QuEST: Training Accurate LLMs over Highly-Compressed Weights and Activations

Andrei Panferov, Jiale Chen, Soroush Tabesh, Roberto L. Castro, Mahdi Nikdan & Dan Alistarh*
Institute of Science and Technology Austria
{firstname.lastname}@ist.ac.at

Abstract

One main approach to reducing the massive costs of large language models (LLMs) is the use of quantized or sparse representations for training or deployment. While post-training compression methods are very popular, obtaining even more accurate compressed models by *directly training* over such representations, i.e., *Quantization-Aware Training (QAT)*, is still largely open. In this paper, we advance this state-of-the-art for QAT via a new method called QuEST, which is Pareto-competitive with FP16, that is, it provides better accuracy at lower model size, while training models with weights and activations in 4-bits or less. Moreover, QuEST allows stable training with 1-bit weights and activations, and is compatible with *weight sparsity*. Experiments on Llama-type architectures show that QuEST induces new, stable scaling laws across the entire range of hardware-supported compressed representations. Moreover, we provide GPU kernel support showing that the models produced by QuEST can be efficiently executed on current hardware.

1 Introduction

One approach to reducing the computational costs of AI has been *quantization-aware training (QAT)* (Rastegari et al., 2016; Jacob et al., 2018)— where models are trained from scratch with low-precision weights and activations on the forward pass, but with a full-precision backward pass—offering the potential for superior accuracy-vs-compression trade-offs, as gradient optimization can correct compression errors. Despite promising results for weight-only quantization (Wang et al., 2023; Kaushal et al., 2024), it is currently not known whether QAT can produce accurate LLMs with extremely low-bitwidth/sparse weights and activations. Here, the key metric is the *Pareto-optimal frontier*, i.e., the minimal representation size (or inference cost) for the model to achieve a certain accuracy under a fixed data or training budget. Recently, Kumar et al. (2024) identified 8-bit precision as Pareto-optimal for QAT methods on LLMs.

Contribution. We present QuEST, a new QAT method that brings the Pareto-optimal frontier to around 4-bit weights and activations and enables stable training at 1-bit precision for both operands. As shown in Figure 1, when data and compute are scaled proportionally to model size, QuEST can train models with 4-bit weights and activations that have superior accuracy relative to BF16 models almost 4x in size. We achieve this by re-thinking two key aspects of QAT methods: 1) the "forward" step, in which continuous-to-discrete tensor distribution fitting is performed on the forward pass, and 2) the "backward" step, in which gradient estimation is performed over the discrete representation.

Starting from this algorithm, we focus on the following question: assuming that training computation is not a limiting factor, what is the "optimal" precision in terms of accuracy-vs-model-size? To address this, we implement QuEST in Pytorch (Paszke et al., 2019) and train Llama-family models (Dubey et al., 2024) of up to 800M parameters on up to 80B tokens from the standard C4 dataset (Raffel et al., 2019), across precisions from INT1 to INT8. Results show that QuEST provides stable and accurate convergence across model sizes and precisions down to 1-bit weights and activations. This induces new scaling laws, which we study across model sizes in the large-data (100 tokens/parameter) regime. QuEST leads INT4 weights and activations to be Pareto-optimal in terms of accuracy at a given model size and inference cost, suggesting that the limits of low-

*Red Hat AI

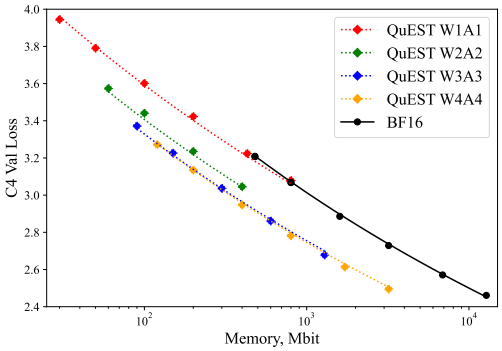

Figure 1: QuEST scaling law for training 30M to 800M Llama-family models on C4, with quantized weights and activations from 1 to 4 bits, in the 100 tokens/parameter regime. QuEST allows for stable training at 1-bit weights and activations (W1A1), and the QuEST W4A4 model is Pareto-dominant relative to BF16, with lower loss at lower size.

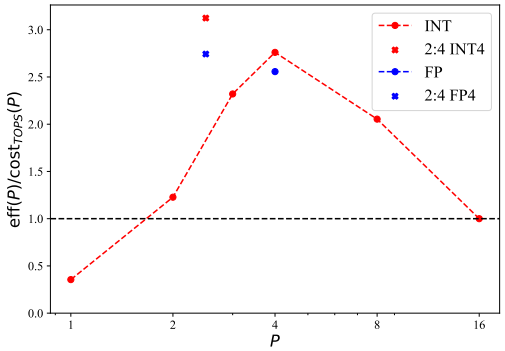

Figure 2: Illustration of the efficiency factors $\text{eff}(P)/\text{cost}(P)$, arising from our analysis, for different numerical precisions P and formats (INT, FP, INT+sparse). Higher is better. 2:4 INT4 appears to have the highest efficiency among hardware-supported formats. Here, $\text{cost}(P)$ the inverse max throughput relative to BF16.

precision training are lower than previously thought. In addition, we provide GPU kernels showing that models produced by QuEST can be run efficiently on commodity hardware.

2 QuEST

Background. Broadly, QAT considers the problem of finding a quantized projection over a *standard-precision tensor* $\mathbf{x}$, representing part of the weights or activations, minimizing output error. For symmetric uniform quantization, the projection onto *the quantized tensor* $\hat{\mathbf{x}}$ is defined as: $\hat{\mathbf{x}} = \alpha \cdot \left\lfloor \frac{\text{clip}(\mathbf{x},\alpha)}{\alpha} \right\rceil$, where the clip function performs a clamping operation over the value distribution for all values above the clipping parameter α, which also acts as a scaling factor, normalizing values to $\mathbf{x}$ to $[-1, 1]$, and the function $\lfloor \cdot \rceil$ rounds each value to its nearest quantization point, defined as a uniform grid whose granularity depends on the number of available bits b. Most QAT methods propose to "learn" the factor α, for instance, via gradient-based optimization. For example, QAT methods usually keep a standard-precision version $\mathbf{w}$ of the weights; the STE gradient is computed *over the quantized weights* $\widehat{\mathbf{w}}$, and then added to the full-precision accumulator, possibly also updating the clipping factor α.

Step 1: Distribution Fitting. While optimizing the quantization grid to best fit the underlying tensor is a core idea across all quantization methods, PTQ methods traditionally use more complex and computationally heavy approaches (Dettmers et al., 2024; Malinovskii et al., 2024). In contrast, QAT methods rely on backpropagation through the scaling factor for error-correction (Esser et al., 2019; Bhalgat et al., 2020) while performing re-fitting. To avoid backpropagation errors impacting the forward pass, we do not use backpropagation for distribution fitting. Instead, we start from the empirical observation that the distribution of weights and activations during LLM training is sub-Gaussian but with long tails (Dettmers et al., 2022; 2023).

Specifically, we choose to optimize the grid to explicitly fit a Gaussian distribution with the same parametrization as the empirical distribution of the underlying tensor $\mathbf{x}$. Concretely, we use *root mean square (RMS)* normalization to first align the empirical distribution of $\mathbf{x}$ with a $\mathcal{N}(0, 1)$ Gaussian distribution. We then perform the projection operation with the scale α^* chosen to minimize the L_2 error resulting from projecting $\mathcal{N}(0, 1)$. Formally:

$$\widehat{\mathbf{x}} = \alpha^* \cdot \text{RMS}(\mathbf{x}) \cdot \left\lfloor \frac{\text{clip}\left(\mathbf{x}/\text{RMS}(\mathbf{x}), \alpha^*\right)}{\alpha^*} \right\rceil := \text{proj}_{\alpha^*}(\mathbf{x}), \text{ where}$$

$$\alpha^* := \arg\min_{\alpha \in \mathbb{R}} \mathbb{E}_{\xi \sim \mathcal{N}(0,1)} \left\| \xi - \alpha \cdot \left\lfloor \frac{\text{clip}(\xi, \alpha)}{\alpha} \right\rceil \right\|_2^2$$

is the MSE-optimal scaling factor, estimated once. If $\mathbf{x}$ were Gaussian-distributed, this would produce an MSE-optimal projection.

Yet, the natural distribution of tensor values may not be Gaussian, especially given the emergence of outlier values (Dettmers et al., 2022; Nrusimha et al., 2024). To mitigate this, we add a Hadamard Transform (HT) step *before* Gaussian Fitting. Thus, our forward pass projection becomes:

$$\hat{\mathbf{x}}_h = \text{proj}_{\alpha^*} \text{HT}(\mathbf{x}). \tag{1}$$

In other words, we transform the target tensor via multiplication with a Hadamard matrix of appropriate shape, applied along the matrix-multiplication dimension, and then project it to an MSE-optimal grid in the Hadamard domain. Here, we leverage 1) the fact that, roughly, multiplication of a matrix with the Hadamard Transform leads the weight distribution to better match a Gaussiant (Ailon & Chazelle, 2009; Suresh et al., 2017); 2) the existence of fast Hadamard multiplication kernels (Tri Dao), and 3) the fact that the HT is orthogonal, so it can be easily inverted. While this HT effect has been utilized in PTQ (Tseng et al., 2024; Ashkboos et al., 2024; Malinovskii et al., 2024), we believe we are the first to harness it for QAT.

Step 2: Trust Gradient Estimation. Next, we focus on the backward pass. For simplicity, we first describe the variant without the Hadamard Transform step and then integrate this component. First, assume that $\widehat{\mathbf{x}} = \text{proj}_{\alpha^*}(\mathbf{x})$. Since the projection operation $\lfloor x \rceil$, is not differentiable w.r.t. x, we need a robust way to estimate our gradient. Expressed as an operator, STE can be written as $\frac{\partial}{\partial \mathbf{x}} \approx \frac{\partial}{\partial \lfloor \mathbf{x} \rceil}$ during the backward pass, allowing gradients to propagate through the network, but can lead to large errors due to components with large quantization error.

Specifically, the factor α^*, chosen to minimize the weight fitting error, acts as a natural scale for how far off their real value the majority of quantized values can be: for values below the scaling factor, this error is not larger than $T = \frac{\alpha^*}{2^b - 1}$, the half-width of a quantization interval.

To mitigate the impact components with large quantization error, we choose to *not trust* the gradient estimations for weights with large errors $S_{\text{large}} = \{k : |\hat{\mathbf{x}} - \mathbf{x}|_k > T\}$. Choosing $T = \frac{\alpha^*}{2^b - 1}$ and masking gradients for those elements we obtain the gradient operator:

$$\frac{\partial}{\partial \mathbf{x}} \approx \mathbf{I}_{|\hat{\mathbf{x}} - \mathbf{x}| \leq T} \odot \frac{\partial}{\partial \hat{\mathbf{x}}} := M_{\alpha^*}(\mathbf{x}; \hat{\mathbf{x}}) \odot \frac{\partial}{\partial \hat{\mathbf{x}}},$$

where $\mathbf{I}_{|\hat{\mathbf{x}} - \mathbf{x}| \leq T}$ is the standard indicator operator. We will refer to M_{α^*} as the "trust mask"; this gradient estimation operator will be called the **trust estimator**.

Trust Estimators for the Hadamard Projection. We now interface the trust estimator with the Hadamard Transform (HT) and its inverse (HT^{-1}) to obtain the following forward scheme: $\mathbf{x}_h = \text{HT}(\mathbf{x})$ and $\hat{\mathbf{x}}_h = \text{proj}_{\alpha^*} \mathbf{x}_h$. Then, the natural approach is to perform trust estimation directly in the Hadamard domain, where quantization takes place:

$$\frac{\partial}{\partial \mathbf{x}} \approx \text{HT}^{-1} \left(M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h) \odot \frac{\partial}{\partial \hat{\mathbf{x}}_h} \right).$$

In other words, after deriving the trust mask w.r.t. distribution fitting in the Hadamard domain, we apply the resulting mask $M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h)$ onto the gradient w.r.t. quantized weights *in the Hadamard domain*.

The Problem of Sparse Gradients. Notice that, in the absence of the HT or regularization effects (e.g., weight decay), the "untrusted" weights S_{large} would receive no gradient and may be permanently removed from optimization. Yet, the addition of the HT means that the trust mask is *no*

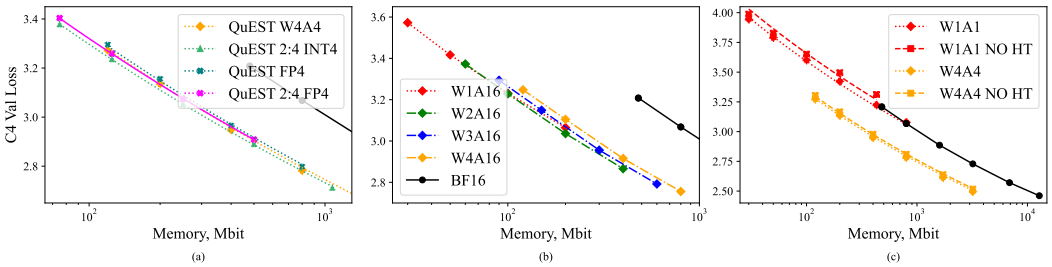

Figure 3: Additional scaling laws induced by QuEST: **(a, left)** compares INT, FP, and INT+sparse formats at 4-bit precision, **(b, middle)** shows the scaling laws for weight-only quantization, where 2-bit appears to be Pareto-dominant, while **(c, right)** shows that trust estimation is stable even without Hadamard normalization.

longer binary in the "standard" domain, allowing for gradient flow towards *all model weights*. We validated this effect empirically by observing that the HT reduced the final cardinality of the "untrusted" weights set S_{large} by ≈ 4x, aligning it with the number of values we would expect to be outside the "trust set" at every step, for weights from a normal distribution. This is investigated in more depth in Appendix A.2.

Implementation. In practice, we use identical Hadamard Transforms along the matrix-multiplication dimension for both the weights $\mathbf{w}$ and the activations $\mathbf{x}$. Since the Hadamard Transform is unitary, the quantized matrix multiplication output $\mathbf{y} = \hat{\mathbf{x}}\hat{\mathbf{w}}^T$ is aligned with the full precision output $\mathbf{x}\mathbf{w}^T$ it approximates. We provide algorithmic description of the forward and the backward pass in Appendix A.1.

Sparsity. QuEST can also be extended to sparsity. Then, the trust estimator will mask out sparsified elements with absolute value above the trust mask; specifically, this covers the majority of sparsified elements, except for the small elements within $\left[-\frac{\alpha^*}{2^b-1}, +\frac{\alpha^*}{2^b-1}\right]$. In practice, we still keep the whole weight matrix in full precision during training. On the forward pass, we first sparsify and then quantize. On the backward pass, we apply the trust mask as usual.

3 Experimental validation

We describe the experimental setup, as well as the model and the training procedure hyperparameters in Appendix B.1.

Scaling laws. Following Frantar et al. (2023), we modify the standard scaling laws Hoffmann et al. (2022) assuming that the training precision P only affects the parameter count N as a multiplicative factor $\text{eff}(P)$, which, for a given quantization method, depends only on the training precision. (For $\text{eff}(16) = 1.0$, we recover standard precision; the fitting process is described in Appendix C.1.)

$$L(N, D, P) = \frac{A}{(N \cdot \text{eff}(P))^{\alpha}} + \frac{B}{D^{\beta}} + E. \tag{2}$$

Results. The overall results were presented in Figure 1, illustrating loss vs. model size. First, we observe that, remarkably, QuEST provides stable training down to 1-bit weights and activations, across model sizes, following a stable scaling law. Second, examining the Pareto frontier, we observe that 4-bit precision is slightly superior to 3-bit, and consistently outperforms all higher precisions. Overall, these results show that QuEST can lead to stable scaling laws, which consistently improve upon prior results (Kumar et al., 2024), moving the Pareto-optimal line to around 4-bit.

3.1 Finding the "optimal" precision

Runtime Cost Estimate. We now focus on the "overtraining" (OT) regime, where the training compute is less relevant, and is only bounded by factors such as the available amount of filtered training data, which allows us to ignore the data term D.

In practice, *runtime* constraints on LLM deployment often include memory, single-user generation speed, highly-parallelized througpht and power consumption constraints. One thing to notice is that, for a large number of applications, all of those quantities are proportional to the model size N for every precision P. As such, we can generalize those constraints and formalize them in a single form of $R \leq R^{max}$ where $R = \gamma N \cdot \text{cost}(P)$ for some constant γ and precision-dependent factor. For example, for model size R_{mem} measured in Mb, we get $\text{cost}_{mem}(P) = \frac{P}{16}$ and $\gamma_{mem} = \frac{Mb}{10^6}$. Then, the problem of minimizing loss while staying within a certain runtime constraint can be re-written as:

$$\min_{N,P} L_{OT}(N,P) = \frac{A}{\left(\frac{R}{\gamma} \cdot \frac{\text{eff}(P)}{\text{cost}(P)}\right)^{\alpha}} + E \text{ s.t. } R \leq R^{max}.$$

To optimally solve this problem, one can choose

$$P^* = \arg\max_P \left\lfloor \frac{\text{eff}(P)}{\text{cost}(P)} \right\rceil$$

independently of the limit R^{max}, and then set $N = R^{max} / \left(\gamma \cdot \text{eff}(P^*)\right)$. Thus, cost-effectiveness $\frac{\text{eff}(P)}{\text{cost}(P)}$ becomes the key quantity for choosing the "optimal" pre-training precision in the OT regime. Here, $\text{eff}(P)$ can be found by fitting Equation 2 for a certain quantization scheme. In Figure 2, we present those quantities for $\text{cost}(P)$ chosen to encode the speedup of quantized matrix multiplication relative to BF16.

3.2 Extensions to different formats

The FP4 Format. We can use the same framework to compare the "effective parameter count" for INT, INT + sparse, and the lower-precision FP format supported by NVIDIA Blackwell (NVIDIA, 2024). QuEST can be extended to this data type by replacing the $\lfloor \cdot \rceil$ rounding operation with rounding to the FP4 grid $\lfloor \cdot \rceil_{\text{FP4}}$ scaled to fit the same $[-1, 1]$ interval. The optimal scaling factor α^*_{FP4} would be defined by simply replacing $\lfloor \cdot \rceil$ with $\lfloor \cdot \rceil_{\text{FP4}}$ in the original definition. We choose the trust factor T for $M_{\alpha^*}(\mathbf{x}; \hat{\mathbf{x}}) = \mathbf{I}_{|\hat{\mathbf{x}} - \mathbf{x}| \leq T}$ as the largest half-interval of the FP4 grid.

Quantization plus sparsity. Figure 3(a) also illustrates the scaling law induced by the 50% sparse + INT4 of NVIDIA Ampere (Abdelkhalik et al., 2022), while Figure 2 (red cross) shows its peak throughput parameter efficiency relative to INT and FP. With QuEST, this format can provide better scaling than any purely quantized representation we tested. (While this format is known as 2:4 sparsity, for INT4 + 2:4 it requires a 4:8 mask with some additional constraints.)

FP sparsity. Additionally, Figure 3(a) illustrates the scaling law induced by the 50% sparse + FP4 of NVIDIA Blackwell (Nvidia). Figure 2 (blue cross) shows its peak throughput parameter efficiency relative to INT, FP and 2:4 INT4. This format can provide better scaling than FP4, but slightly inferior to both INT and 2:4 INT4.

4 Discussion and future work

We introduced QuEST, a new QAT method that achieves stable LLM training of in extremely low precision (down to 1-bit) weights and activations. Our results demonstrate that, if data and compute are appropriately scaled, 4-bit models can outperform standard-precision baselines in terms of accuracy and inference cost, suggesting that the fundamental limits of low-precision QAT are much lower than previously thought. Further, our analysis provides new insights into the relationship between training precision and model efficiency, suggesting that low-precision may be a good target for large-scale training runs in the overtrained regime. Third, we have shown that our approach can lead to inference speedups.

Several promising directions emerge for future work. First, while we demonstrated QuEST's effectiveness up to 800M parameters, its scaling behavior for much larger models is an interesting direction we plan to pursue in future work. Second, our work focused primarily on decoder-only architectures; extending QuEST to encoder-decoder models and other architectures could broaden its applicability.

REFERENCES

Hamdy Abdelkhalik, Yehia Arafa, Nandakishore Santhi, and Abdel-Hameed Badawy. Demystifying the nvidia ampere architecture through microbenchmarking and instruction-level analysis, 2022. URL https://arxiv.org/abs/2208.11174.

Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009. doi: 10.1137/060673096.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35: 30318–30332, 2022.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus, 2021. URL https://arxiv.org/abs/2104.08758.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017. URL https://arxiv.org/abs/1702.03118.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

Elias Frantar, Carlos Riquelme, Neil Houlsby, Dan Alistarh, and Utku Evci. Scaling laws for sparsely-connected foundation models, 2023. URL https://arxiv.org/abs/2309.08520.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Ayush Kaushal, Tejas Vaidhya, Arnab Kumar Mondal, Tejas Pandey, Aaryan Bhagat, and Irina Rish. Spectra: Surprising effectiveness of pretraining ternary language models at scale. *arXiv preprint arXiv:2407.12327*, 2024.

Tanishq Kumar, Zachary Ankner, Benjamin F Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. Scaling laws for precision. *arXiv preprint arXiv:2411.04330*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Vladimir Malinovskii, Andrei Panferov, Ivan Ilin, Han Guo, Peter Richtárik, and Dan Alistarh. Pushing the limits of large language model quantization via the linearity theorem. *arXiv preprint arXiv:2411.17525*, 2024.

Aniruddha Nrusimha, Mayank Mishra, Naigang Wang, Dan Alistarh, Rameswar Panda, and Yoon Kim. Mitigating the impact of outlier channels for language model quantization with activation regularization, 2024. URL https://arxiv.org/abs/2404.03605.

Nvidia. Nvidia rtx blackwell gpu architecture. URL https://images.nvidia.com/aem-dam/Solutions/geforce/blackwell/nvidia-rtx-blackwell-gpu-architecture.pdf.

NVIDIA. Nvidia blackwell architecture technical brief. "https://resources.nvidia.com/en-us-blackwell-architecture", 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and H. Brendan McMahan. Distributed mean estimation with limited communication, 2017. URL https://arxiv.org/abs/1611.00429.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Hungyueh Chiang Tri Dao, Nikos Karampatziakis. Fast hadamard transform in cuda, with a pytorch interface. URL https://github.com/Dao-AILab/fast-hadamard-transform.

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

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

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

A Additional "trust" details

A.1 Algorithmic description

The algorithm 1 describes the forward pass over a linear layer actively quantized with QuEST for a row-major weight representation.

The algorithm 2 describes the backward pass over the same layer using the quantized weight and activations from the forward pass as well as error gradient w.r.t $\mathbf{y}$. We note that, although the backward computation is performed w.r.t. the *quantized* weights and activations, the multiplications and gradient operands are performed in standard 16-bit precision.

Algorithm 1 `QuEST Training Forward`

Input: Input activations $\mathbf{x}$, row-major weight $\mathbf{w}$
$\mathbf{x}_h = \text{HT}(\mathbf{x})$
$\hat{\mathbf{x}}_h = \text{proj}_{\alpha^*}\, \mathbf{x}_h$
$\mathbf{w}_h = \text{HT}(\mathbf{w})$
$\hat{\mathbf{w}}_h = \text{proj}_{\alpha^*}\, \mathbf{w}_h$
$\mathbf{y} = \hat{\mathbf{x}}_h \hat{\mathbf{w}}_h^T$
Return: $\mathbf{y}$, $\hat{\mathbf{x}}_h$, $\hat{\mathbf{w}}_h$, $M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h)$, $M_{\alpha^*}(\mathbf{w}_h; \hat{\mathbf{w}}_h)$

Algorithm 2 `QuEST Training Backward`

Input: $\frac{\partial L}{\partial \mathbf{y}}$, $\hat{\mathbf{x}}_h$, $\hat{\mathbf{w}}_h$, $M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h)$, $M_{\alpha^*}(\mathbf{w}_h; \hat{\mathbf{w}}_h)$
$\frac{\partial L}{\partial \hat{\mathbf{x}}_h} = \frac{\partial L}{\partial \mathbf{y}} \hat{\mathbf{w}}_h$
$\frac{\partial L}{\partial \mathbf{x}} = \text{HT}^{-1}\left(M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h) \odot \frac{\partial L}{\partial \hat{\mathbf{x}}_h}\right)$
$\frac{\partial L}{\partial \hat{\mathbf{w}}_h} = \hat{\mathbf{x}}_h^T \frac{\partial L}{\partial \mathbf{y}}$
$\frac{\partial L}{\partial \mathbf{w}} = \text{HT}^{-1}\left(M_{\alpha^*}(\mathbf{w}_h; \hat{\mathbf{w}}_h) \odot \frac{\partial L}{\partial \hat{\mathbf{w}}_h}\right)$
Return: $\frac{\partial L}{\partial \mathbf{x}}$, $\frac{\partial L}{\partial \mathbf{w}}$

A.2 Trust Mask Analysis

For the purposes of weight trust masks interpretation, we trained a 30M model over 3B tokens (11,444 iterations at bs=512) with QuEST weights and activations quantization to 8-bit with and without the Hadamard Transform (HT). We logged the trust masks every 500 iterations. Figure 4 shows the fraction of masked weights. We can see that adding the HT leads to an $\approx$4x decrease in the amount of masked values, corresponding to the fraction of expected clipped weights for a standard normal distribution. We can also see that without the HT the fraction deviates significantly from the expected fraction under the assumption of weights normality.

Moreover, we looked at the percentage of masked elements at a fixed iteration in the past, that remain masked at a fixed later iteration. We plot these percentages in Figure 5. As we can see, for

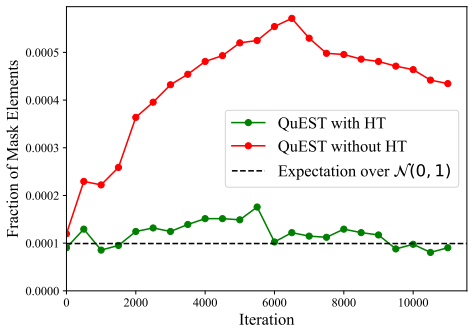

Figure 4: Fraction of weights for which $M_{\alpha^*} = 0$ as a function of number of training iterations for a 30M model trained with QuEST.

the run without the HT, around 69% of masked elements at iteration 6000 (roughly halfway through training) remain masked at iteration 10000 (towards the end of the training). This percentage is more than twice as small for the run with the HT at 30%. This implies that the HT makes masks less persistent, as expected. In addition, we note that weight decay is applied on all weights (including masked ones). Thus, a masked weight will slowly decay until it may "exit" the masked interval, obtaining gradient again.

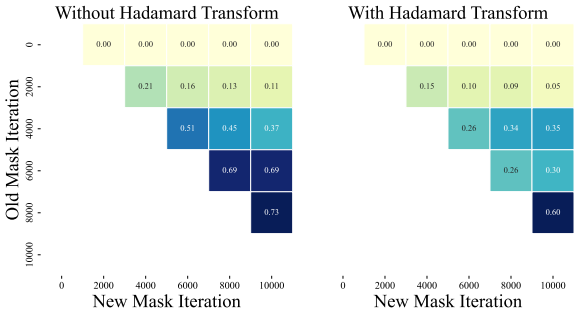

Figure 5: Fraction of masked values retained from an old iteration to a new iteration for a 30M model trained with QuEST W8A8.

A.3 The 1-bit case

In our original trust estimation formulation, we proposed to set the trust factor as half the quantization interval, $T = \frac{\alpha^*}{2^b-1}$. Thus, the trust regions increase exponentially as the bitwidth decreases. In particular, for 1-bit weights and activations, QuEST will suffer from trust regions that extend out of the grid by a whole $\alpha^\star$. To fix this, we reduce the size of the "outermost" trust regions, outside the clipping factor, by a scaling factor s.

To determine the optimal outer trust scaling factor s^*, discussed in Section 2, we conduct a sweep over s, varying the outer size of the outermost trust regions as $T = s \cdot \frac{\alpha^*}{2^b-1}$. The results for 1-bit, shown in Figure 6, indicate that $s^* = 1.30$ for the standard QuEST setup and $s^* = 1.25$ for the setup without the Hadamard Transform (HT), corresponding to exactly a quarter of the quantization interval.

We use this scaling factor for all the 1-bit QuEST runs in this paper (unless stated otherwise). This modification is necessary (and leads to an improvement) only in the extreme 1-bit compression regime.

A.4 Zero-shot evaluation of QuEST models

To assess the effectiveness of QuEST beyond perplexity, we conducted a zero-shot evaluation on the HellaSWAG benchmark (Zellers et al., 2019), which tests commonsense reasoning capabilities. We compared an 800M parameter model trained with QuEST in 4-bit precision against its BF16 counterpart, both of which were trained on 80B tokens.

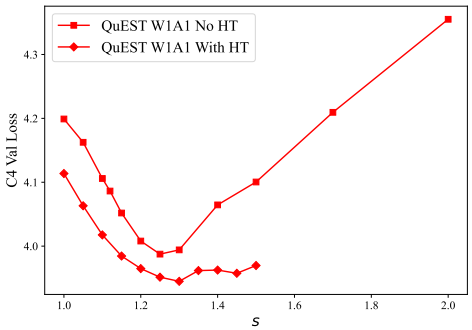

Figure 6: Performance of QuEST as a function of the outer trust scaling factor s for a 30M model pretraining.

Table 1 presents the accuracy results. The near-identical performance between the two models indicates that QuEST's quantization-aware training is essentially lossless, preserving the model's ability to generalize while significantly reducing precision and computational costs.

Model	HellaSWAG Accuracy (%) ↑
BF16 (800M, 80B tokens)	39.52
QuEST 4-bit (800M, 80B tokens)	39.22

Table 1: Zero-shot evaluation on HellaSWAG comparing QuEST 4-bit to its BF16 counterpart. The results are nearly identical, confirming that training with QuEST is lossless.

B Additional information on the experimental Setup

B.1 Implementation details

Models and Hyperparameters. We tested our method on pre-training decoder-only Transformers (Vaswani, 2017) following the Llama architecture (Touvron et al., 2023), in the range of 30, 50, 100, 200, 430 and 800 million non-embedding parameters. We trained all models on tokens from the C4 (Dodge et al., 2021) dataset, tokenized with the Llama 2 tokenizer. We used the AdamW (Loshchilov & Hutter, 2019) optimizer with a cosine learning rate schedule and a 10% warmup period, with gradient clipping (1.0 threshold, decoupled weight decay of 0.1). We identified the learning rate optimally for a 50M FP16 model via a learning-rate sweep. For other models, as standard, we scale the learning rate inverse-proportionally to the number of non-embedding parameters. We reuse the exact learning rates for all QuEST training runs.

Unless stated otherwise, we train every model on a number of tokens equal to 100x its number of "free" parameters, e.g., 10B tokens for a Llama 100M model, regardless of precision. This allows us to explore the data-saturation regime. We aim for comparisons that are iso-size: That is, to match the size / FLOPs of a 100M FP16 Llama model (trained on 10B parameters), we will train a 400M-parameter model with 4-bit weights and activations, using 40B total tokens. This allows us to explore accuracy for fixed model sizes, across compression ratios (see Figure 1). We discuss different D/N regimes in Appendix C.2.

B.2 Model hyper-parameters

For our experiments, we chose to use the Llama 2 (Touvron et al., 2023) model as the base architecture. For the attention block, this architecture utilizes multi-head attention (Vaswani et al., 2023) with rotary positional embeddings (Su et al., 2023). For the MLP block, it uses additional gate projection and SiLU (Elfwing et al., 2017) activation function. We kept the MLP intermediate dimension equal to $8/3$ of the hidden size, padding it to 256 for increased kernel compatibility. For the AdamW optimizer, we used $\beta_1 = 0.90$ and $\beta_2 = 0.95$. We did not apply weight decay to any biases and layer normalizations. Table 2 describes size-specific models and optimizer hyper-parameters for all model sizes used in this work.

Model size	30M	50M	100M	200M	430M	800M
Num. Blocks	6	7	8	10	13	16
Hidden Size	640	768	1024	1280	1664	2048
Num. Attn. Heads	5	6	8	10	13	16
Learning Rate	0.0012	0.0012	0.0006	0.0003	0.00015	0.000075
Num. Tokens	3B	5B	10B	20B	43B	80B

Table 2: Hyper-parameters used for each model size.

B.3 Training stability and convergence

Here we present the loss curves for BF16, LSQ, PACT, and QuEST (ours) to analyze training stability and convergence. As shown in Figure 7(a), QuEST smoothly converges throughout training, closely tracking the BF16 baseline while consistently outperforming LSQ. Meanwhile, PACT struggles with much higher loss, indicating poor convergence. To better highlight the differences between QuEST and LSQ in the later stages of training, Figure 7(b) focuses on steps after 1000, removing PACT for clarity. This zoomed-in view shows that QuEST maintains a consistently lower loss trajectory than LSQ, further reinforcing its superior stability and accuracy across training.

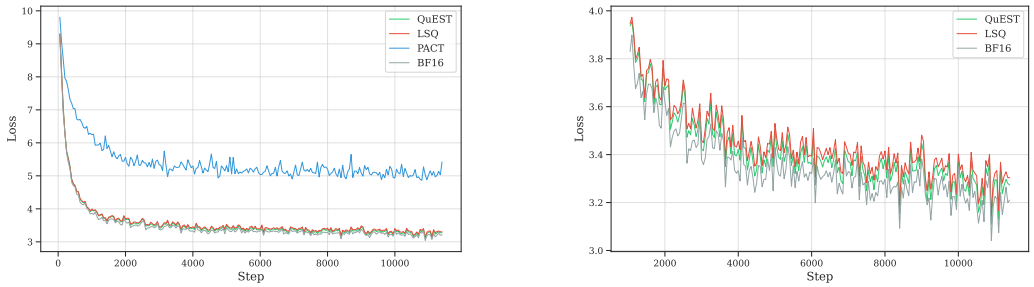

Figure 7: Training loss curves for a 30M model trained on 3B tokens with W4A4 bitwidth, comparing QuEST (ours), LSQ, PACT, and BF16. **(a)** Full training loss curves, showing that QuEST closely follows BF16 and consistently outperforms LSQ, while PACT struggles with high loss. **(b)** Zoomed-in view of training steps after 1000, excluding PACT for clarity, highlighting that QuEST maintains a lower loss than LSQ throughout training.

B.4 Hyper-parameter search for baseline methods

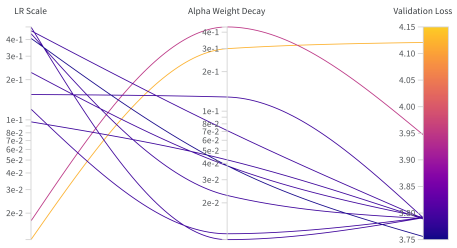

Figure 8: Hyperparameter search for PACT on a 30M parameter model with 4-bit weights and activations, trained on 10% of the dataset. The search explores different values for learning rate scaling (LR Scale) and alpha weight decay, with validation loss indicated by the color gradient. Lower validation loss (darker colors) corresponds to better configurations.

To ensure fair comparisons between QuEST and prior QAT methods, we conducted hyperparameter searches for both PACT and LSQ. Given PACT's instability at lower bitwidths, we extensively tuned two key hyperparameters: weight decay and learning rate scaling s for the quantization parameter

α (i.e., $\eta_\alpha = s \times \eta$). Figure 8 shows the loss achieved across different weight decay and LR scale values.

For LSQ, we only tuned weight decay, as the LSQ formulation already applies scaling internally to the gradient of α, making additional learning rate adjustments unnecessary. Table 3 summarizes the results of the weight decay search across 2-bit, 3-bit, and 4-bit LSQ models, where the best-performing configuration (highlighted in bold) was used for final model comparisons.

Weight Decay	2-bit PPL ↓	3-bit PPL ↓	4-bit PPL ↓
0.001	37.02	31.10	27.93
0.01	36.91	30.89	27.72
0.1	**36.54**	**30.26**	**27.51**
1.0	38.12	31.16	28.67

Table 3: Weight decay hyperparameter search results for LSQ across different bitwidths of 30M model. The best-performing setting is highlighted in bold.

Our hyperparameter search ensured that LSQ and PACT were tuned optimally before comparing against QuEST, leading to a fair evaluation of performance across all tested quantization methods.

C Scaling Laws

C.1 Description of the fitting procedure

To estimate A, B, E, α, β and eff(P) for every quantization precision P we need, we fit Equation 2 by minimizing the Huber loss (Huber, 1964) between the predicted and the observed log loss.

We closely follow the fitting procedure of Hoffmann et al. (2022). Specifically, we copied their grid of initialization given by: $\alpha \in \{0., 0.5, \ldots, 2.\}$, $\beta \in \{0., 0.5, \ldots, 2.\}$, $e \in \{-1., -.5, \ldots, 1.\}$, $a \in \{0, 5, \ldots, 25\}$, and $b \in \{0, 5, \ldots, 25\}$. We also reuse their $\delta = 10^{-3}$ for the Huber loss. In addition, we fit the eff(P) coefficient for a number of quantization schemes described below:

- QuEST for $P \in \{1, 2, 3, 4, 8\}$.
- Weight-only QuEST for $P \in \{1, 2, 3, 4\}$.
- QuEST without the HT for $P \in \{1, 2, 3, 4, 8\}$.
- QuEST with FP4 grid.
- QuEST with 2:4 INT4.

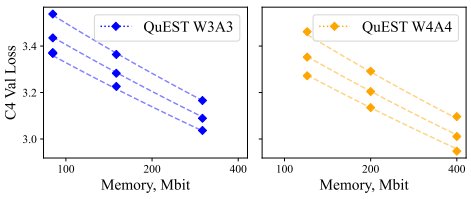

Figure 9: Scaling law (2) fit for 3 and 4 bit QuEST with tokens/parameters ratios in $\{25, 50, 100\}$.

Specifically, we fit the model on the range of parameters $P \in \{1, 2, 3, 4, 16\}$, $N \in \{30, 50, 100, 200, 430, 800\} \times 10^6$ and $D = 100 \times N$. The resulting fit is presented on Figures 1, 3 and C.1. To capture a larger range of D, we fit the model on additional runs with $P \in \{2, 3, 4\}$, $N \in \{30, 50, 100\} \times 10^6$ and $D/N \in \{25, 50\}$.

C.2 Analysis of the transitory data regime

The results in Section 3.1 suggest that 4-bit training is optimal in the $D/N \to \infty$ regime. Here, we use the fitted scaling law (2) to verify that 4 bit is also close to optimal for D/N ratios that are reasonable in practice. We formulate the question as follows: for a fixed model size (e.g. in Gb), for which amount of compute is QuEST 4-bit the optimal precision?

Figure 10 demonstrates the (predicted) dependence of performance as a function of $\frac{D}{N} \cdot \frac{16^2}{P^2}$. For BF16, this quantity becomes D/N. For other P, it ensures the same amount of training computed ($\sim ND$). As such, models there are compared at both the same size and the same training compute. We can see that 4-bit quantization becomes optimal after it passes a certain compute threshold that

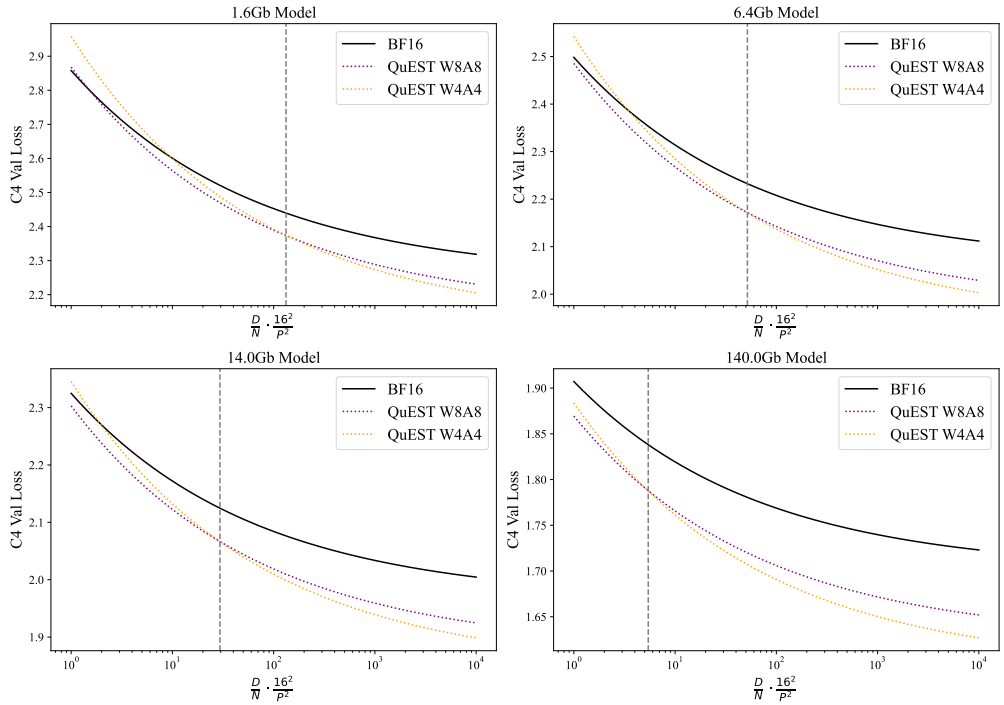

Figure 10: Different QuEST precision performance as a function of tokens-to-parameters ratio at a fixed model memory footprint. The gray line indicates a 4-bit optimality threshold.

depends on model size. We can also see that the threshold value decreases as the model size (in Gb) grows. For a 14.0Gb model (corresponding to 7B parameters in BF16), the threshold is around $D/N \approx 30$, which is significantly below the amount of data that models of that size are currently trained on (see Section 3). For even larger models, the threshold eventually becomes less than the "Chinchilla-optimal" ratio of $D/N \approx 20$. This validates that the regime in which 4-bit pre-training is optimal can, in fact, be easily achieved in practice.

D GPU EXECUTION SUPPORT FOR QUEST MODELS

Kernel Overview. Finally, we describe GPU kernel support. Our forward-pass pipeline for the quantized linear layer in QuEST consists of three main stages: (1) applying the Hadamard transformation to the BF16 activations, (2) quantizing the BF16 activations and packing them into the low-precision format, and (3) performing low-precision matrix multiplication on the quantized activations and weights, followed by dequantization of the result back to BF16.

For the first stage, we utilize an existing Hadamard kernel (Tri Dao). We developed a custom Triton kernel for the second stage to fuse scaling, quantization and memory packing. This kernel computes RMS-based group scales, performs centered quantization on the activations and packs the quantized activations. The third stage involves fused matrix multiplication and dequantization using our enhanced CUTLASS kernel. In this stage, both activations and weights are read and processed as integers to utilize low precision tensor cores for higher throughput. The results are then dequantized back to BF16 within the same kernel. We also apply CUDA Graph end-to-end to reduce the kernel launching overhead.

To optimize GEMM performance, we carefully tuned the CUDA thread-block and warp tile sizes and leveraged the high levels of the memory hierarchy to fuse the dequantization step before writing the results back to Global Memory in a custom CUTLASS *epilogue*.

Runtime Results. The per-layer speedups achievable using our kernel at 4-bit precision, relative to standard 16-bit matrix multiplications on the corresponding layers, are illustrated in Figure 11. We provide a breakdown across layers of the same shape, for 800M (which we have already trained),

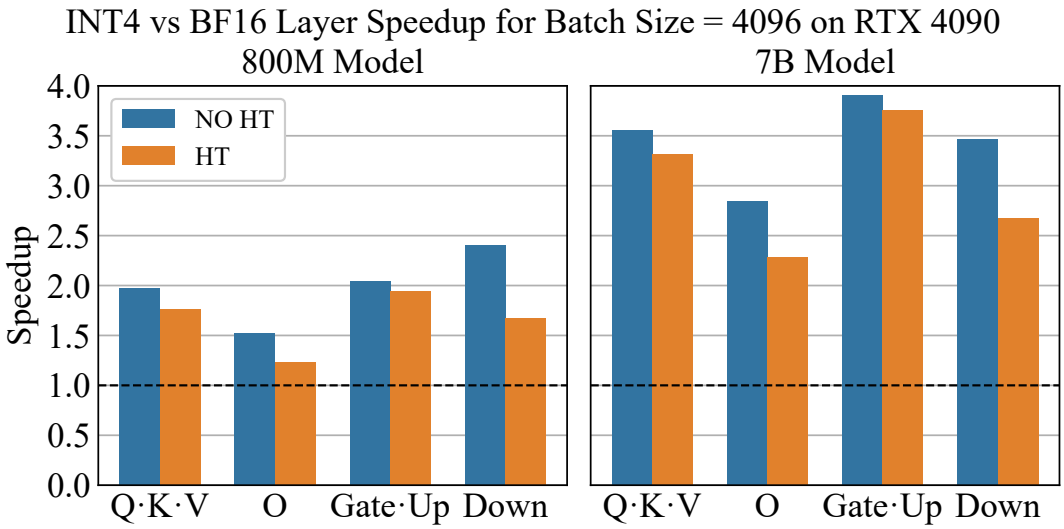

Figure 11: Per-layer speedups for QuEST INT4 vs BF16, on a single RTX 4090 GPU. The results take into account quantization/dequantization costs for QuEST, and include the cost of the Hadamard transform (orange bar). We present results for the 800M 4-bit QuEST model we trained, as well as inference speedups for a proportional 7B-parameter model.

and a proportionally-scaled 7B model (which we plan to train in future work). These measurements include all auxiliary overheads (e.g. quantization/dequantization) for QuEST; in addition, we separate out the performance impact of the Hadamard transform.

For the smaller 800M model, the per-layer speedups vary between 1.2× (on the smallest layers, with Hadamard) and 2.4× (largest down-projection layer, no Hadamard). The largest overhead of the Hadamard transform, of around 30%, is on the down-projection layer, which presents the largest dimension for the Hadamard. The speedups increase significantly (2.3-3.9×) when we move to the 7B-parameter model, as the MatMuls are much more expensive.