our proposed block absmax $\sqrt[3]{p}$ Normal format, but optimises for absolute rather than squared error and uses a different approximation for the block maximum. In SqueezeLLM (Kim et al., 2024), the authors augment Lloyd-Max codepoint selection (Lloyd, 1982; Max, 1960) with Fisher sensitivity information, improving the efficacy of the obtained codebook. Their results highlight the utility of Fisher sensitivity, which we use to allocate variable precision across weight tensors. Number formats can be augmented in various ways: storing sparse outliers separately (Dettmers et al., 2022a; 2024), using random or trained rotations to suppress outliers (Tseng et al., 2024; Ashkboos et al., 2024; Liu et al., 2024) or with lossless compression (Han et al., 2016; Zhang et al., 2025).

Limitations Our analysis and experiments are restricted to direct-cast quantisation, which is a suboptimal method. While this does not affect our conclusion that block and sparse-outlier formats exploit variable-length coding, it does limit the direct applicability of the formats that we derive under our assumptions. Additionally, we consider only the weight compression constraint, with no consideration of compute efficiency. Finally, we note that we only tested transformer LLMs in the 0.5–14B parameter range, although the optimal quantisation framework is much more general than this.

Conclusions Framing weight quantisation as an optimisation problem highlights the importance of choosing the right compression constraint. Under a codebook length constraint, $\sqrt[3]{p}$ and Lloyd-Max quantisers are optimal. Under an entropy constraint, uniform quantisation followed by lossless compression is optimal. We have shown that both block absmax and sparse outlier formats can be viewed as forms of variable-length encoding, exploiting the advantage of the entropy constraint. For the format designer, this suggests opportunities to develop practical formats that further close the gap with lossless compression. For the format user, it suggests coherent constructions to use, for example, block absmax formats or tensor RMS formats with sparse outliers. Finally, the fact that Fisher information predicts performance loss due to quantisation motivates variable bit-width allocation across tensors, enabling a principled replacement for heuristic schemes.

Acknowledgements

We would like to thank Paul Balança, Robert Hu, Callum McLean, Luke Hudlass-Galley and Andrew Fitzgibbon for their invaluable advice and support throughout the development of this work.

References

Abdin, M. I., Aneja, J., Behl, H. S., Bubeck, S., Eldan, R., Gunasekar, S., Harrison, M., Hewett, R. J., Javaheripi, M., Kauffmann, P., Lee, J. R., Lee, Y. T., Li, Y., Liu, W., Mendes, C. C. T., Nguyen, A., Price, E., de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Wang, X., Ward, R., Wu, Y., Yu, D., Zhang, C., and Zhang, Y. Phi-4 technical report. *CoRR*, 2024.

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. GQA: training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 4895–4901. Association for Computational Linguistics, 2023.

Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pp. 1027–1035. SIAM, 2007.

Ashkboos, S., Mohtashami, A., Croci, M. L., Li, B., Cameron, P., Jaggi, M., Alistarh, D., Hoefler, T., and Hensman, J. Quarot: Outlier-free 4-bit inference in rotated llms. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Dettmers, T. and Zettlemoyer, L. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 7750–7774. PMLR, 2023.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022a.

Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. 8-bit optimizers via block-wise quantization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. QLoRA: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

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. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Dotzel, J., Chen, Y., Kotb, B., Prasad, S., Wu, G., Li, S., Abdelfattah, M. S., and Zhang, Z. Learning from students: Applying t-distributions to explore accurate and efficient formats for llms. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Rozière, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I. M., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., and et al. The llama 3 herd of models. *CoRR*, 2024.

Gernigon, C., Filip, S., Sentieys, O., Coggiola, C., and Bruno, M. Low-precision floating-point for efficient onboard deep neural network processing. *CoRR*, 2023.

Gersho, A. and Gray, R. M. *Vector quantization and signal compression*, volume 159 of *The Kluwer international series in engineering and computer science*. Kluwer, 1991. ISBN 978-0-7923-9181-4.

Gish, H. and Pierce, J. N. Asymptotically efficient quantizing. *IEEE Trans. Inf. Theory*, 14(5):676–683, 1968.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Huffman, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A. M., Goedeckemeyer, A., Saade, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A. S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C. L., Choquette-Choo, C. A., Carey, C., Brick, C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D. S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F., Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Plucinska, H., Singh, H., Mehta, H., Lehri, H. T., Hazimeh, H., Ballantyne, I., Szpektor, I., and Nardini, I. Gemma 3 technical report. *CoRR*, 2025.

Kim, S., Hooper, C., Gholami, A., Dong, Z., Li, X., Shen, S., Mahoney, M. W., and Keutzer, K. Squeezellm: Dense-and-sparse quantization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

Kunstner, F., Hennig, P., and Balles, L. Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4158–4169, 2019.

Leadbetter, M., Lindgren, G., and Rootzen, H. *Extremes and Related Properties of Random Sequences and Processes*. Springer Series in Statistics. Springer New York, 2012. ISBN 9781461254492.

Lippens, S. dahuffman, 2017. URL https://pypi.org/project/dahuffman/.

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. *CoRR*, 2024.

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. *CoRR*, 2025.

Lloyd, S. P. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982.

Max, J. Quantizing for minimum distortion. *IRE Trans. Inf. Theory*, 6(1):7–12, 1960.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Noune, B., Jones, P., Justus, D., Masters, D., and Luschi, C. 8-bit numerical formats for deep neural networks. *CoRR*, 2022.

Panter, P. and Dite, W. Quantization distortion in pulse-count modulation with nonuniform spacing of levels. *Proceedings of the IRE*, 39(1):44–48, 1951.

Rouhani, B. D., Zhao, R., More, A., Hall, M., Khodamoradi, A., Deng, S., Choudhary, D., Cornea, M., Dellinger, E., Denolf, K., Stosic, D., Elango, V., Golub, M., Heinecke, A., James-Roxby, P., Jani, D., Kolhe, G., Langhammer, M., Li, A., Melnick, L., Mesmakhosroshahi, M., Rodriguez, A., Schulte, M., Shafipour, R., Shao, L., Siu, M. Y., Dubey, P., Micikevicius, P., Naumov, M., Verilli, C., Wittig, R., Burger, D., and Chung, E. S. Microscaling data formats for deep learning. *CoRR*, 2023.

Shannon, C. E. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

Sun, X., Wang, N., Chen, C., Ni, J., Agrawal, A., Cui, X., Venkataramani, S., Maghraoui, K. E., Srinivasan, V., and Gopalakrishnan, K. Ultra-low precision 4-bit training of deep neural networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Tseng, A., Chee, J., Sun, Q., Kuleshov, V., and Sa, C. D. Quip#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

Tunstall, L., von Werra, L., and Wolf, T. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, 2022. ISBN 9781098103248. URL https://books.google.co.uk/books?id=_0uezgEACAAJ.

Wang, N., Choi, J., Brand, D., Chen, C., and Gopalakrishnan, K. Training deep neural networks with 8-bit floating point numbers. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 7686–7695, 2018.

Wang, S. and Kanwar, P. Bfloat16: The secret to high performance on cloud tpus, 2019. URL https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus. Accessed: 2025-05-10.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pp. 38–45. Association for Computational Linguistics, 2020.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report. *CoRR*, 2024.

Yoshida, D. NF4 isn't information theoretically optimal (and that's good). *CoRR*, 2023.

Zador, P. L. Asymptotic quantization error of continuous signals and the quantization dimension. *IEEE Trans. Inf. Theory*, 28(2):139–148, 1982.

Zhang, T., Sui, Y., Zhong, S., Chaudhary, V., Hu, X., and Shrivastava, A. 70% size, 100% accuracy: Lossless llm compression for efficient gpu inference via dynamic-length float. *CoRR*, abs/2504.11651, 2025.

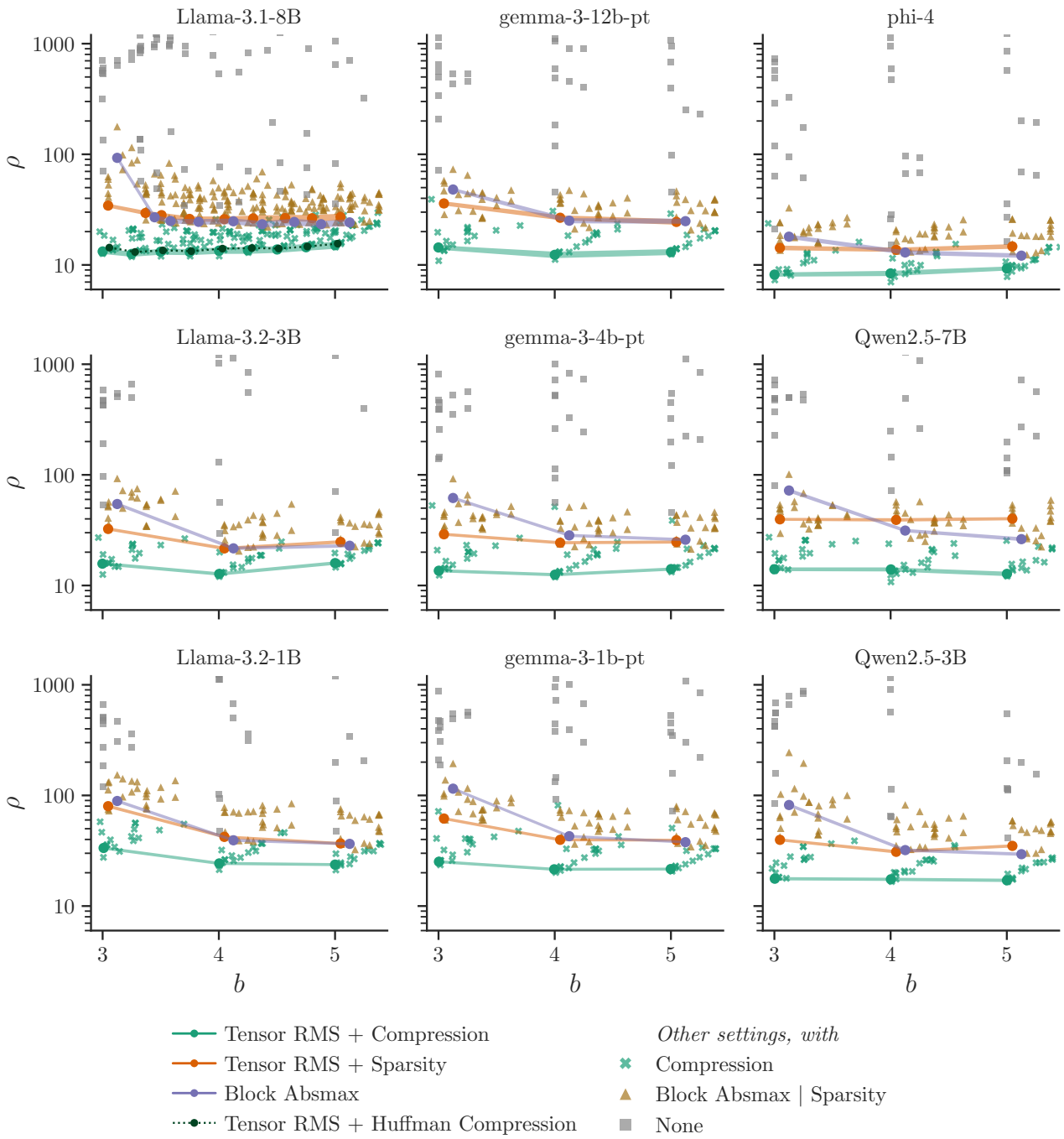

Figure 6. The trade-off between average bits per element and scaled top-k KL divergence over different block scaling schemes (RMS, absmax), sparse outlier removal (on, off) and optimal lossless compression (on, off). This is similar to Figure 1, but on the vertical axis we show $\rho := \mathrm{D}_{\mathrm{KL}} \cdot 2^{2b}$ to flatten the curve based on the error scaling limit of Zador (1982). We also show that simple per-element Huffman coding performs very close to the optimal compression which assumes the Shannon limit (Llama 3.1 8B only). Results are highly consistent across model families and sizes.

Note: shaded lines show ± 2 standard error over the evaluation data. Where trends appear noisy but error bars are tight (especially in Figure 1), this is due to the model itself — we are unable to quantify uncertainty due to model parameters, since there is only one independent fully-trained checkpoint for each family & size.

A. Optimal Quantisation Formats

In this section, we explain our framework for format design in detail. We begin by defining the overall optimisation problem at the model level, showing how this objective can be reduced to minimising the squared error of individual tensors through appropriate approximations. Next, we present solutions for RMS and absmax scaling schemes, as well as a lossless compression scheme. Finally, we revisit model-level optimisation by proposing a scheme for optimal bit allocation across tensors.

A.1. Model KL

Consider a probabilistic model with output y conditioned on input drawn from a dataset $x \sim \mathcal{X}$, where the pdf $\mathrm{p}_\theta(y \mid x)$ is represented by a neural network, parametrised by $\theta \in \mathbb{R}^{|\theta|}$. We wish to find a quantised parameter vector $\tilde{\theta}^\star$ which is in $\tilde{\Theta} \subset \mathbb{R}^{|\theta|}$, minimising the expected KL divergence of $\mathrm{p}_{\tilde{\theta}}(y \mid x)$ against the reference model,

$$\begin{aligned} \tilde{\theta}^\star &\coloneqq \arg\min_{\tilde{\theta} \in \tilde{\Theta}} \mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right), \\ \mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right) &\coloneqq \mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\mathop{\mathbb{E}}_{\mathrm{p}_\theta(y|x)}\left[\log \frac{\mathrm{p}_\theta(y \mid x)}{\mathrm{p}_{\tilde{\theta}}(y \mid x)}\right]\right], \\ &\text{with } |\tilde{\Theta}| \leq 2^{|\theta| \cdot b}. \end{aligned} \tag{3}$$

The final line induces a compression constraint on $\tilde{\Theta}$, specifying an average of b bits per parameter. Now we introduce a sequence of approximations.

2nd order approximation Consider a Taylor expansion of $\mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right)$ around $\tilde{\theta} = \theta$,

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right) \approx\ & \mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_\theta\right) \\ & + (\tilde{\theta} - \theta)^\top \left(\nabla_{\tilde{\theta}} \mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right)\right)_{\tilde{\theta}=\theta} \\ & + \frac{1}{2} \cdot (\tilde{\theta} - \theta)^\top \left(\nabla^2_{\tilde{\theta}} \mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right)\right)_{\tilde{\theta}=\theta} (\tilde{\theta} - \theta), \end{aligned}$$

and observe that the first two terms $= 0$, since $\tilde{\theta} = \theta$ is a minimum. Now we expand the second derivative

$$\begin{aligned} \nabla^2_{\tilde{\theta}} \mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right) &= -\mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\mathop{\mathbb{E}}_{\mathrm{p}_\theta(y|x)}\left[\nabla^2_{\tilde{\theta}} \log \mathrm{p}_{\tilde{\theta}}(y \mid x)\right]\right], \\ &= \mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\mathop{\mathbb{E}}_{\mathrm{p}_\theta(y|x)}\left[-\frac{\nabla^2_{\tilde{\theta}} \mathrm{p}_{\tilde{\theta}}(y \mid x)}{\mathrm{p}_{\tilde{\theta}}(y \mid x)} + \frac{\nabla_{\tilde{\theta}} \mathrm{p}_{\tilde{\theta}}(y \mid x)(\nabla_{\tilde{\theta}} \mathrm{p}_{\tilde{\theta}}(y \mid x))^\top}{\mathrm{p}_{\tilde{\theta}}(y \mid x)^2}\right]\right], \\ &= \mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\sum_y -\nabla^2_{\tilde{\theta}} \mathrm{p}_{\tilde{\theta}}(y \mid x)\right] + \mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\mathop{\mathbb{E}}_{\mathrm{p}_\theta(y|x)}\left[\frac{\nabla_{\tilde{\theta}} \mathrm{p}_{\tilde{\theta}}(y \mid x)(\nabla_{\tilde{\theta}} \mathrm{p}_{\tilde{\theta}}(y \mid x))^\top}{\mathrm{p}_{\tilde{\theta}}(y \mid x)^2}\right]\right]. \end{aligned}$$

The first term $= 0$, since the second derivative can be moved outside the sum. For the second term, we substitute the Fisher information $F \in \mathbb{R}^{|\theta| \times |\theta|}$, defined as

$$\begin{aligned} F &\coloneqq \mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\mathop{\mathbb{E}}_{\mathrm{p}_\theta(y|x)}\left[(\nabla_\theta \log \mathrm{p}_\theta(y \mid x))(\nabla_\theta \log \mathrm{p}_\theta(y \mid x))^\top\right]\right] \\ &= \mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\mathop{\mathbb{E}}_{\mathrm{p}_\theta(y|x)}\left[\frac{\nabla_{\tilde{\theta}} \mathrm{p}_{\tilde{\theta}}(y \mid x)(\nabla_{\tilde{\theta}} \mathrm{p}_{\tilde{\theta}}(y \mid x))^\top}{\mathrm{p}_{\tilde{\theta}}(y \mid x)^2}\right]\right]. \end{aligned} \tag{4}$$

Therefore, $\left(\nabla^2_{\tilde{\theta}} \mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right)\right)_{\tilde{\theta}=\theta} = F$, and

$$\mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right) \approx \frac{1}{2}(\tilde{\theta} - \theta)^\top F\,(\tilde{\theta} - \theta). \tag{5}$$

Diagonal approximation As a further simplification, we assume the cross terms are small, $F_{ij} \approx 0 \quad \forall\, i \neq j$, so we can simplify the approximate KL divergence to

$$\mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right) \approx \frac{1}{2} \sum_i F_{ii} \cdot (\tilde{\theta}_i - \theta_i)^2. \tag{6}$$

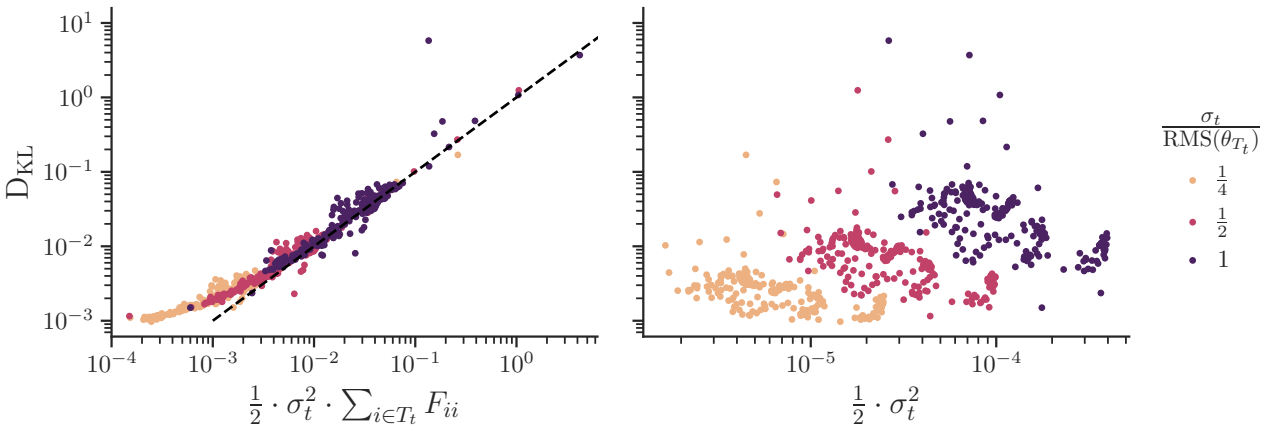

Figure 7. The KL divergence from modifying each parameter tensor in turn with iid noise, compared against *(Left)* the predicted KL divergence from Equation (6), and *(Right)* the scale of perturbation without using Fisher information. For each parameter tensor of Llama 3.1 8B, we perturb $\tilde{\theta}_{T_t} = \theta_{T_t} + \sigma_t \cdot \epsilon$, for a range of σ_t and with $\epsilon \sim \mathrm{N}^{|T_t|}(0, 1)$, and measure the top-$k$ KL divergence of outputs against the original model. This result indicates that Fisher information is able to predict KL divergence — tensors with higher Fisher information are more sensitive to perturbation.

Tensor-constant approximation Deep learning model parameters can be partitioned into *tensors*, often the maximal sets of parameters that can be applied in parallel to the intermediate activations — in other words, where the input to the forward pass operation of any scalar parameter does not depend on any other parameter from the same tensor. It is convenient to encode each tensor using a single format, although different tensors may use different formats, so we expand $\tilde{\Theta} = \prod_t \tilde{\Theta}_t$, and $\tilde{\theta}_{T_t} \in \tilde{\Theta}_t$ where $T_t \in \mathbb{N}^{|T_t|}$ is a vector of parameter indices belonging to tensor t.

For much of our analysis, we further approximate the diagonal of the Fisher matrix as constant within each parameter tensor ($F_{ii} = \bar{f}_t$, for $i \in T_t$). With this approximation, the increase in KL divergence due to quantisation is a weighted sum across parameter tensors of the unweighted squared error,

$$\mathrm{D_{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right) \approx \frac{1}{2} \sum_t \bar{f}_t \cdot \sum_{i \in T_t} (\tilde{\theta}_i - \theta_i)^2. \tag{1}$$

This form is convenient, as the squared error is easy to compute and the diagonal of F can be estimated with computational and memory costs comparable to a few steps of SGD. We test this equation for predicting the KL divergence after perturbing with iid normal noise in Figures 7 and 8.

A.2. Optimal tensor formats for known distributions

We now turn to the problem of designing a format to represent a single parameter tensor. The formats we consider all operate on blocks of data — all or part of the tensor. The *block size* B is fixed within a tensor, but may vary across tensors. For the ith block of the tth tensor, we wish to quantise the parameters sub-vector $\theta_{C_{ti}}$, given block indices $C_{ti} \in \mathbb{N}^B$, but to aid readability we will drop the indices and use θ directly.

Using the approximate relationship given by Equation (1), KL divergence is minimised by minimising the squared reconstruction error of each parameter. We therefore consider the following optimisation problem for a block of parameters $\theta \in \mathbb{R}^B$:

$$\begin{aligned} \text{find: } & \texttt{quantise} : \mathbb{R}^B \to Q \ , \ \texttt{dequantise} : Q \to \mathbb{R}^B \\ \text{to minimise: } & E^2 \coloneqq \sum_{i \in [1..B]} (\theta_i - \texttt{dequantise}(\texttt{quantise}(\theta))_i)^2 \\ \text{such that: } & |Q| \leq 2^{B \cdot b}, \end{aligned} \tag{7}$$

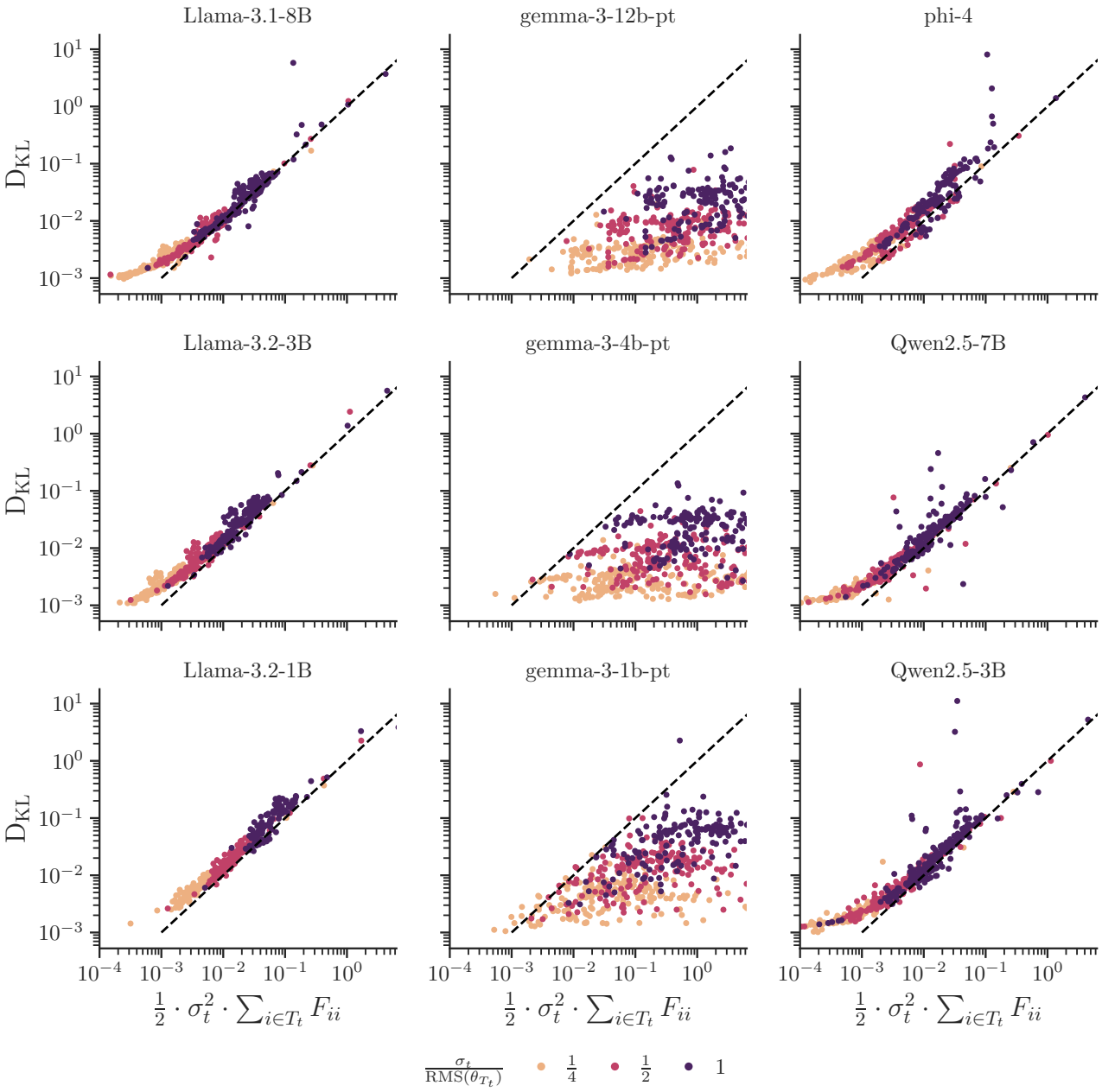

Figure 8. How Fisher information predicts KL divergence given a single-tensor iid random perturbation (as per Figure 7). Most models show a clear trend following the prediction rule, with some outliers. In contrast, the whole Gemma 3 family fails, suggesting a family-specific or code-specific issue. We believe that the flattening of the curve for small noise levels (where the measured KL is higher than predicted) may be due to the `bfloat16` precision of the model computation amplifying the noise.

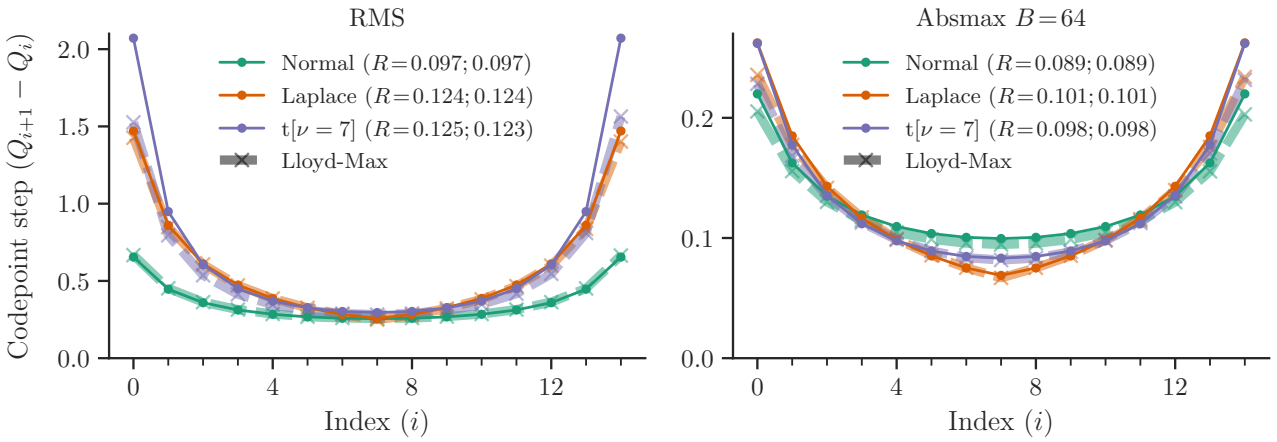

Figure 9. 4-bit quantisation curve gradients for Normal, Laplace, Student-t distributions, showing strong agreement between cube root density and Lloyd-Max. The legend shows relative quantisation error R for "(cube root quantiser; Lloyd-Max)" over data from the appropriate distribution. *(Left)* RMS-scaled formats. The cube root density rule breaks down with the heavy tails of Student-t. *(Right)* absmax-scaled formats. The discrepancy at the extremes occurs because the cube root quantiser has a special case for ± 1, whereas Lloyd-Max treats it as a single distribution to quantise.

where the set of quantised representations Q is subject to a compression constraint of b bits per element in the block of B elements. For our analysis, θ is iid, with $\theta_i \sim \mathcal{D}$ from a Normal, Laplace or Student-t distribution.

Cube root density quantiser When $B = 1$ and $\mathcal{D}$ is known with pdf $\mathrm{p}^{\mathcal{D}}$, we can use the cube root density quantiser (Panter & Dite, 1951) to minimise the error:

$$
\begin{aligned}
&Q^{\text{elem}} \subset \mathbb{R}\,,\ \mathtt{density}(Q^{\text{elem}}) \propto \sqrt[3]{\mathrm{p}^{\mathcal{D}}} \\
&\mathtt{quantise}^{\text{elem}}(\theta) = \operatorname{argmin}_{q \in Q^{\text{elem}}} |\theta - q| \\
&\mathtt{dequantise}^{\text{elem}}(q) = q,
\end{aligned}
$$

When $\mathcal{D}$ is Normal, Laplace or Student-t, we observe that $\sqrt[3]{\mathrm{p}^{\mathcal{D}}}$ is proportional to the pdf of $\mathcal{D}'$, the same parametric distribution as $\mathcal{D}$ but with different parameters. Therefore Q^{elem} can be derived from the inverse cdf of $\mathcal{D}'$, see Appendix B.3 for details and F for code examples.

Linear scaling When $B > 1$, if we assume $\mathcal{D}$ is a known parametric distribution but with *unknown scale*, we can adapt the scalar quantisation technique above:

$$
\begin{aligned}
&Q^{\text{linear}} := \mathbb{R} \times (Q^{\text{elem}})^B\ ,\ \mathtt{norm} : \mathbb{R}^B \to \mathbb{R} \\
&\mathtt{quantise}^{\text{linear}}(\theta) = \left[n,\ \mathtt{quantise}^{\text{elem}} \left(\frac{\theta_i}{n} \right)_{\forall i \in [1..B]} \right] \quad \text{where } n = \mathtt{norm}(\theta) \\
&\mathtt{dequantise}^{\text{linear}}(n, q)_i = n \cdot \mathtt{dequantise}^{\text{elem}}\left(q_i\right),
\end{aligned}
$$

In this scheme, we store and use a statistic $\mathtt{norm}(\theta)$ to normalise the block values, such that $\mathtt{quantise}^{\text{elem}}$ can assume some properties of its input distribution, $\mathcal{D}^{\text{elem}}$. Note that these equations cannot yet obey the compression constraint, as Q^{linear} is uncountable; for a practical scheme we must instead store $\mathtt{quantise}^{\text{scale}}(n)$ using an appropriate format.

RMS scaling Applying linear scaling with $\mathtt{norm}(\theta) = \sqrt{\frac{1}{B}\sum_i \theta_i^2}$, we obtain RMS scaling. If we replace the random variable n with a point estimate at its expected value, then $\frac{\theta_i}{n}$ follows $\mathcal{D}^{\text{elem}}$, a scaled version of $\mathcal{D}$ such that $\mathbb{E}\left[\theta_i^2\right] = 1$. Moment matching of the RMS can provide the parameters of $\mathcal{D}^{\text{elem}}$ needed for an optimal format according to the cube root density rule (Table 1). See Figure 9 for example quantisation curves including a comparison against Lloyd-Max, which is trained against transformed samples from $\mathcal{D}$ to optimise RMS error.

Absmax scaling Linear scaling with $\texttt{norm}(\theta) = \max_i |\theta_i|$ gives absolute-maximum scaling, a popular block quantisation scheme. In this case $\mathcal{D}^{\text{elem}}$ has support $-1 \leq \theta_i \leq 1$. Following Yoshida (2023), we consider $\mathcal{D}^{\text{elem}}$ as a mixture of two components: (1) the normalised maximum value, which is a transformed Bernoulli distribution and (2) the normalised distribution of everything else in the block, which was not the maximum. To approximate (2), we observe empirically that the marginal distribution of $\theta_{i \neq \arg\max_j |\theta_j|}$ is a good match to a truncated $\mathcal{D}$, where the truncation point is the block maximum (Figure 12). We then use a closed form approximation to $\mathbb{E}\left[\max_i |\theta_i|\right]$ (Table 1) to calculate the truncation points. To construct Q^{elem}, we always include ± 1, then use the inverse cdf of the truncated-and-scaled $\mathcal{D}$ to distribute the rest of Q^{elem} according to the cube root rule. Example quantisation curves are shown in Figure 9.

Signmax scaling Observing that the distribution of block-scaled data is well-approximated by a mixture of the maximum and non-maxima, it seems natural to also try *signmax* scaling. In this scheme, the block scale is set to the signed absolute maximum, $\texttt{norm}(\theta) = \theta_{\hat{i}}$ where $\hat{i} = \arg\max_i |\theta_i|$. The element format can then assume that the maximum is always at $+1$ (not ± 1) and allocate a pair of special codepoints $\{0, 1\}$ with the rest specified according to $\sqrt[3]{\mathrm{p}}$ (see Figure 2). This comes at the cost of requiring a sign bit for the block scale, i.e. $\frac{1}{B}$ bits per element.

Symmetric/Asymmetric variants One important detail is the representation of zero. Practical implementation considerations often provide an even number of codepoints, so allocating a codepoint for zero mandates asymmetry or waste. However, exact zero has been shown empirically to be valuable (Liu et al., 2025). The $\sqrt[3]{\mathrm{p}}$ scheme for allocating codepoints is easily adapted to provide symmetric and asymmetric variants for both RMS and absmax scaling (Figure 2). For block scaling the asymmetry is purely in resolution, while for RMS it naturally provides both additional resolution and range on one side.

A.3. Unknown distributions

In general, the distribution of parameters of neural networks after training is unknown, so we cannot apply the techniques described in Appendix A.2 directly. If we assume the data comes from a given distribution family there remains an unknown *scale* parameter, which must be estimated. In the case of absmax and signmax scaling it's natural to match the range of the element format with the normalised data. For RMS scaling, one solution is moment-matching — after normalising the data to have RMS $= 1$, we use the optimal quantiser for the assumed distribution with RMS $= 1$. Moment matching, however, isn't guaranteed to minimise E^2 (see Figure 21). Therefore we also evaluate per-tensor search to find the scale and any distribution parameters (e.g. Student-t ν) that minimise E^2. An alternative to assuming a canonical distribution is to directly optimise a non-uniform format against the data, to solve Equation (7). A standard solution is the Lloyd-Max algorithm (Lloyd, 1982; Max, 1960), i.e. 1D k-means. Both distribution parameter search and Lloyd-Max can solve a weighted objective, so we can drop the scaled-identity approximation to the Fisher information, and use $\mathrm{diag}(F)$ as a weight on the importance of each parameter, as proposed by Kim et al. (2024).

A.4. Optimal tensor formats with compression

An alternative approach is to first use a (lossy) quantiser such as those described in Appendix A.2 then follow it with a lossless compressor, operating on quantised data in Q. In this case we can optimise the quantiser using the objective from Equation (7), substituting the last line with an entropy coding constraint:

$$\ldots \text{such that: } \mathbb{E}\left[\mathcal{I}(\texttt{quantise}(\theta))\right] \leq B \cdot b$$

$$\text{where: } \mathcal{I}(q) = -\sum_{i \in [1..B]} \log_2 \mathrm{p}^{\mathcal{Q}}(q_i)\,.$$

I.e. $\mathcal{I}(q)$ computes the information content of a quantised block under a model of their values given by $\mathrm{p}^{\mathcal{Q}}$ and we assume an optimal compressor approaching the Shannon (1948) limit.

Compressed grid Under this new constraint if $B = 1$ the optimal distribution of elementwise codepoints is a uniform grid i.e. $\texttt{density}(Q^{\text{elem}}) = \text{const}$ (Gish & Pierce (1968), see also Appendix C.2). The probability model $\mathrm{p}^{\mathcal{Q}}$ for compression can either be estimated based on samples or derived by transforming $\mathcal{D}$ by $\texttt{quantise}(\theta)$, which in the case of elementwise quantisers is trivial: via the cdf or approximately via the pdf of $\mathcal{D}$.

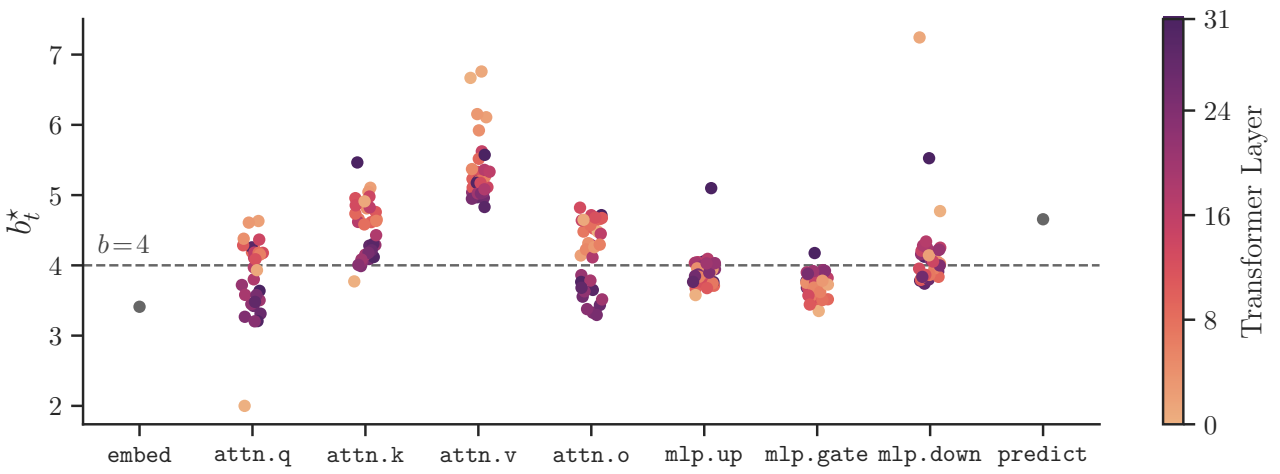

Figure 10. Variable allocation of bits across tensors in Llama 3.1 8B using Equation (2) with a target of 4 bits per parameter. For many element formats, $b_t^\star$ would need to be rounded to the nearest integer. Other members of the Llama 3 and Qwen 2.5 families show a similar trend of requiring additional bits for the attention key and value projections. We suspect this is due to grouped query attention (Ainslie et al., 2023), where the outputs of key and value projections are reused across a group of multiple attention heads.

A.5. Optimal bit-width allocation

In this section, we derive the variable bit-width allocation scheme of Equation (2). We start from the constant-per-tensor Fisher approximation to KL divergence of Equation (1), repeated here:

$$\mathrm{D}_{\mathrm{KL}}\left(\mathrm{p}_\theta \| \mathrm{p}_{\tilde{\theta}}\right) \approx \frac{1}{2} \sum_t \bar{f}_t \cdot \sum_{i \in T_t} (\tilde{\theta}_i - \theta_i)^2.$$

Now, to forecast how the squared error term depends on bit width, we use the asymptotic limit of Zador (1982), which can be stated as

$$\mathbb{E}\left[(\tilde{\theta}_i - \theta_i)^2\right] = \epsilon_t^2 \cdot \hat{\sigma}_t^2 \cdot 2^{-2 \cdot b'_t},$$

where b'_t is the bit width used for tensor t, ϵ_t depends on the distribution of θ and $\hat{\sigma}_t^2 := \frac{\sum_{i \in T_t} \mathbb{E}\left[\theta_i^2\right]}{N_t} \approx \mathrm{RMS}^2(\theta_{T_t})$ with $N_t := |T_t|$. This gives the optimisation

$$\begin{aligned} \text{minimise} \quad & J := \frac{1}{2} \sum_t N_t \cdot \bar{f}_t \cdot \epsilon_t^2 \cdot \hat{\sigma}_t^2 \cdot 2^{-2 \cdot b'_t}, \\ \text{subject to} \quad & \sum_t b'_t \cdot N_t \leq b \cdot \sum_t N_t. \end{aligned}$$

Using the Lagrange multiplier λ, and removing constant factors, we pursue the constrained optimisation,

$$\begin{aligned} J' &= \sum_t N_t \cdot \bar{f}_t \cdot \epsilon_t^2 \cdot \hat{\sigma}_t^2 \cdot 2^{-2 \cdot b'_t} + \lambda \cdot N_t \cdot (b'_t - b), \\ \frac{\mathrm{d}J'}{\mathrm{d}b_t^\star} &= -2 \cdot \ln 2 \cdot N_t \cdot \bar{f}_t \cdot \epsilon_t^2 \cdot \hat{\sigma}_t^2 \cdot 2^{-2 \cdot b_t^\star} + \lambda \cdot N_t = 0, \\ b_t^\star &= b^0 + \log_2 \hat{\sigma}_t + \frac{1}{2} \log_2 \bar{f}_t + \log_2 \epsilon_t, \end{aligned}$$

for some constant b^0. As a final approximation, we assume that $\epsilon_t = \text{const}$ across t, so it can be folded into b^0.

We show an example variable bit allocation computed from this procedure in Figure 10. Most tensors are ± 1 bit from the average, and there is a general trend toward representing some groups of tensors more accurately, e.g. `attn.v`.

Table 1. Statistics required for deriving optimal RMS and absmax scaled $\sqrt[3]{\mathrm{p}}$ quantisers. See Appendix B.3 for a derivation of parameters of $\mathcal{D}'$. The expected absmax is taken from extreme value theory (Leadbetter et al., 2012), or in the case of Student-t from our empirical approximation (see Figure 11). Note that γ is the Euler–Mascheroni constant.

Value	Normal (s)	Laplace (s)	Student-t (s, ν)
RMS $\sqrt{\mathbb{E}\left[\theta_i^2\right]}$	s	$\sqrt{2} \cdot s$	$\sqrt{\frac{\nu}{\nu-2}} \cdot s$
$\mathbb{E}\left[\max_{i \in [1..B]} \lvert\theta_i\rvert\right] \approx$	$\sqrt{2 \log \frac{B}{\pi}} \cdot s$	$(\gamma + \log B) \cdot s$	$\left(2 \log \frac{B}{\pi}\right)^{\frac{\nu-3}{2\nu}} \cdot B^{\frac{1}{\nu}} \cdot \sqrt{\frac{\nu}{\nu-2}} \cdot s$
$\mathcal{D}'$ params	$s' = \sqrt{3} \cdot s$	$s' = 3 \cdot s$	$\nu' = \frac{\nu-2}{3}$, $s' = \sqrt{\frac{\nu}{\nu'}} \cdot s$

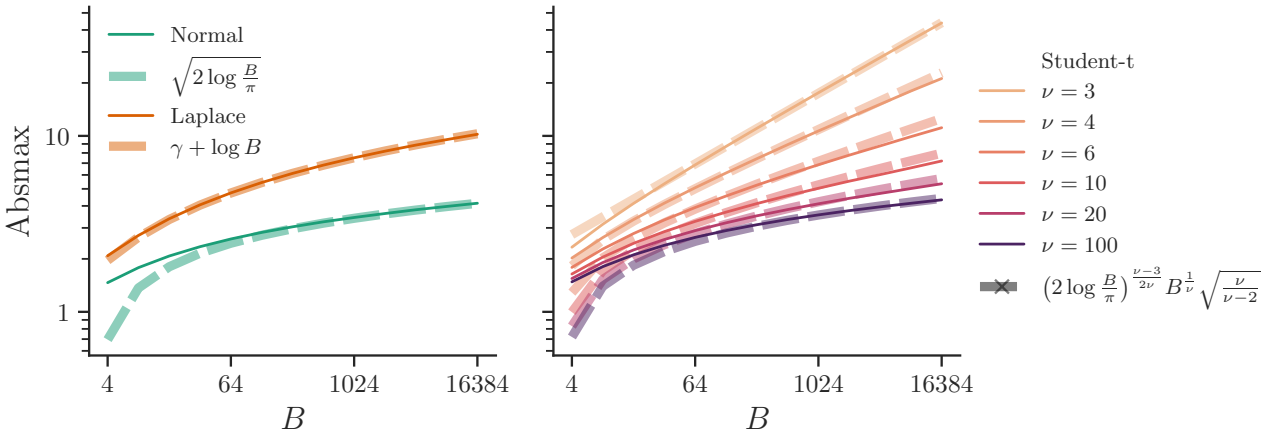

Figure 11. Approximations (*dashed*) to the expected block absmax value for Normal, Laplace and Student-t distributions with scale $s = 1$, versus simulation (*solid*) with $\frac{2^{20}}{B}$ samples. *(Left)* Normal and Laplace distributions. The fit for Normal at small $B \leq 8$ is poor, but typical block sizes are larger than this. *(Right)* Student-t for various degree-of-freedom $\nu \geq 3$, showing good fit across a range of sizes, converging to the Normal approximation as $\nu \to \infty$.

B. Cube root density quantisers

B.1. Recipe

1. Compute parameters of the target distribution $\mathcal{D}$.
    - For RMS scaling, set RMS $= 1$ and use Table 1 to calculate s.
    - For Absmax scaling, set $\mathbb{E}\left[\max_{i \in [1..B]} \lvert\theta_i\rvert\right] = 1$ and use Table 1 to calculate s.
2. Compute parameters of $\mathcal{D}'$, which has pdf $\mathrm{p}^{\mathcal{D}'} \propto \sqrt[3]{\mathrm{p}^{\mathcal{D}}}$ from Table 1.
3. Use the inverse cdf to select quantisation codepoints with density given by $\mathcal{D}'$.

Code examples are given in Appendix F.

B.2. The cube root rule

The cube root rule states that, under some assumptions, the optimal quantiser for distribution $\mathcal{D}$ should have a codepoint density proportional to the cube root of the pdf of $\mathcal{D}$. This is contrasted with *quantile quantisation* (Gersho & Gray, 1991; Dettmers et al., 2022b), which attempts to distribute quantised values evenly, where the density is proportional to the pdf directly. See Figure 13 for an illustration.

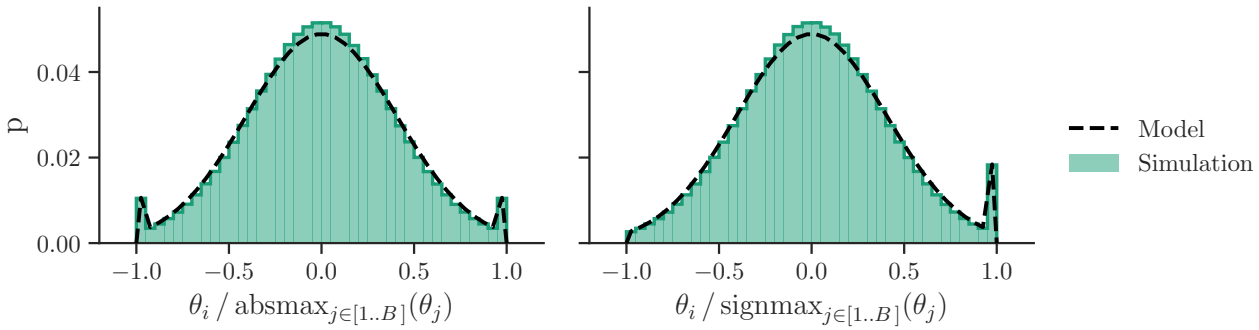

Figure 12. An example of block-scaled Normal data, $B = 64$, using *(left)* absmax and *(right)* signmax scaling: an empirical histogram from sampled data (filled colour) and our mixture model (dashed), using the approximate maximum from Table 1. The empirical marginal distribution is a good fit to our mixture of ± 1 (signmax $+1$) maximum and truncated-Normal non-maxima.

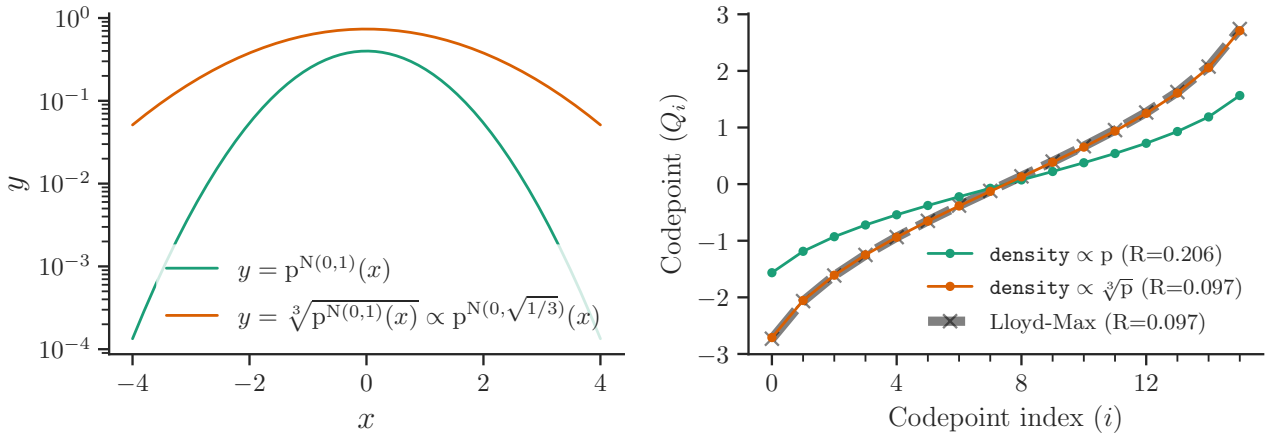

Figure 13. An example of the cube root density rule. *Left:* The density of a standard Normal and $\sqrt[3]{}$ of that density, which is a scaled normal pdf. *Right:* The quantisation curves and error for 4-bit formats derived from the cube root rule, a naive "proportional rule" and a Lloyd-Max quantiser trained on standard Normal samples, showing good match between cube root density and Lloyd-Max.

Derivation For a sketch derivation of the cube root rule (Panter & Dite, 1951), consider a piecewise-uniform probability distribution $\{p_i\}$ and a piecewise-uniform quantiser with n_i codepoints in section i. Then for a single piece of width w, the RMS error is

$$E_i = 2\,n_i \cdot p_i \cdot \int_0^{\frac{w}{2n_i}} \frac{x^2}{w}\,\mathrm{d}x = \frac{p_i \cdot w^2}{12\,n_i^2}.$$

So, with a constraint on number of codepoints i.e., $\sum_i n_i = 2^b$, we use the Lagrange multiplier λ to optimise

$$E' = \sum_i \frac{p_i \cdot w^2}{12\,n_i^2} + \lambda \cdot \left(\sum_i n_i - 2^b\right).$$

This gives the gradients

$$\frac{\mathrm{d}E'}{\mathrm{d}n_i} = -\frac{p_i \cdot w^2}{6\,n_i^3} + \lambda,$$

therefore with $\frac{\mathrm{d}E'}{\mathrm{d}n_i}\big|_{n_i=n_i^\star} = 0$, we see that $n_i^\star \propto \sqrt[3]{p_i}$.

B.3. Parameters of $\mathcal{D}'$

In this section, we derive the rules for s' and ν' for the Normal, Laplace and Student-t distributions given in Table 1. For all of these distributions, there is a distribution of the same family, but with different parameters such that the new distribution's pdf is proportional to the cube root of the original pdf.

Normal For a Normal distribution $N(0, s^2)$,

$$\mathrm{p}(x|s) = \frac{1}{\sqrt{2\,\pi \cdot s^2}} \cdot e^{-\frac{x^2}{2\,s^2}}.$$

If we set $\mathrm{p}(x|s') \propto \sqrt[3]{\mathrm{p}(x|s)}$, we see that for some constant C

$$\frac{1}{\sqrt[6]{2\,\pi \cdot s^2}} \cdot e^{-\frac{x^2}{6\,s^2}} = \frac{C}{\sqrt{2\,\pi \cdot s'^2}} \cdot e^{-\frac{x^2}{2\,s'^2}},$$

therefore $s' = \sqrt{3}\,s$.

Laplace For a Laplace distribution,

$$\mathrm{p}(x|s) = \frac{1}{2\,s} \cdot e^{-\frac{|x|}{s}}.$$

If we set $\mathrm{p}(x|s') \propto \sqrt[3]{\mathrm{p}(x|s)}$, we see that for some constant C

$$\frac{1}{\sqrt[3]{2\,s}} \cdot e^{-\frac{|x|}{3\,s}} = \frac{C}{2\,s'} \cdot e^{-\frac{|x|}{s'}},$$

therefore $s' = 3\,s$.

Student-t For a Student-t distribution,

$$\mathrm{p}(x \,|\, \nu, s) = \frac{1}{s \cdot \sqrt{\nu} \cdot \mathrm{B}(\frac{1}{2}, \frac{\nu}{2})} \cdot \left(1 + \frac{x^2}{s^2 \cdot \nu}\right)^{-\frac{\nu+1}{2}}.$$

If we set $\mathrm{p}(x \,|\, \nu', s') \propto \sqrt[3]{\mathrm{p}(x \,|\, \nu, s)}$, we see that for some constant C

$$\frac{1}{\sqrt[6]{s^2 \cdot \nu \cdot \mathrm{B}(\frac{1}{2}, \frac{\nu}{2})^2}} \cdot \left(1 + \frac{x^2}{s^2 \cdot \nu}\right)^{-\frac{\nu+1}{6}} = \frac{C}{s' \cdot \sqrt{\nu'} \cdot \mathrm{B}(\frac{1}{2}, \frac{\nu}{2})} \cdot \left(1 + \frac{x^2}{s'^2 \cdot \nu'}\right)^{-\frac{\nu'+1}{2}},$$

therefore $\nu' = \frac{\nu-2}{3}$ and $s' = \sqrt{\frac{\nu}{\nu'}} \cdot s$.

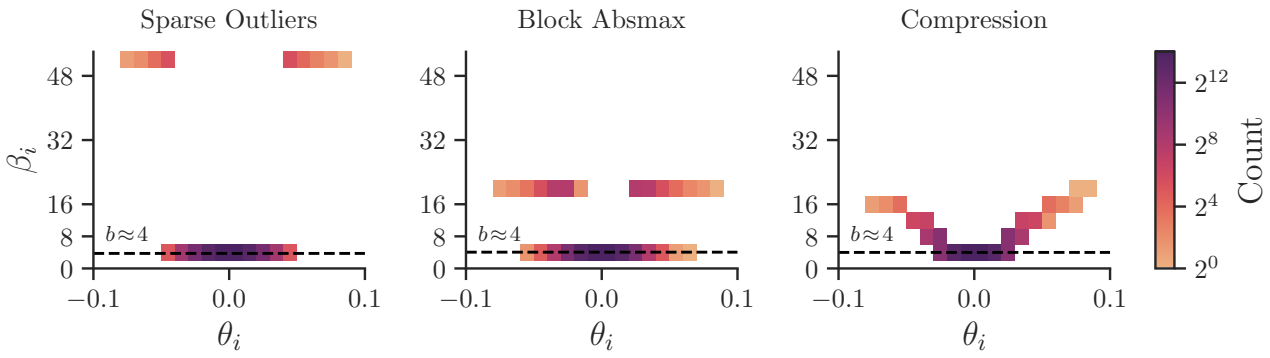

Figure 14. A 2D histogram of bits β_i used to encode parameter i from the first MLP down-projection from Llama 3.1 8B, illustrating how different schemes achieve variable-length encoding. *(Left)* sparse outliers create a distinct step between regular values and the 0.1% largest absolute values, which require $32{+}16$ bits for the position and value respectively. *(Center)* block absmax with a `bfloat16` scale can be seen as using $16{+}4$ bits to represent the block maximum and ≈ 4 for everything else. The histogram has an overlap, since the maximum is per-block not global, but fewer bits are required to store its location. *(Right)* lossless compression on a uniform grid with $\beta_i = -\log_2 p_i$, where p_i is the proportion of parameters assigned to that quantisation bucket.

C. Entropy coding quantisers

C.1. Recipe

1. Choose a resolution for the grid, δ, so that the quantisation codepoints are $\{\delta \cdot k \mid k \in \mathbb{N}\}$.
2. Either compute the density of values mapped to each codepoint analytically, or via samples.
3. Build an entropy code from this distribution, e.g. using Huffman coding.

To reach a target b, this procedure can be wrapped in a search to find an appropriate δ.

C.2. The uniform density rule

The previous method constrained the total number of codepoints, which is appropriate for an uncompressed data stream. If the quantiser is followed by an optimal lossless compressor, we should instead use an entropy constraint:

$$H = -\sum_i n_i \cdot \frac{p_i}{n_i} \log \frac{p_i}{n_i} = b$$

This gives the optimisation objective

$$E'' = \sum_i \frac{p_i \cdot w^2}{12\, n_i^2} + \lambda \cdot \left(\sum_i p_i \log \frac{p_i}{n_i} + b \right),$$

and gradients

$$\frac{\mathrm{d}E''}{\mathrm{d}n_i} = -\frac{p_i \cdot w^2}{6\, n_i^3} - \lambda \cdot \frac{p_i}{n_i},$$

and when $\frac{\mathrm{d}E''}{\mathrm{d}n_i}\big|_{n_i = n_i^\star} = 0$, p_i cancels and $n_i^\star = \mathrm{const}$.

Somewhat surprisingly, the RMS-optimal quantiser when followed by a perfect lossless compressor is a uniform grid (lattice), where the tradeoff between b and E is made by varying the resolution of the grid.

D. Analysis Details (Simulated Data)

For our analysis on simulated data, we draw data iid from Normal, Laplace or Student-t distributions and measure the quantisation error. Since the scale of a distribution is easily absorbed in the formats we consider, our primary evaluation

metric is the ratio of RMS error to data RMS,

$$R \coloneqq \sqrt{\left(\sum_i [E]_i^2\right) / \left(\sum_i \sum_{j \in [1..B]} \theta_{B \cdot i + j}^2\right)},$$

where i is a block index. We often report $R \cdot 2^b$ for legibility when b varies across an experiment, as R tends to scale as 2^{-b}.

Unless noted, we sample $|\theta| = 2^{24}$ scalar values for each experiment, and use `float32` compute precision throughout. For compression results, we use a sampling-based method to calculate the model $\mathrm{p}^{\mathcal{Q}}$ with a fresh set of samples from the target distribution, and use $+1$ smoothing of the counts (within the training sample range) to avoid zeros.

D.1. Results

Additional observations To validate the cube root rule, we test a generalised quantiser with pdf exponent α, where $\alpha = \frac{1}{3}$ matches the cube root rule in Figure 20 and $\alpha = 1$ is quantile quantisation, finding that the cube root setting indeed performs best and similarly to Lloyd-Max. For block scaled formats, we must choose an appropriate block size. Smaller blocks have lower error from a tighter block range but incur greater space overhead from storing the block scale. For a fair comparison, smaller blocks must use a narrower element bit width. See Figure 19, showing that $B = 128$ is a good choice for these distributions. The figure also validates our default choice of `bfloat16` over `E8M0`.

Question	Figures
How to choose between compression & scaling schemes?	15
How to choose an element format?	16, 17
How to choose a scale format?	18, 19
How to choose block size?	19
Does the cube-root rule work?	20
Is moment matching sufficient for choosing quantiser scale?	21
How well does practical compression approach the optimal limit?	22

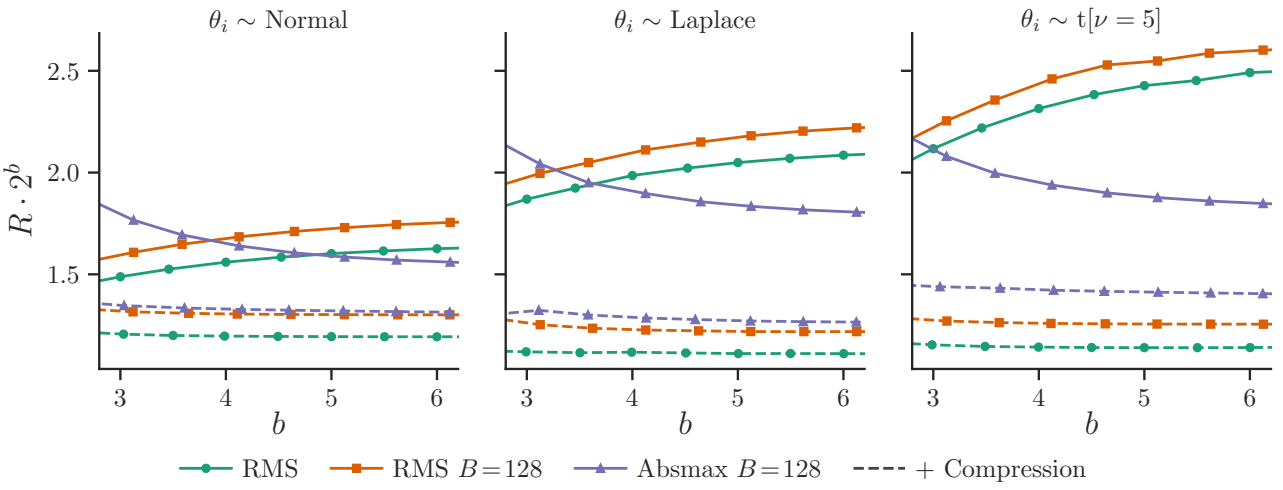

Figure 15. The error/size tradeoff for different data distributions (column) and optimal quantisers (hue). Surprisingly, block absmax quantisers can outperform tensor RMS formats for iid data, even though there is no inherent block structure. However this situation is reversed when adding optimal compression, implying that block absmax quantisers exploit some form of variable-length coding.

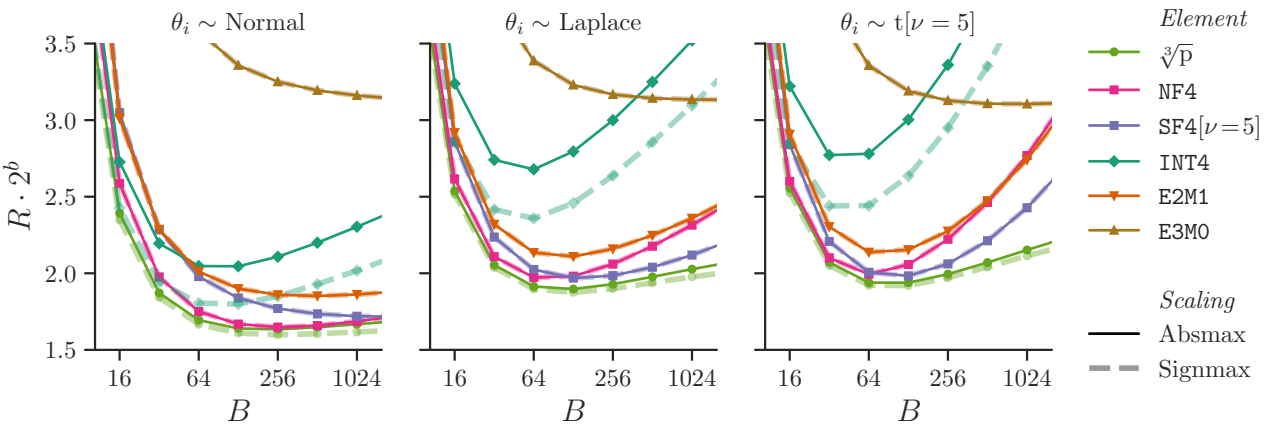

Figure 16. The performance of optimal and extant 4-bit element formats as block size B varies. Note that the total bit width varies with B, but is consistent across element formats. We see that the $\sqrt[3]{\mathrm{p}}$ formats are marginally better than NF4 and SF4, which don't optimise for RMS error. Of the floating-point and integer formats, E2M1 is generally the best. Signmax quantisation improves INT4 considerably and makes it competitive for Normal data, although performance is still poor for heavier-tailed distributions.

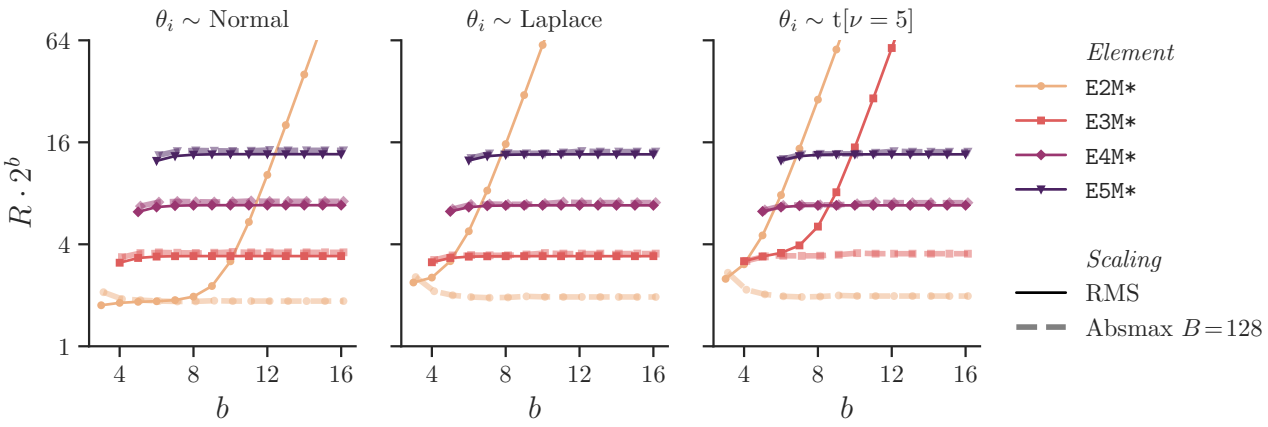

Figure 17. Floating-point element format performance as the total bit width b varies. In general, the optimal number of exponent bits doesn't depend on total bit width. The exception is for RMS scaling, where low-exponent formats eventually stop improving with more mantissa bits (so $R \cdot 2^b$ starts increasing). This is due to the error in quantising the distribution tails, which lie outside the format's range — increasing the number of mantissa bits has negligible effect on range, so this source of error eventually dominates. Note that for this plot, it was important that the `bfloat16` scaling factor used round-away rather than round-to-nearest, to avoid range issues from rounding the scale down.

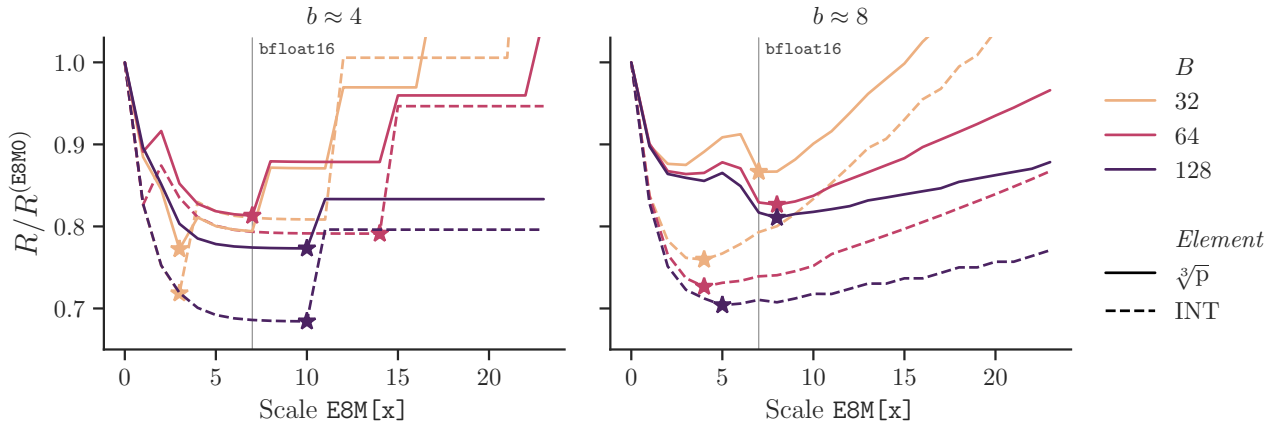

Figure 18. The performance advantage of scale mantissa bits, keeping average bit width b approximately constant by varying the element bit width, for Student-t ($\nu = 5$) data. Both $\sqrt[3]{\mathrm{p}}$ and integer formats benefit from 4-10 scale exponent bits, and integers show greater benefit. Note the jumps in the $b \approx 4$ plot are due to a discrete number of codepoints in the element format.

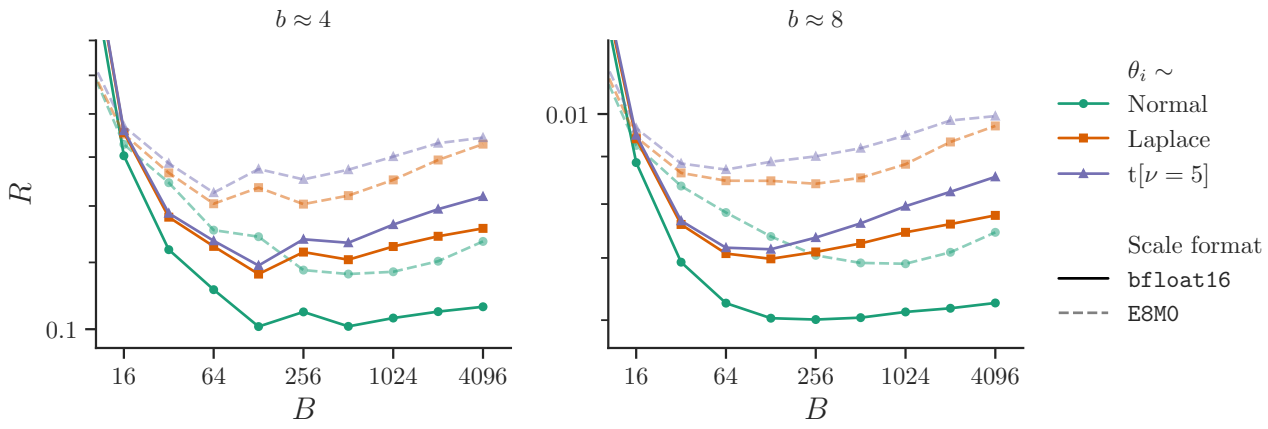

Figure 19. Absmax-scaled format error versus block size B, for different approximate bit widths, data distributions and scale format. As B decreases, the element bit width is reduced to keep $b = b^{\text{element}} + \frac{b^{\text{scale}}}{B}$ approximately constant. `bfloat16` (or `E8M7`) outperforms the mantissa-less `E8M0` format. The optimum for Normal data is generally slightly to the right of that for heavy-tailed Laplace and Student-t distributions, generally in the range 64–256.

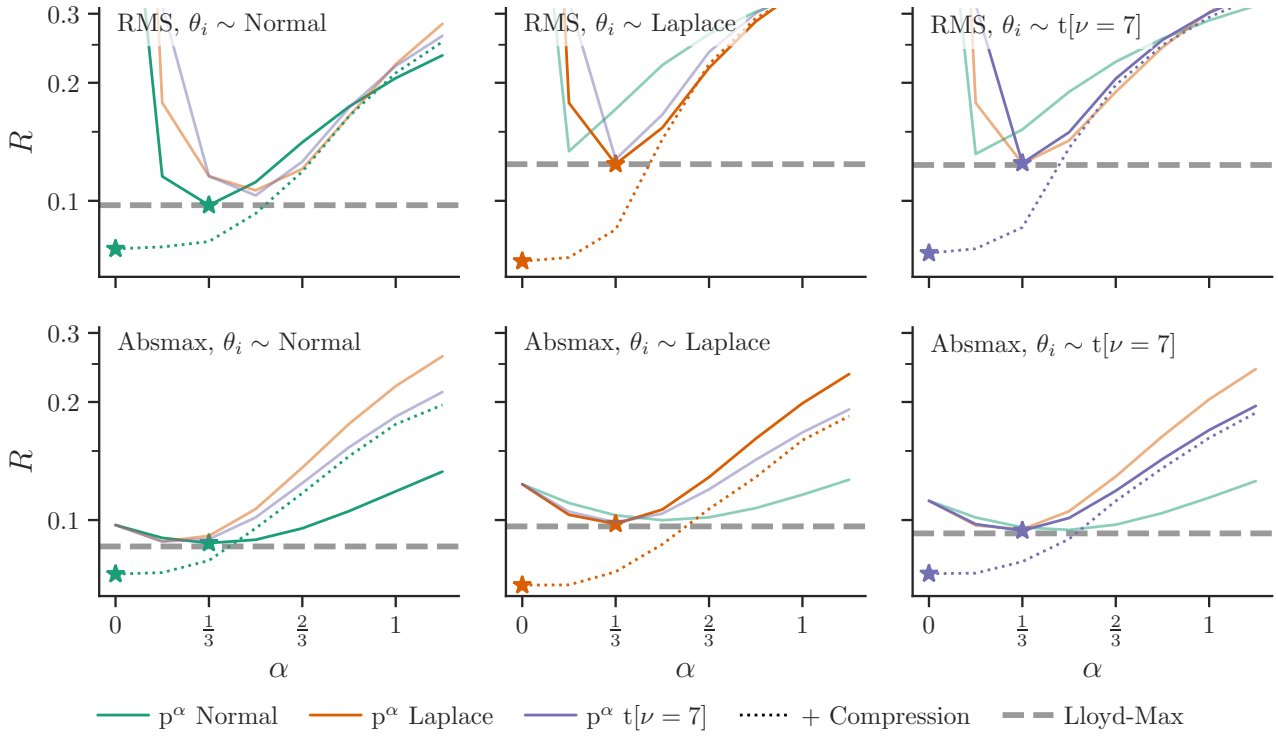

Figure 20. Validation of optimal 4-bit $\sqrt[3]{\text{p}}$ quantisers via simulation. We generalise the $\sqrt[3]{\text{p}}$ rule to a p$^\alpha$ rule for various α (horizontal axis) and try quantisers derived from different distributions (hue) using moment-matching. *(Top)* RMS scaling. *(Bottom)* Absmax scaling, $B = 64$. We find that the best quantiser is consistently the matching $\sqrt[3]{\text{p}}$ ($\alpha = \frac{1}{3}$), which performs comparably to a Lloyd-Max trained quantiser. We also show the curve for a compressed quantiser with $b \approx 4$, which has optimum at $\alpha = 0$, i.e. a uniform grid that is independent of the pdf.

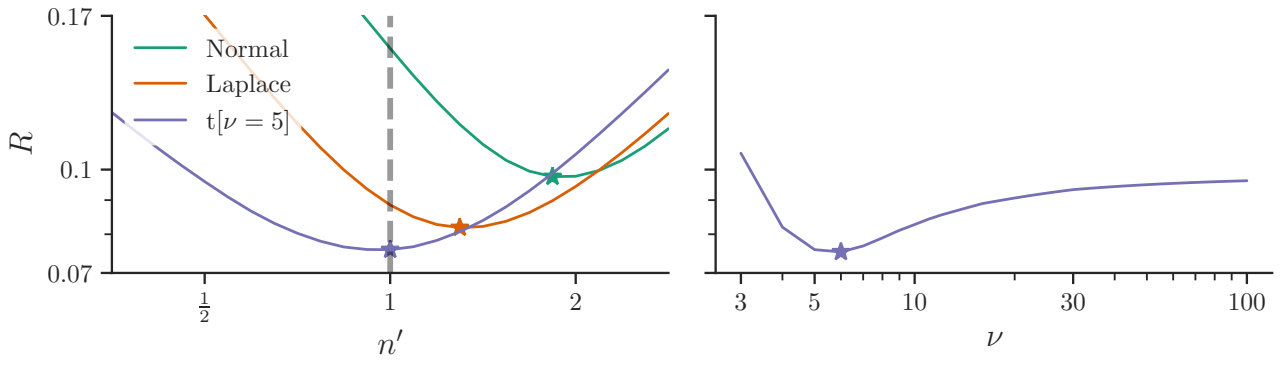

Figure 21. Search to find the best 5-bit quantiser parameters after RMS scaling of data generated from a Student-t ($\nu = 5$) distribution. *(Left)* search over the scale applied to the quantiser, such that $\tilde{\theta}_i = n' \cdot \texttt{dequantise}\left(\texttt{quantise}\left(\frac{\theta_i}{n'}\right)\right)$. Note that each quantiser (Normal, Laplace, etc) is optimal for data of their matching distribution, with RMS = 1. For the correct Student-t quantiser, RMS moment matching ($n' = 1$) works well, but moment matching performance is suboptimal for mismatched quantisers. *(Right)* search to find the correct Student-t quantiser ν. For each ν, we search for the scale n' that minimises R.

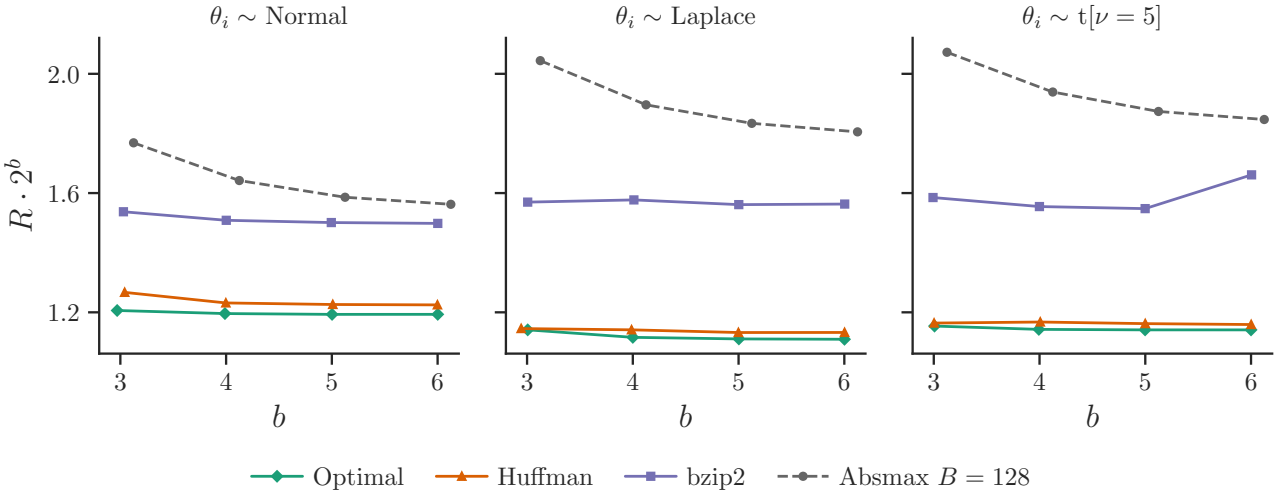

Figure 22. The performance of practical compressors using RMS scaling and $\sqrt[3]{\text{p}}$ element formats, compared with the theoretical limit, over $|\theta| = 2^{20}$ samples. An elementwise Huffman code (Huffman, 1952) using `dahuffman` (Lippens, 2017) performs close to optimal. Bzip2 doesn't reach the same compression ratio, however it still outperforms an uncompressed block format.

E. Experimental Details

Our language modelling experiments use the WikiText-103 (Merity et al., 2017) combined validation and test sets. For each sequence in the dataset, we generate full sequence (teacher-forcing) logits from the reference model and test model for evaluation using cross entropy and top-k KL divergence, which is described below. Hyperparameters are given in Table 2. The 11 models evaluated are: Llama 3.1 8B, Llama 3.2 1B, Llama 3.2 3B, Phi 4 (14B), Qwen 2.5 {0.5B, 1.5B, 3B, 7B} and Gemma 3 {1B, 4B, 12B} (Dubey et al., 2024; Yang et al., 2024; Kamath et al., 2025; Abdin et al., 2024). Where multiple variants exist, we use the bare pretrained model.

The division of parameters into tensors follows the Huggingface `transformers` (Wolf et al., 2020) checkpoints, which differ slightly between models. For example, Phi-4 contains a single "stacked" projection matrix for query-key-value, while the other models tested store them separately.

Our k-means results use a custom implementation which iterates until the proportion of cluster assignments that change drops below 10^{-4} and uses `k-means++` (Arthur & Vassilvitskii, 2007) initialisation for RMS-scaled data and uniform $(-1, 1)$ initialisation for absmax-scaled data — settings which we found to be robust during early testing.

Table 2. Experimental settings.

Hyperparameter	Value
Sequence length	4096
KL top-k	128
Evaluation tokens	$\approx 5 \cdot 10^5$
Fisher estimation tokens	$4 \cdot 10^6$
Reference weight format	`bfloat16`
`transformers` version	`4.51.3`
Scale search range	$[2^{-2}, 2^{-1.75}, \ldots, 2^2]$ (17 steps)
Student-t ν search range	`logspace`$(\log_2 3, \log_2 100, \text{steps=12}, \text{base=2})$

Top-k KL divergence Our comparison metric is top-k KL divergence, defined for a single pair of logits that specify $\mathrm{p}_\theta(y_i \mid x)$ and $\mathrm{p}_{\tilde{\theta}}(y_i \mid x)$ as

$$\mathrm{D_{KL}}^{\text{top-}k}(\mathrm{p}_\theta, \mathrm{p}_{\tilde{\theta}}) \coloneqq \left(\sum_{y \in \text{argtop}k(p)} p_y \cdot \log \frac{p_y}{q_y} \right) + p^{\text{tail}} \cdot \log \frac{p^{\text{tail}}}{q^{\text{tail}}}$$

$$\text{where } p_y \coloneqq \mathrm{p}_\theta(y \mid x) \text{ and } q_y \coloneqq \mathrm{p}_{\tilde{\theta}}(y \mid x) \, ,$$

$$p^{\text{tail}} \coloneqq \sum_{y \notin \text{argtop}k(p)} p_y \, ,$$

$$q^{\text{tail}} \coloneqq \sum_{y \notin \text{argtop}k(p)} q_y \, .$$

Note that the top-k always applies to the reference model, never the target model. The *tail* term is required to ensure that the KL divergence is ≥ 0. The logic is equivalent to creating a modified distribution where the non-top-k classes are collapsed into a single output class, followed by regular KL divergence over $k + 1$ classes.

We use top-k KL divergence rather than full KL divergence because the vocabulary size of language models (typically $> 10^5$) makes it prohibitive to store a dataset of reference logits. Top-k KL divergence stores $2 \cdot k$ scalar values per token, an index and a log-probability, versus a log-probability per vocabulary term for full KL divergence.

Fisher estimation To estimate the diagonal of the Fisher information defined in Equation (4) for the reference model, we sample text sequences from the WikiText-103 training set. For each sequence, we generate logits from the model, sample a single output token per position in the sequence from the predicted distribution and backpropagate the cross entropy loss of the sampled token to get the gradient with respect to activations. We replace the calculation of weight gradients with a

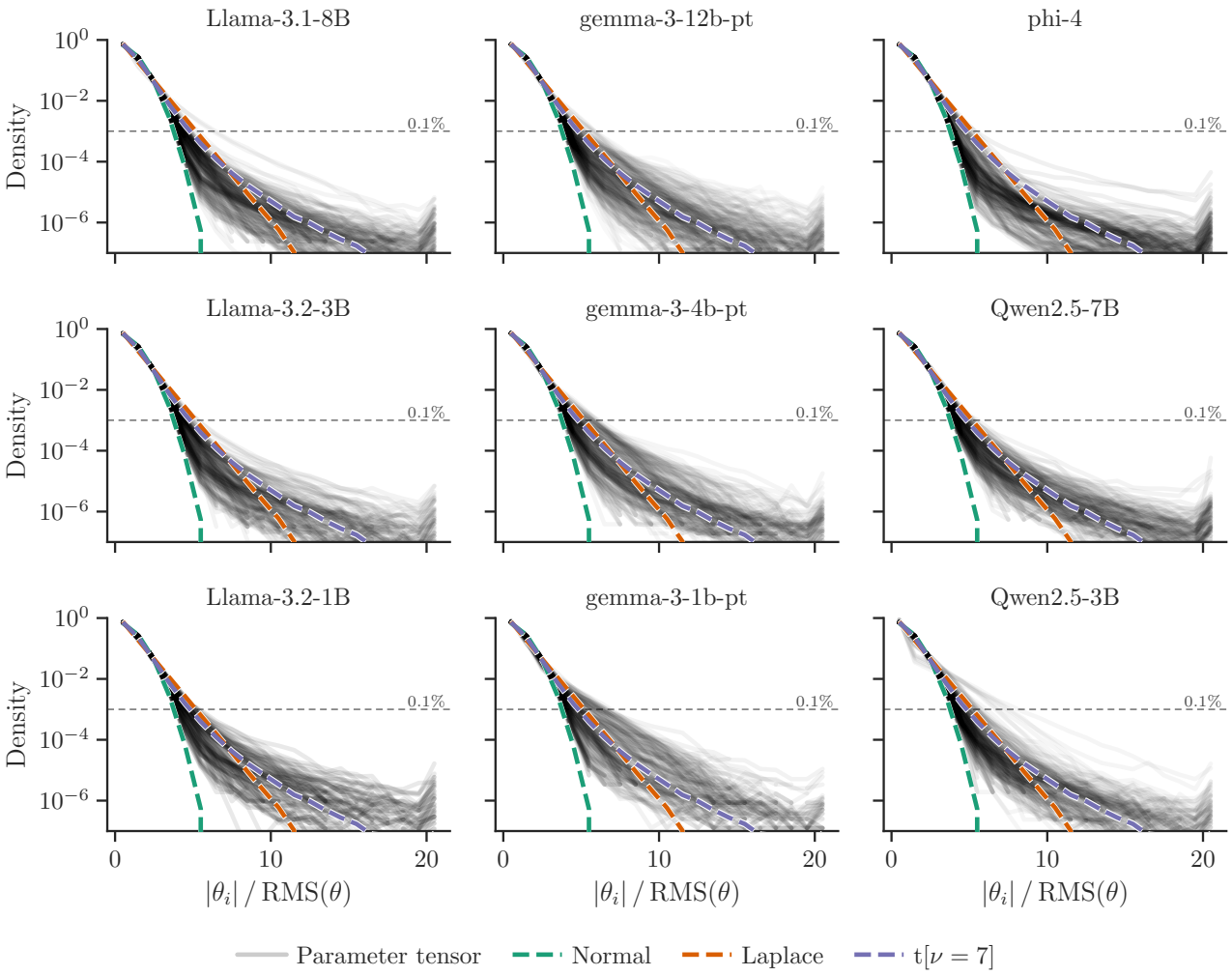

Figure 23. A histogram of absolute parameter values for various models. Each line corresponds to a parameter tensor in the model. As we care about tails not scales (the overall scale of a tensor is easily absorbed into a scaling factor), we divide each parameter value by the RMS of the tensor. We note that different models show the same general trends: heavy tails that seem closest to a Student-t distribution in shape, with some variability across tensors in the model.

custom version, which squares the gradients before accumulation (see Appendix F.3). That is, we calculate

$$F_{ii} \approx \frac{1}{M \cdot L} \sum_{m \in [1..M]} \sum_{p \in [1..L]} \left(\nabla_{\theta_i} \log \mathrm{p}_\theta(\hat{y}_p^{(m)} \mid x_{<p}^{(m)}) \right)^2 \quad \text{where } \hat{y}_p^{(m)} \sim \mathrm{p}_\theta(y \mid x_{<p}^{(m)}). \tag{8}$$

over $M = 1024$ sequences of length $L = 4096$. Note that we use a sampled target label rather than the ground truth from WikiText in order to be closer to estimating the Fisher rather than empirical Fisher, a difference explored by Kunstner et al. (2019), at the cost of increased variance of our estimator. Despite this effort, since we use "teacher forcing" of inputs in an autoregressive setting, the method remains somewhat empirical.

Since this method accumulates the diagonal Fisher, it stores $|\theta|$ additional values, a similar amount of memory to training with SGD. Although the parameters may be represented in `bfloat16`, it is important to accumulate the Fisher statistics in a format with more mantissa bits, as `bfloat16` updates will be swamped after $\mathrm{O}(2^8)$ steps. To support Fisher estimation with limited accelerator memory, we implement a 2-stage accumulator that accumulates 64 steps in `bfloat16` on device, then accumulates these batched updates in `float32` on the host CPU.

Moment matching baselines For RMS scaling with $\sqrt[3]{\mathrm{p}}$ formats, the moment matching baseline sets the RMS of the quantiser to match that of the data. For standard formats it scales such that data RMS $= 1$ in the case of `E2M*` and $\frac{2^{b-1}-1}{\sqrt{3}}$ (to match the RMS of a uniform distribution) in the case of INT. With Absmax scaling, the moment matching baseline sets the scale such that the minimum of the positive and negative range of the quantiser matches that of the normalised data, i.e. to cover $(-1, 1)$.

E.1. Results

Question	Figures
How to choose between (compression, scaling, outlier) schemes?	1, 6, 24
How do random rotations help?	25
Does variable bit allocation help?	26, 27
How to choose an element format	28, 29
How to choose a scale format?	30
How to choose block size?	30
Signmax, Asymmetric or Symmetric scaling?	31
Moment matching or scale search?	32

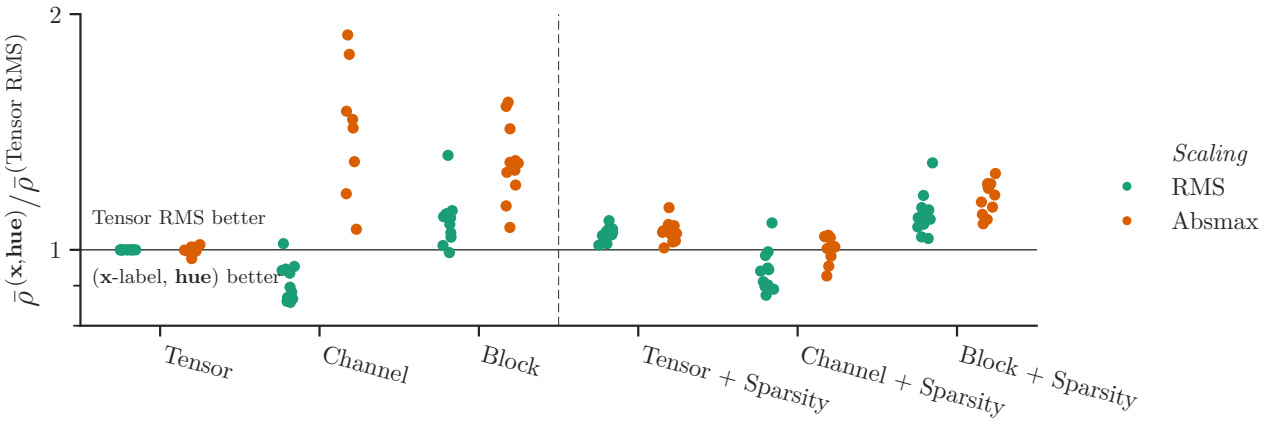

Figure 24. The average change in scaled KL for various scaling schemes and sparsity, *when combined with optimal lossless compression*. Note that each point is the scaled KL for a model, averaged over bit widths, and divided by the tensor RMS baseline. In the presence of lossless compression, there is no benefit to block scaling or separating sparse outliers, consistent with our claim that they exploit the same variable-length encoding benefit offered by compression. The only scaling mode that outperforms simple tensor RMS scaling when combined with compression is channel RMS scaling, which exploits structure in the tensor data.

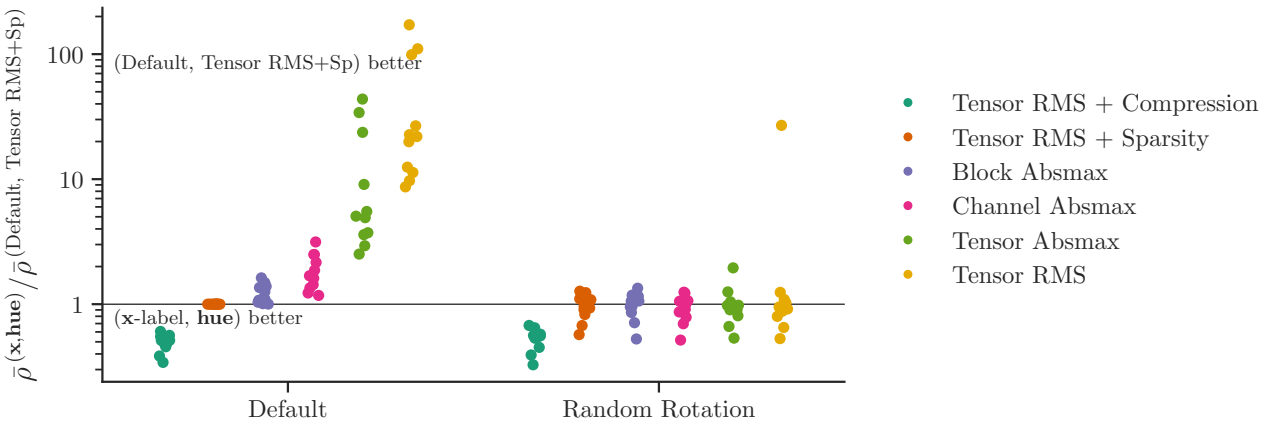

Figure 25. An evaluation of random rotations, where the rotations V and W are applied before quantising the rows and columns respectively of a 2D parameter tensor, i.e. $\tilde{\theta} = V^\top \texttt{dequantise}(\texttt{quantise}(V\,\theta\,W))W^\top$. Since we expect rotated parameters to be roughly normally distributed, we use the $\sqrt[3]{\text{p}}$ normal quantiser, optionally with a block scaling scheme, sparse outliers or compression. Our results show that random rotations are useful for fixed-length schemes such as tensor scaling without sparse outliers, but unnecessary for schemes that employ variable-length coding. This is what we'd expect: rotations transform heavy-tailed marginal distributions, where fixed-length quantisation performs much worse than variable-length quantisation (Figure 15 *(right)*), towards the Normal distribution, for which fixed-length quantisation performs better (Figure 15 *(left)*).

Note that the outlier point for Tensor RMS scaling with rotation corresponds to the Phi-4 model, which is likely an experimental issue — for sake of memory, we skip rotations where the dimension is too large (e.g. embedding vocabulary dimension), and with the large hidden size of Phi-4, our code also skipped rotating the output dimension of the stacked MLP up-and-gate projection.

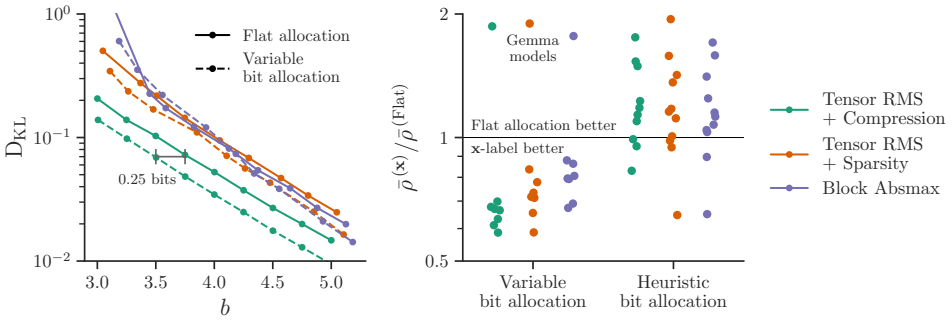

Figure 26. The performance of the Fisher-based variable bit allocation scheme of Equation (2). *(Left)* the tradeoff curve for Llama 3.1 8B, showing a general shift to the left, although some settings for absmax scaling are degraded. *(Right)* the average scaled KL of different bit allocation schemes compared with flat allocation, for all models. For 7 of the 11 models tested, the variable allocation scheme improves the average KL across bit widths, versus flat allocation. The exceptions are for all 3 Gemma models (see Figure 6) and for one format with Qwen 2.5 3B. See Figure 27 for a cross-domain result and an explanation of *heuristic bit allocation*.

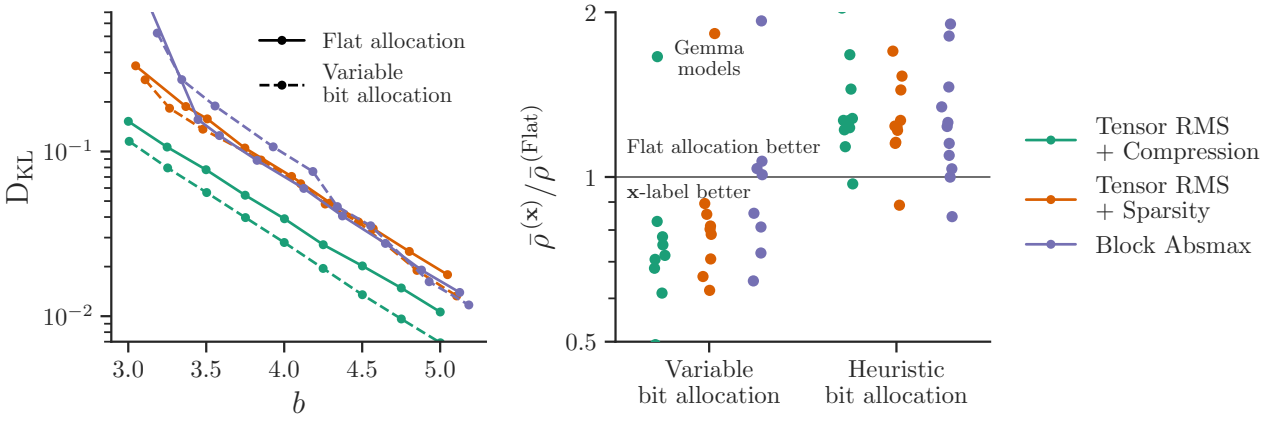

Figure 27. The performance of the Fisher-based variable bit allocation scheme of Equation (2) for `codeparrot/github-code` (Tunstall et al., 2022), when the Fisher information was calculated over WikiText, a substantially different dataset. *(Left)* the tradeoff curve for Llama 3.1 8B, showing a general shift to the left, although some settings for absmax scaling are degraded. *(Right)* the average scaled KL of different bit allocation schemes compared against flat allocation, for all models. Much of the in-domain improvement is retained, indicating that the Fisher information can generalise across datasets. Note that the *heuristic bit allocation* scheme allocates +2 bits to all parameters in the first 2 and last 2 transformer layers, and to embedding and final projection weights; this performs poorly.

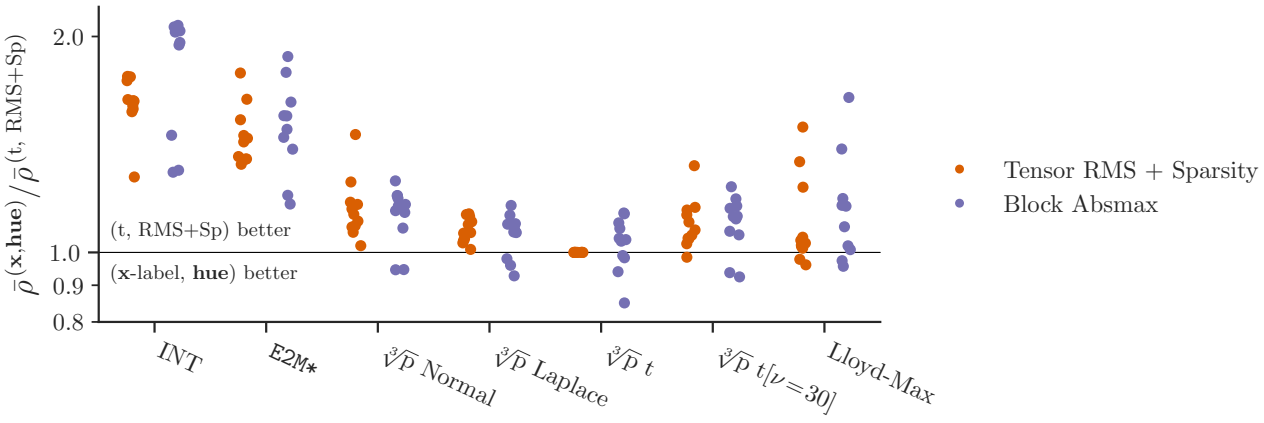

Figure 28. A comparison of different element formats, each point the best setting for a given model, over {moment matching/search/Fisher-weighted search, symmetric/asymmetric variant}, compared with Student-t with RMS scaling and sparse outliers. No setting consistently beats this baseline across models; surprisingly, this includes Lloyd-Max with Fisher weighting.

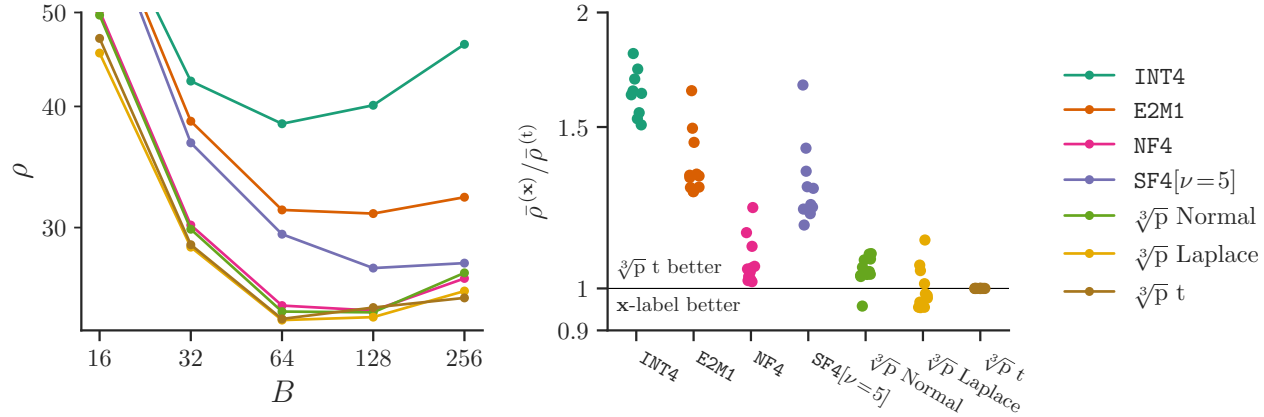

Figure 29. A comparison of $\sqrt[3]{\mathrm{p}}$ against extant formats with block absmax scaling, 4-bit elements and `bfloat16` scale, i.e. $b = 4 + \frac{16}{B}$. *(Left)* Llama 3.1 8B performance as B varies. We see that the $\sqrt[3]{\mathrm{p}}$ formats and `NF4` perform similarly. Note that $\sqrt[3]{\mathrm{p}}$ Normal is different from `NF4`, since $\sqrt[3]{\mathrm{p}}$ formats optimise for RMS not incompressibility and use a model of the block-maximum, meaning that the curve depends on B. *(Right)* average performance across different models, where each point gives the average ρ across block size, divided by the performance of (model, $\sqrt[3]{\mathrm{p}}$ Student-t). We see that $\sqrt[3]{\mathrm{p}}$ Laplace and Student-t perform best in general, and there is little to choose between $\sqrt[3]{\mathrm{p}}$ Normal and `NF4`. Surprisingly, `SF4` is worse, at odds with the findings of Dotzel et al. (2024). One possible explanation for the difference is our use of a `bfloat16` scale, which provides a tighter bound on the block maximum, compared with an `E8M0` exponent.

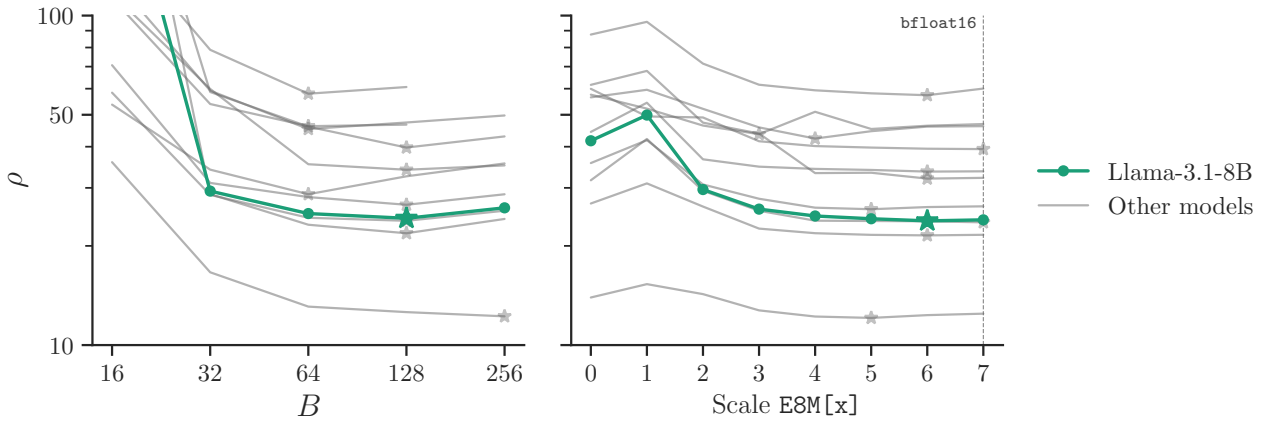

Figure 30. Hyperparameter sweep for block absmax formats using the Student-t element quantiser and $b \approx 4$. *(Left)* block size sweep, showing that almost all models agree on $B = 128$, given a `bfloat16` scale, consistent with our simulations in Figure 19. *(Right)* scale mantissa bits sweep with round-away, showing that most models benefit from 4-6 scale mantissa bits, given $B = 128$, consistent with Figure 18. For both sweeps, a fair comparison is made by adjusting the element width to account for the different scale overhead. For example, for $B = 64$ with a `bfloat16` scale, the element bit width is set as close to $4 - \frac{16}{64}$ as possible.

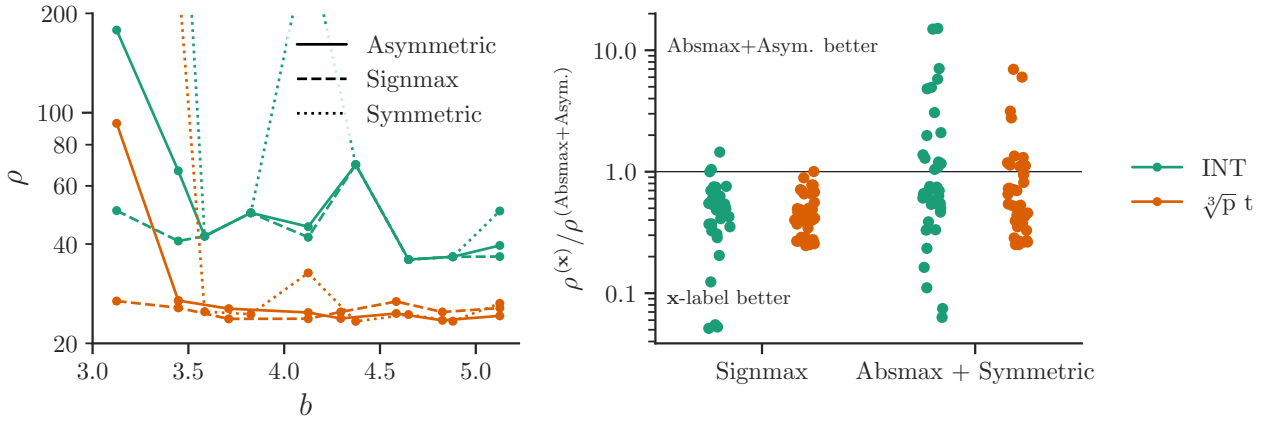

Figure 31. A comparison of scaling variants (see Figure 2) for INT and $\sqrt[3]{\mathrm{p}}$ Student-t element formats, using block scaling with $B = 128$. *(Left)* the tradeoff curve for Llama 3.1 8B, showing that signmax outperforms regular absmax scaling at small b. Symmetric scaling, which does not include a representation for 0 does not perform consistently for this model. *(Right)* scaled KL over all models, relative to the absmax + asymmetric variant. The improvement from signmax is consistent. The symmetric format is sometimes better and sometimes worse than asymmetric.

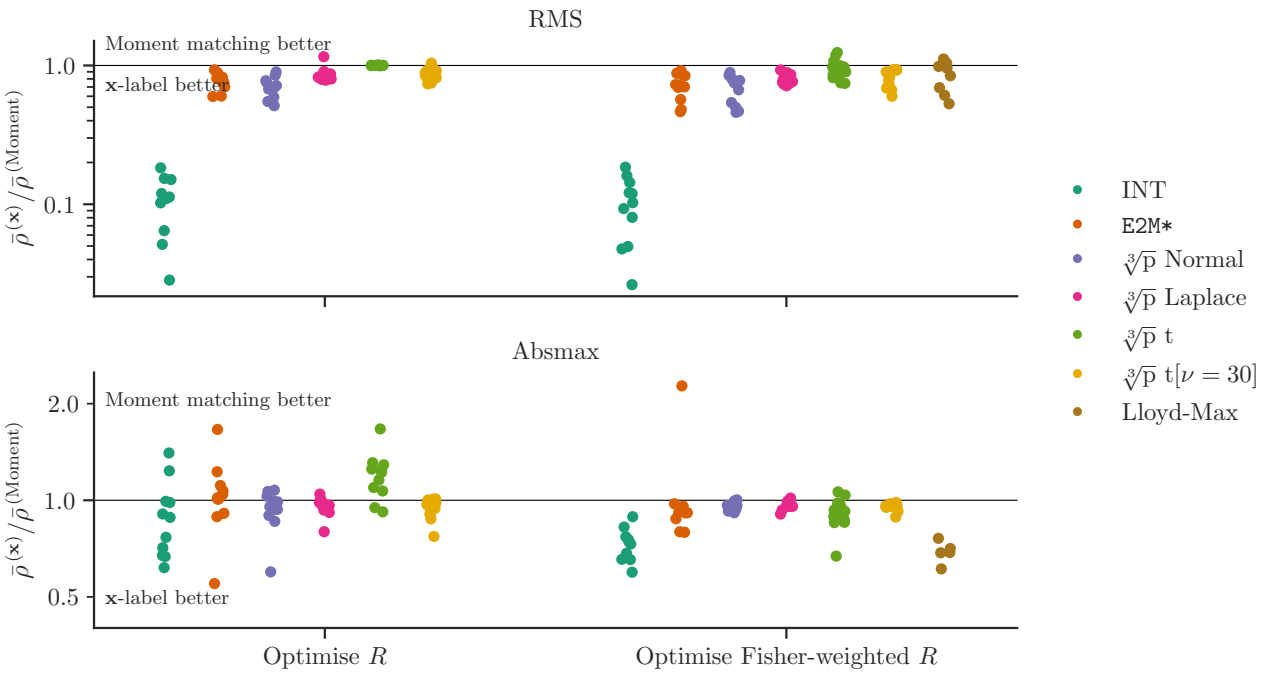

Figure 32. Moment matching vs search over quantiser scale to optimise R and Fisher-weighted search. Each point corresponds to the average ρ over bit width for one of the 11 models tested. The results suggest that search is helpful for RMS scaling, but not reliable for absmax scaling, although Fisher weighting seems to help here. Note that all formats for Qwen2.5-3B perform very badly using Fisher-weighted search, with a ratio $\frac{\rho^{(\text{Fisher})}}{\rho^{(\text{Moment})}} > 2$. Only one of these results is visible in-range.

F. Code examples

This section provides illustrative implementations that compute optimal quantisation curves for RMS and block absmax Normal, Laplace and Student-t distributions as well as code to estimate the diagonal of the Fisher information matrix. For the full implementation used for our experiments, please see https://github.com/graphcore-research/optimal-weight-formats.

F.1. Cube root density (RMS scaling)

Illustrative implementations of 4-bit cube root density quantisation of different distributions (symmetric variant).

Normal

```
b = 4
p = torch.linspace(0, 1, 2**b + 2)
Q = torch.tensor(scipy.stats.norm.ppf(p[1:-1], scale=sqrt(3)))

def quantise(x): return torch.bucketize(x, (Q[1:] + Q[:-1]) / 2)
def dequantise(i): return Q[i]
```

Note the `scale` for the ppf (inverse cdf) is set to $\sqrt{3}$, according to the rule from Table 1.

Laplace (RMS $= 1$):

```
b = 4
p = torch.linspace(0, 1, 2**b + 2)
Q = torch.tensor(scipy.stats.laplace.ppf(p[1:-1], scale=3/sqrt(2)))
```

Student-t ($\nu =$ `df`, RMS $= 1$):

```
b, df = 4, 7
p = torch.linspace(0, 1, 2**b + 2)
Q = torch.tensor(scipy.stats.t.ppf(p[1:-1], (df-2)/3, scale=sqrt(3)))
```

F.2. Cube root density (block absmax scaling)

Illustrative implementations of 4-bit cube root density quantisation of different distributions, scaled by their block absmax (symmetric variant).

Normal

```
b, block_size = 4, 64
p = torch.linspace(0, 1, 2**b)
scale = sqrt(3 / (2 * log(block_size/pi)))
Q = torch.tensor(scipy.stats.truncnorm.ppf(p, -1/scale, 1/scale, scale=scale))
```

Note the `scale` for the inverse cdf is $\frac{s'}{\mathbb{E}[\max_i \theta_i]}$ from Table 1.

Laplace

```
def trunclaplace_ppf(q, x0, x1, scale):
    c0, c1 = scipy.stats.laplace.cdf([x0*scale, x1*scale], scale=scale)
    return scipy.stats.laplace.ppf(c0 + (c1-c0)*q, scale=scale)

b, block_size = 4, 64
p = torch.linspace(0, 1, 2**b)
scale = 3 / (0.57721566 + log(block_size))
Q = torch.tensor(trunclaplace_ppf(p, -1/scale, 1/scale, scale=scale))
```

Student-t ($\nu =$ `df`)

```
def trunct_ppf(q, df, x0, x1, scale):
    c0, c1 = scipy.stats.t.cdf([x0*scale, x1*scale], df, scale=scale)
```

```
    return scipy.stats.t.ppf(c0 + (c1-c0)*q, df, scale=scale)

b, block_size, df = 4, 64, 7
p = torch.linspace(0, 1, 2**b)
scale = (2*log(block_size/pi))**((3-df)/(2*df)) * block_size**(-1/df) * sqrt(3)
Q = torch.tensor(trunct_ppf(p, (df-2)/3, -1/scale, 1/scale, scale=scale))
```

F.3. Fisher estimation

Illustrative code for wrapping a `torch.nn.Linear` layer with logic to compute the sum of squared gradients in order to estimate the diagonal of the Fisher information matrix.

```
class FisherWrappedLinear(torch.nn.Module):
    def __init__(self, m: torch.nn.Linear):
        super().__init__()
        self.m = m
        self.gW2 = torch.zeros_like(self.m.weight, dtype=torch.float32)

    def forward(self, x):
        y = self.m(x)
        y.requires_grad_(True).register_hook(
            lambda gy: self.gW2.addmm_(
                gy.detach().flatten(0, -2).float().square().T,
                x.detach().flatten(0, -2).float().square(),
            ) is None or None
        )
        return y
```