Distributional Quantization of Large Language Models

Radostin Cholakov

Under the direction of

Han Guo
Prof. Yoon Kim
Massachusetts Institute of Technology

Research Science Institute
August 1, 2023

Abstract

As large language models (LLMs) continue to grow in size and complexity, efficiently storing and utilizing them without overwhelming computational resources emerges as a crucial challenge. We present a novel method for quantization of LLMs, which stores their parameters in 4 bits while maintaining a performance level comparable to full-precision models. The weight matrices are split into blocks, and for each block, quantization bins are computed as quantiles of a probability distribution. We analyze blocks following Gaussian, Beta, and Student's t distributions and provide intuition about the parametric assumptions made for each of them. Also, a numerical optimization algorithm is introduced to further minimize the loss of precision for quantizing each block. Our method significantly reduces reconstruction error compared to current 4-bit quantile quantization works with the same storage requirements. A further discussion on when the loss reduction can enhance actual language modeling performance is provided, and we successfully report state-of-the-art results on the LLaMA-2 model in terms of text generation perplexity on the WikiText-2 benchmark.

Summary

Despite the usefulness of large language models, their significant size and computational needs pose a barrier to widespread usage and accessibility on consumer hardware. In this work, we investigate methods to reduce the size of LLMs with a novel approach for compressing their weights by rounding to samples from a probability distribution. We explore what the optimal distribution is for each block and how to efficiently compute its parameters. An optimization algorithm that can further refine the initial assumptions is also introduced. Using the same storage as previous techniques, we achieve a significant reduction in precision loss for quantizing weights of modern LLMs. Finally, we provide an analysis of when a decrease in that loss could lead to an increase in model performance and report state-of-the-art text generation quality with the LLaMA-2 language model, released just in July 2023.

1 Introduction

The field of Natural Language Processing has evolved significantly with the integration of Large Language Models (LLMs) [1, 2, 3]. These models are now used across a broad spectrum of tasks including language understanding, text generation, translation, summarization, sentiment analysis, question answering, and code generation, and can achieve near-human abilities in all of them. Earlier approaches primarily relied on recurrent neural networks processing text one item at a time based on the result from the previous predictions in the sequence. This imposed two key obstacles: (i) modeling relationships far apart in input sequences was challenging due to propagation constraints; (ii) the computational complexity caused by the sequential nature of these networks made them inefficient for scaling. The introduction of the Transformer architecture [4] changed the paradigm of language modeling by breaking away from recurrent processing. Instead, Transformers use attention mechanisms to capture dependencies irrespective of where they occur in the input text. They also allow all sequence items to be processed in parallel, facilitating scalability.

Having these properties, Transformer models started growing large, and the concept of pre-training them on unstructured text corpora and then fine-tuning for a specific task with less data emerged. Presently, state-of-the-art LLMs [5, 6, 7, 8] are trained on trillions of text tokens, and their parameter counts vary from a few billion up to more than 540 billion. This results in significant computational power and storage requirements far surpassing what is currently available with standard consumer hardware. For instance, even the relatively modest-sized model LLaMA-65B [7] needs 130GB of GPU RAM for inference and more than 780 GB for fine-tuning with new data.

A way of compressing the size of the models is to quantize their parameters, which are usually stored in 32-bit or 16-bit floating points, to lower precision data types and reconstruct an approximation when needed. It is usually achieved by either scaling and rounding values

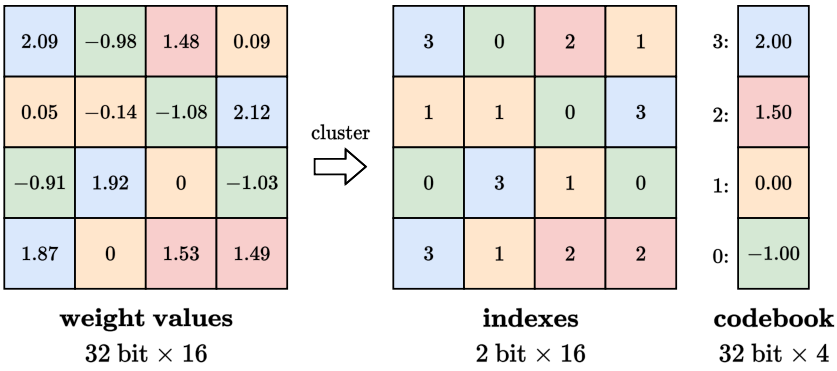

Figure 1: An example of *K-means quantization* in 2 bits illustrating a weight matrix of a language model. Its floating-point elements are grouped into four clusters, and within each group, the values are replaced by a 2-bit integer index. Only four full-scale floating points, representing the centroids for each cluster, are stored.

to a limited set of approximations or grouping values into clusters, each represented by an approximate value.

Past research has demonstrated the successful storage of LLMs in 8-bit [9, 10, 11] and even the ability to train or use them [12] with low-precision arithmetic without significant performance degradation. Recent advancements exploit structural characteristics of the distribution of weight values to push the limit to 4-bit [13] and even 3-bit [14, 15] quantization. Some of them rely on the fact that deep neural networks, and language models in particular, have normally distributed weights to perform **quantile quantization**. In it, all the possible weight values are rounded to quantiles of a normal distribution with empirically optimal mean and standard deviation.

Though this already allows for inference and fine-tuning of performant language models on consumer-grade hardware, there are no extensive studies on where the limits of quantile quantization are. In this work, we explore the cases in which quantiles for quantization are computed with various distribution types and distribution parameters. We show that a numerical optimization algorithm is sufficient to learn block-wise standard deviations for normally distributed weights leading to a considerable reduction in reconstruction loss with

no increase in storage. We can further decrease the reconstruction loss with a storage trade-off by allowing certain blocks of the model to be quantized with quantiles from Beta or Student's t distributions. Finally, we discuss the relation between reconstruction error and real language modeling performance and present state-of-the-art results for LLaMA-2.

Section 2 provides key terms and prior quantization methods. Section 3 presents our novel approach for quantization with quantiles from different distributions. Experiments quantifying performance gains and storage costs are described in Section 4. Section 5 contains the results.

2 Background

2.1 Uniform Quantization

The most widely used quantization methods are **zero-point** and **absolute maximum** quantization. Both of them work by rescaling and rounding off the original values within a specific range. For instance, if we want to store each parameter in 8 bits instead of 32, we can use an integer data type that has 2^8 possible uniform values in the range $[-128, 127]$. All the parameters will be rounded to those values (mentioned later as *bins*) and a single 32-bit scale will be saved. In this case, the memory usage decreases from $n \times 32$ to $n \times 8 + 32$ bits for n parameters. In the reverse operation, multiplying the scale by the 8-bit integers produces an approximation of the original values.

Zero-point integer quantization scales entries of a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ by the scale s to the range $[0, 2^b - 1]$ or, when signed, $[-(2^{b-1} - 1), 2^{b-1} - 1]$ for a b-bit integer data type. The method is used when the values are asymmetrically distributed with an additional zero-point parameter z to shift the values. More formally, it is defined as:

$$\text{quantize}_{\text{i}b}(\mathbf{W}_{\text{f32}}) = \left\lfloor s_{\text{f32}} \cdot \mathbf{W}_{\text{f32}} \right\rceil - z_{\text{i}b} = \mathbf{W}_{\text{i}b}, \tag{1}$$

$$s_{\text{f32}} = \frac{2^b - 1}{\max(\mathbf{W}_{\text{f32}}) - \min(\mathbf{W}_{\text{f32}})}, \quad z_{\text{i}b} = \left\lfloor s_{\text{f32}} \cdot \min(\mathbf{W}_{\text{f32}}) \right\rceil, \tag{2}$$

$$\text{dequantize}(\mathbf{W}_{\text{i}b}, s_{\text{f32}}, z_{\text{i}b}) = \frac{\mathbf{W}_{\text{i}b} + z_{\text{i}b}}{s_{\text{f32}}} = \mathbf{W}_{\text{f32}}, \tag{3}$$

where nearest-integer rounding is denoted as $\lfloor \cdot \rceil$. It utilizes the full range of the low-precision data type, which leads to lower reconstruction error. However, it does have practical constraints, specifically requiring more storage space to accommodate an additional zero point.

Absmax quantization also scales the values down to the range of the low-precision data type by s_{32} as follows:

$$\text{quantize}_{\text{i}b}(\mathbf{W}_{\text{f32}}) = \left\lfloor \frac{(2^{b-1} - 1) \cdot \mathbf{W}_{\text{f32}}}{\max(|\mathbf{W}_{\text{f32}}|)} \right\rceil = \left\lfloor s_{\text{f32}} \cdot \mathbf{W}_{\text{f32}} \right\rceil = \mathbf{W}_{\text{i}b}, \tag{4}$$

$$\text{dequantize}(\mathbf{W}_{\text{i}b}, s_{\text{f32}}) = \frac{\mathbf{W}_{\text{i}b}}{s_{\text{f32}}} = \mathbf{W}_{\text{f32}}. \tag{5}$$

For each set of quantized numbers, a single scale s_{32} is stored, thus saving more space compared to zero-point quantization. A notable limitation of this method is the inability to account for asymmetrically distributed data or outliers. When such are present, some of the quantization bins will contain a few or no values.

2.2 Non-uniform Quantization

As noted in [15, 16], whether the ranges of each bin are uniformly sized is an important factor to consider depending on the use case. Uniform quantization enables direct arithmetic operations on the quantized values and easy computation of the bin thresholds. However, when the data being quantized is not uniform certain bins will remain underutilized and will produce higher reconstruction errors.

The weights of machine learning models, especially language models, do not distribute uniformly. Instead, they tend to adhere to a normal distribution as shown in Appendix B. Because of that, non-uniform methods are being developed and currently lie behind the state-of-the-art works on quantization. In simple terms, they allocate more bins to zones that exhibit a high density of weights.

K-means quantization groups the values to be quantized into k clusters. Each value is then replaced by one of k indices, typically integers in the range $[0, k-1]$ occupying $\log_2 k$ bits of storage each. A lookup table with k centroids for each cluster is stored in full-precision and then used during dequantization. Usually, the K-means algorithm is implemented by starting with a random selection of cluster centers and assigning each data point to the closest centroid. Then the centroids are recalculated as the mean of all points assigned to them. These steps repeat iteratively until convergence. Formally, given m points $x_1, \ldots, x_m$ it finds centroids $\mathcal{M} = \{\mu_1, ..., \mu_k\} \subset \mathbb{R}$ minimizing the objective

$$\sum_{i=1}^{m} \min_{\mu \in \mathcal{M}} (x_i - \mu)^2. \tag{6}$$

However, in the case of quantization, we are working with scalar values only, and thus a 1-dimensional K-means solution could be used to avoid convergence to local optima. An algorithm for optimal 1D clustering using $\mathcal{O}(mk)$ time and space is given in [17]. Based on Eq. (6), it becomes clear that when quantizing weights $\mathbf{W}$ into $\mathbf{W}_Q$ with elements in $b = \log_2 k$ bits, K-means will consistently produce the lowest reconstruction error $\|\mathbf{W} - \mathbf{W}_Q\|_F$ across all b-bit methods. However, this method requires $n \times b + k \times 32$ bits in total for quantizing n values and storing centroids for k clusters, which is much more expensive than both absolute maximum and zero-point quantization.

Quantile quantization utilizes that fact to produce a code with which values are rounded to quantiles of a normal distribution. It usually requires equivalent storage to

absmax quantization since only the absolute maximum value is needed for quantile computation. For example, the NormalFloat 4-bit type (NF4) [13], designed to be information-theoretically optimal, has values computed as $\tilde{q}_i = \Phi^{-1}(p_i)$ for cumulative probability values $p = \{\delta, \ldots, 0.5, \ldots, 1-\delta\}$ where the elements of the subsets $\{\delta, \ldots, 0.5\}$ and $\{0.5, \ldots, 1-\delta\}$ are evenly spaced. The constant offset δ is set to $\delta = \frac{1}{2}(\frac{1}{32} + \frac{1}{30})$. The $\tilde{q}_i$'s are normalized to the range $[-1, 1]$ as $q_i = \frac{\tilde{q}_i}{\max_i |\tilde{q}_i|}$. Thus, the final code consists of quantiles of the Gaussian $\mathcal{N}(0, \sigma^2)$ with the rather specific standard deviation:

$$\sigma = \frac{1}{\Phi^{-1}(1-\delta)} = \frac{1}{\sqrt{2}\,\mathrm{erf}^{-1}(1-2\delta)}. \tag{7}$$

Yoshida D. (2023) [18] proves that NF4 is not information-theoretically optimal but is near-empirically optimal. However, neither work addresses why the specific offset of $\frac{1}{2}(\frac{1}{32}+\frac{1}{30})$ is used, and, if we can choose a different code distribution based on the weight values.

2.3 Block-wise Quantization

Employing any of the described methods to quantize an entire weight matrix in an LLM can prove to be inefficient due to the presence of outlier elements. A single outlier can distort the representation of the quantization bins, leading to suboptimal utilization. Furthermore, the high variability in weight elements makes their effective representation with specific parameters (such as absolute maximum value in absmax, the standard deviation in quantile quantization, or 2^k centroids in K-means) particularly challenging. To mitigate this, it might be worth making a minor memory trade-off by quantizing sections of the weight matrix independently. Current literature [12, 15] presents various methods to partition a weight matrix into blocks, for instance, by separately quantizing each row or column of the matrix.

2.4 Possible Optimization Methods and Parameterization

To minimize a non-linear objective, such as the reconstruction error, a numerical solver could be used. Among many possibilities, in this project, we make use of the Nelder-Mead algorithm [19, 20]. To optimize d parameters, it utilizes a polyhedron with $d+1$ vertices and iteratively updates its vertices toward an optimal solution by reflection, expansion, contraction, or shrinkage. A full outline is provided in Appendix D. In the context of quantile quantization, the parameter space to be optimized is presented in Table 1.

Distribution	**Range**	**Optimizable parameters**
Normal	$(-\infty, \infty)$	μ - mean, σ - standard deviation, δ - quantile offset
Student's t	$(-\infty, \infty)$	ν - degrees of freedom, δ - quantile offset, c - scaling factor
Beta	$[0, 1]$	α, β - shape, rescaling and shifting might be needed

Table 1: Possible distributions and their parameterization considered for optimization.

3 Methods

In this work, we perform block-wise quantile quantization by choosing an empirically optimal distribution for each block. Initially, we consider the error $\|\mathbf{W} - \mathbf{W}_Q\|_F$ for the weight matrices $\mathbf{W}$ as an objective and show that numerical optimization is enough to produce significant decreases in reconstruction error with no or little memory increase.

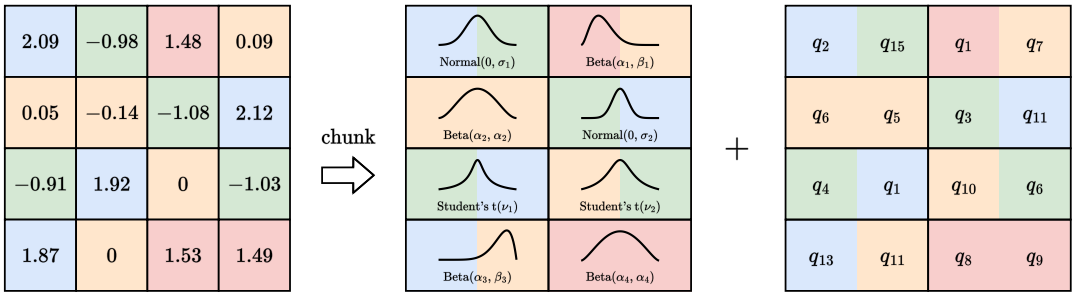

Figure 2: An example of distributional quantization in 4 bits with block size B. A distribution and its parameters are chosen for each block of weights. Then 2^4 quantiles are computed as bins, and the index of each quantile is stored in 4 bits.

As illustrated in Figure 2, weight matrices are divided into contiguous arrays, followed by quantile quantization. Quantiles from a specific distribution for each block are calculated, storing only the distribution parameters in full-scale floating point numbers. To maintain the same storage level as absmax quantization or the quantile quantization outlined in [13], simplified parameterization of distributions is also considered. For instance, setting $\mu = 0$ as a constant for normal distribution and focusing solely on optimizing its standard deviation or constraining α to be equal to β in the beta distribution.

3.1 Numeric Optimization

A way to minimize the reconstruction error for quantizing with an arbitrary distribution is to make a good initial guess about its parameters from the empirical variance of the weights and then use a numerical solver to decrease the objective further. In our work, we choose to use the Nelder-Mead algorithm because it can optimize a various number of parameters, does not require gradient computations, and is able to converge to optima even for complex non-linear functions. The steps performed by the algorithm in the implementation we use are shown in figure 3. Having this intuition, we make parametric assumptions about the distribution of each block in a model and initialize the Nelder-Mead method with parameter estimates to facilitate convergence.

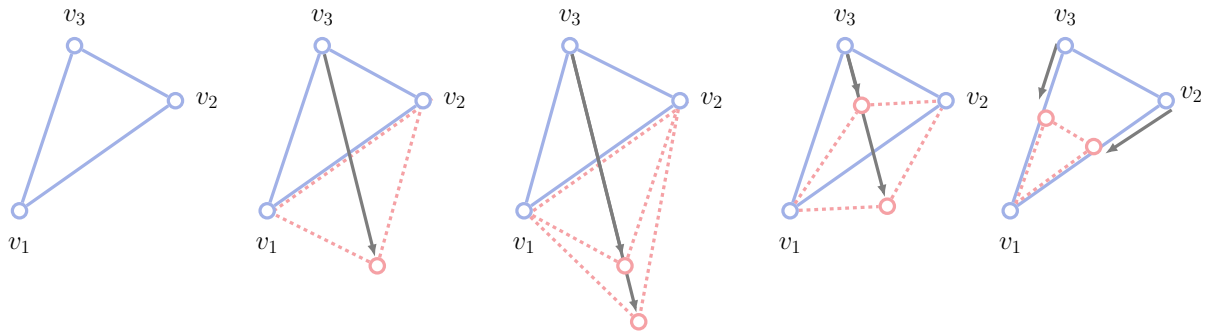

Figure 3: Visualisation of the steps taken by the Nelder-Mead algorithm in 2D.

3.2 Parametric Assumptions

3.2.1 Gaussian Case

Considering that the weights in LLMs typically follow a normal distribution, we first investigate quantizing using quantiles $q_i = \Phi^{-1}(p_i)$ from an optimized Gaussian. The cumulative probability values are $p = \{\delta, \ldots, 0.5, \ldots, 1-\delta\}$ for an offset δ to utilize the full range of a 4-bit floating point type. The initial parameterization is $q_i \sim \mathcal{N}(0, s^2)$ where s is the standard deviation computed from the weight values in each block. Then s is further optimized by the Nelder-Mead algorithm on the reconstruction loss objective. This quantization takes up $B \times 4 + 32$ bits of storage per block with size B. An alternative parameterization to optimize the offset δ instead of the standard deviation s is given in Appendix C.

3.2.2 Student's t Case

The Student's t distribution, controlled by its degrees of freedom ν, generalizes the standard normal distribution. When $\nu = 1$, it resembles the standard Cauchy distribution, and as $\nu \to \infty$, it approaches $\mathcal{N}(0, 1)$. We explore cases where quantization quantiles follow a Student's t distribution with low degrees of freedom ν and thus have heavier tails.

- Set $\nu = 1$, $\nu = 2$, or $\nu = 4$ and draw $\tilde{q}_i \sim$ Student's $t(\nu)$. Optimize a scaling factor c for each case such that $q_i = c\tilde{q}_i$. This approach requires $B \times 4 + 32$ bits per block.

- Set the scale as $c = s$, with s being the standard deviation from Section 3.2.1, and optimize ν. The intuition from Section 3.1 is followed to provide an initial 2D simplex (line segment) to the numerical optimizer with one guess at a low degree of freedom, such as $\nu_1 = 1$, and another at a relatively high value, e.g., $\nu_2 = 64$. This approach allows for the exploration of smaller values, with ν_2 as a fallback. High ν's approximate the Gaussian case and ensure that reconstruction error will remain at least as low during optimization. The method requires $B \times 4 + 2 \times 32$ bits per block.

3.2.3 Beta Case

The weights in some blocks, especially with small block sizes, and in specific parts of the models might follow a Beta distribution. We explore that case by using quantiles $\tilde{q} \sim \mathcal{B}(\alpha, \beta)$ as the quantization bins. Since our data is in the range $[-m, m]$ for a block with absolute maximum value $m = \max(|\mathbf{B}|)$, we need to rescale the Beta samples from their original range $[0, 1]$ to $q_i = m(2\tilde{q}_i - 1)$. The expected Beta mean $\tilde{\mu}$ and variance $\tilde{s}^2$ before rescaling can be calculated from the block mean μ and its empirical variance s^2 as follows:

$$\tilde{\mu} = \frac{1}{N}\sum_{i=1}^{N}\tilde{q}_i = \frac{1}{2m}\frac{1}{N}\sum_{i=1}^{N}(q_i + m) = \frac{\mu}{2m} + \frac{1}{2}, \tag{8}$$

$$\tilde{s}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\tilde{q}_i - \tilde{\mu})^2 = \frac{1}{4m^2}\frac{1}{N-1}\sum_{i=1}^{N}(q_i - \mu)^2 = \frac{s^2}{4m^2}. \tag{9}$$

Then, to find the optimal block-wise values for α and β we run the Nelder–Mead algorithm by initializing it with method-of-moments [21] estimates $\hat{\alpha}$ and $\hat{\beta}$ derived from

$$\tilde{\mu} = \frac{\hat{\alpha}}{\hat{\alpha} + \hat{\beta}}, \quad \tilde{s}^2 = \frac{\hat{\alpha}\hat{\beta}}{(\hat{\alpha} + \hat{\beta})^2(\hat{\alpha} + \hat{\beta} + 1)}. \tag{10}$$

In the general case, they are

$$\hat{\alpha} = \tilde{\mu}\Big(\frac{\tilde{\mu}(1-\tilde{\mu})}{\tilde{s}^2} - 1\Big), \quad \hat{\beta} = (1-\tilde{\mu})\Big(\frac{\tilde{\mu}(1-\tilde{\mu})}{\tilde{s}^2} - 1\Big). \tag{11}$$

To use as little storage as possible, we independently explore the case for quantiles of a symmetric Beta distribution where $\alpha = \beta$. When that assumption is made, the initial estimates simplify to

$$\hat{\alpha} = \hat{\beta} = \frac{1}{8\tilde{s}^2} - \frac{1}{2}. \tag{12}$$

When $\alpha = \beta$ the distribution is symmetric and the CDF evaluated after rescaling samples to $[-m, m]$ is $\Phi_{\mathrm{r}}(0) = 0.5$. This allows us to compute quantiles similarly to the previous cases as $q_i = \Phi_{\mathrm{r}}^{-1}(p_i)$ with probabilities $p = \{0, \ldots, 0.5, \ldots, 1\}$ ensuring 0 is represented as a bin. Otherwise, probabilities are kept as $p = \{0, \frac{1}{15}, \ldots, \frac{14}{15}, 1\}$ for simplicity.

Initialization with random parameters or an estimate different from the one outlined usually prevents convergence to sensible reconstruction errors.

4 Experiments

In order to evaluate the decrease in reconstruction error, an experiment is conducted with the LLaMA model where all weight matrices for the following seven layer types are quantized: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `gate_proj`, and `down_proj`. The storage requirements of each quantization method are computed to assess the potential for error reduction without exceeding the storage requirements of previous works. Additionally, the "best" reconstruction is calculated by choosing the distribution with the lowest error for each block, and its corresponding storage requirements are also quantified.

To assess which of these methods yield real gains in language modeling performance, we run the recently released LLaMA-2 [22] model on the WikiText-2 [23] dataset. Validation perplexities for causal language modeling with a context length of 2048 tokens are computed as follows:

$$\mathrm{PPL}(X) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\right\}, \tag{13}$$

where $X = (x_0, x_1, \ldots, x_t)$ is a tokenized text sequence and $\log p_\theta(x_i|x_{<i})$ represents the log-likelihood of the i-th token in the sequence, according to the language model conditioned with the preceding tokens $x_{<i}$. We calculate PPL for the quantized model with optimized Gaussian and symmetric Beta distribution which demonstrate better reconstruction abilities from the previous experiment.

5 Results

5.1 Reconstruction Error

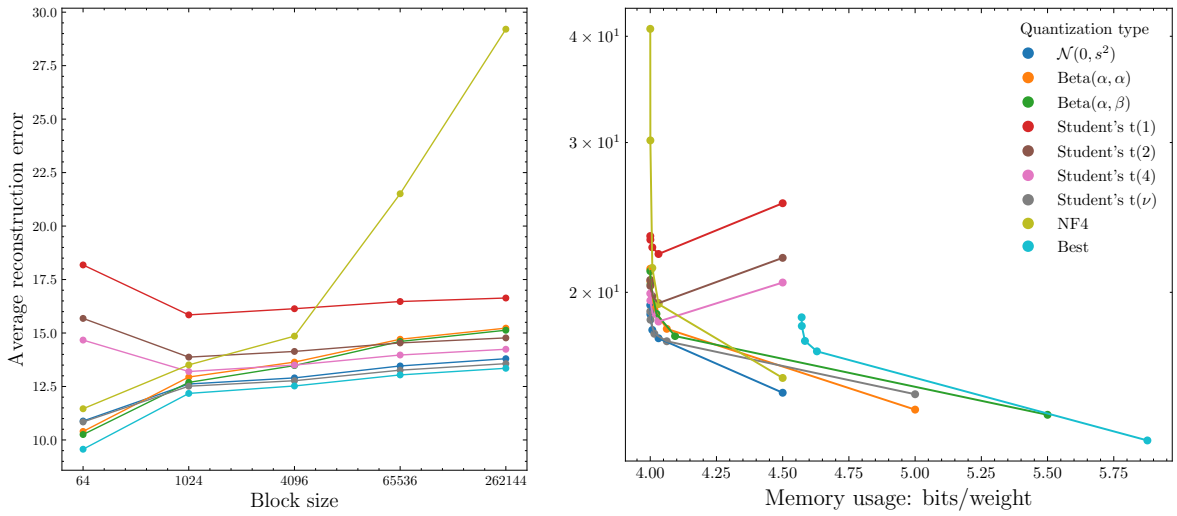

Figure 4: Reconstruction errors for different distributions on LLaMA weights.

Our experiments, illustrated in Figure 4, indicate that quantizing with an optimized normal distribution significantly reduces reconstruction loss compared to NF4 without additional storage. By increasing the block size, the storage requirement for each weight approaches 4.0 bits, and in that case, distributional quantization scales better than NF4, which produces reconstruction errors over twice as high. Quantizing with a mixture of distributions yields further improvements but at the cost of more storage. For example, the naive choice

for a "best" distribution per block requires more than 5.75 bits per weight for a block size of 64. However, as discussed in Appendix F, the difference between reconstruction errors between distributions for some blocks is negligible, so the one with a lower cost could still be used. A detailed overview of how each quantization type performs for different block sizes is given in Appendix E and the count of blocks per optimal distribution in Appendix F.

5.2 Language Modelling

Quantization Method	**Average $\|\mathbf{W} - \mathbf{W}_Q\|_F \downarrow$**			**PPL on WikiText-2 $\downarrow$**		
	64	**1024**	**4096**	**64**	**1024**	**4096**
NormalFloat4	9.26	10.84	11.86	5.64	**5.75**	**5.83**
Optimized Normal (s)	8.89	**10.02**	**10.23**	5.66	6.20	7.12
Optimized Normal (δ)	**8.31**	10.37	10.91	**5.62**	5.77	5.86
Optimized Beta ($\alpha = \beta$)	8.54	10.42	10.88	5.63	5.79	6.12

Table 2: PPL of LLaMA-2 on WikiText-2 and average reconstruction loss for all layers.

Results from the WikiText-2 PPL evaluation show that our distributional quantization outperforms NF4 with LLaMA-2 for some block sizes. For instance, optimizing the offset for a Gaussian produces a perplexity of 5.62 compared to 5.64 for NF4.

5.3 Gaussian Case Ablation Study

In the Gaussian case, we further examine why a standard deviation minimizes reconstruction error yet worsens perplexities. By ablating which layers are quantized, we can confirm that perplexities are always better for optimizing the offset or using normal float quantization rather than optimizing the standard deviation. Our hypothesis is that when q_1 and q_{16} are not explicitly set to $-m$ and m (the absolute maximum value), large improbable values are inaccurately represented in quantization bins, impacting the forward pass

Quantized Layers	**NF4**	Optimized Normal (s)	Optimized Normal (δ)
All	5.640	5.656	5.620
Last layer of each type	5.477	5.706	5.476
Last `down_proj` layer	5.472	5.477	5.471
None	5.467	5.467	5.467

Table 3: PPL of LLaMA-2 with ablated number of quantized layers on WikiText-2.

matrix multiplication. In that case, the difference between logits from an unquantized and a normally-quantized model should be higher than the difference between logits from an unquantized and an NF4-quantized model. Formally,

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\|x\mathbf{W} - x\mathbf{W}_{\mathrm{NL}_\delta}\|_F \quad \leq \quad \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\|x\mathbf{W} - x\mathbf{W}_{\mathrm{NF4}}\|_F \quad \leq \quad \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\|x\mathbf{W} - x\mathbf{W}_{\mathrm{NL}_s}\|_F \tag{14}$$

where $\mathcal{D}$ consists of hidden states from an unquantized model treated as input data to the last layer with weights $\mathbf{W}$, $\mathbf{W}_{\mathrm{NF4}}$ when quantized with NF4, or $\mathbf{W}_{\mathrm{NL}}$ for learned Gaussian. Eq. (14) holds true after experimental evaluation on WikiText-2 with values 632.9, 670.4, and 788.5 for the reconstructed matrices $\mathbf{W}_{\mathrm{NL}_\delta}$, $\mathbf{W}_{\mathrm{NF4}}$, and $\mathbf{W}_{\mathrm{NL}_s}$ respectively.

6 Discussion and Future Work

In terms of reconstruction error, we are able to outperform NF4 with relatively little or no storage tradeoffs. In terms of language modeling abilities, we show that achieving state-of-the-art PPL results is possible with distributional quantization. In addition, it is especially curious that optimizing the standard deviation s for a normal distribution yields better reconstruction loss but noticeably worse perplexity. After examination, we found that the outliers ignored during the minimization of the reconstruction objective might be significant during forward propagation due to their high magnitude. In contrast, NF4 or Beta quantization has the absolute maximum value always represented as a bin.

As a current limitation, we recognize that despite outperforming NF4 with our Beta and

Gaussian (δ) cases, the gains are limited and lack consistency across some block sizes. Thus, the immediate next steps that we identify include:

- Exploration of objectives other than reconstruction loss. They might be more strongly correlated with an increase in language modeling performance.
- Implementation of data-aware distributional quantization. It involves assessing the importance of weight values through gradient analysis and parameterizing the distributions so that the sensitive values are suitably represented as bins.
- Evaluation of model performance on diverse downstream tasks in addition to pure PPL. These might include language understanding [24], question answering [25], etc.

7 Conclusions

To the best of our knowledge, this is the first study that investigates the use of quantile quantization with bins drawn from distributions other than normal. We provide a justification for the initial parameters for each distribution and introduce a numerical optimization method that successfully reduces the reconstruction error in LLMs. It offers improvements over the current state-of-the-art in quantile quantization with the same storage usage as NormalFloat4 in the Gaussian case. Further improvements are made by optimizing one or a few additional parameters per block for blocks following a Beta or Student's t distribution. Moreover, this work reports an enhancement in the perplexity of the LLaMA-2 model, released just in July of 2023, when quantized in small block sizes. We also discuss that lower reconstruction error doesn't necessarily equate to improved perplexity and model abilities and provide a framework to further increase the performance of quantized models in future work.

8 Acknowledgments

I would like to thank my mentors, Prof. Yoon Kim and Han Guo, for the topic suggestion and their advice throughout the development of the project. Thank you to my tutor Dr. Jenny Sendova for the continuous feedback on the different versions of the paper and to Allen Lin, Sally Zhu, AnAn Desimone, Rich Wang, Max Bee-Lindgren, and Donald Liveoak for formatting. Many thanks to Kenneth Choi, Rumen Dangovski, Peter Gaydarov, and my last week TA Michael Huang for the editing help. Recognition for the support of Samet Karaibryamov, Dr. Konstantin Delchev, Todor Kolev (RSI'03), and Acad. Petar Kenderov.

Thank you to the Center for Excellence in Education (CEE), the Massachusetts Institute of Technology (MIT), and the Research Science Institute (RSI) for giving me the opportunity to work in this environment. Finally, I would like to express my gratitude to my sponsors - Prosveta-Sofia Foundation, St. Cyril and St. Methodius International Foundation, MB Consulting, Union of Bulgarian Mathematicians, and Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences.

References

[1] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[5] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[6] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[7] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[8] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[9] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

[10] Y. Lin, Y. Li, T. Liu, T. Xiao, T. Liu, and J. Zhu. Towards fully 8-bit integer inference for the transformer model. *arXiv preprint arXiv:2009.08034*, 2020.

[11] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE, 2019.

[12] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[13] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

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

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

[16] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[17] X. Wu. Optimal quantization by matrix searching. *Journal of algorithms*, 12(4):663–673, 1991.

[18] D. Yoshida. Nf4 isn't information theoretically optimal (and that's good). *arXiv preprint arXiv:2306.06965*, 2023.

[19] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[20] M. Baudin. Nelder-mead user's manual. *Consortium Scilab-Digiteo*, 2010.

[21] K. O. Bowman and L. Shenton. Estimation: Method of moments. *Encyclopedia of statistical sciences*, 3, 2004.

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

[23] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[24] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[25] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[26] J. D. Garrett. garrettj403/SciencePlots. Sept. 2021.

A Quantization Types Comparison

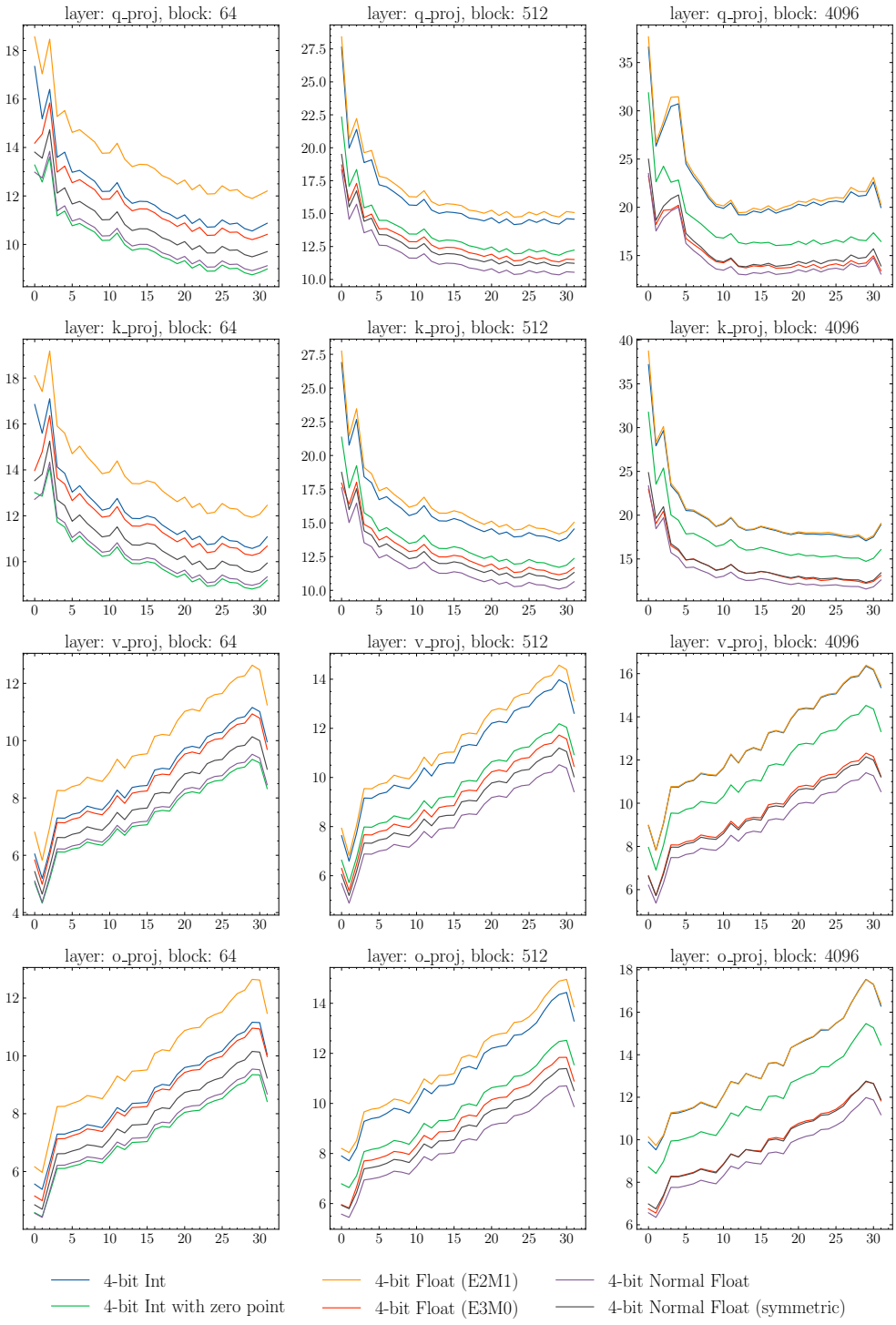

Figure 5: Linear quantization of the LLaMA attention layers with different data types.

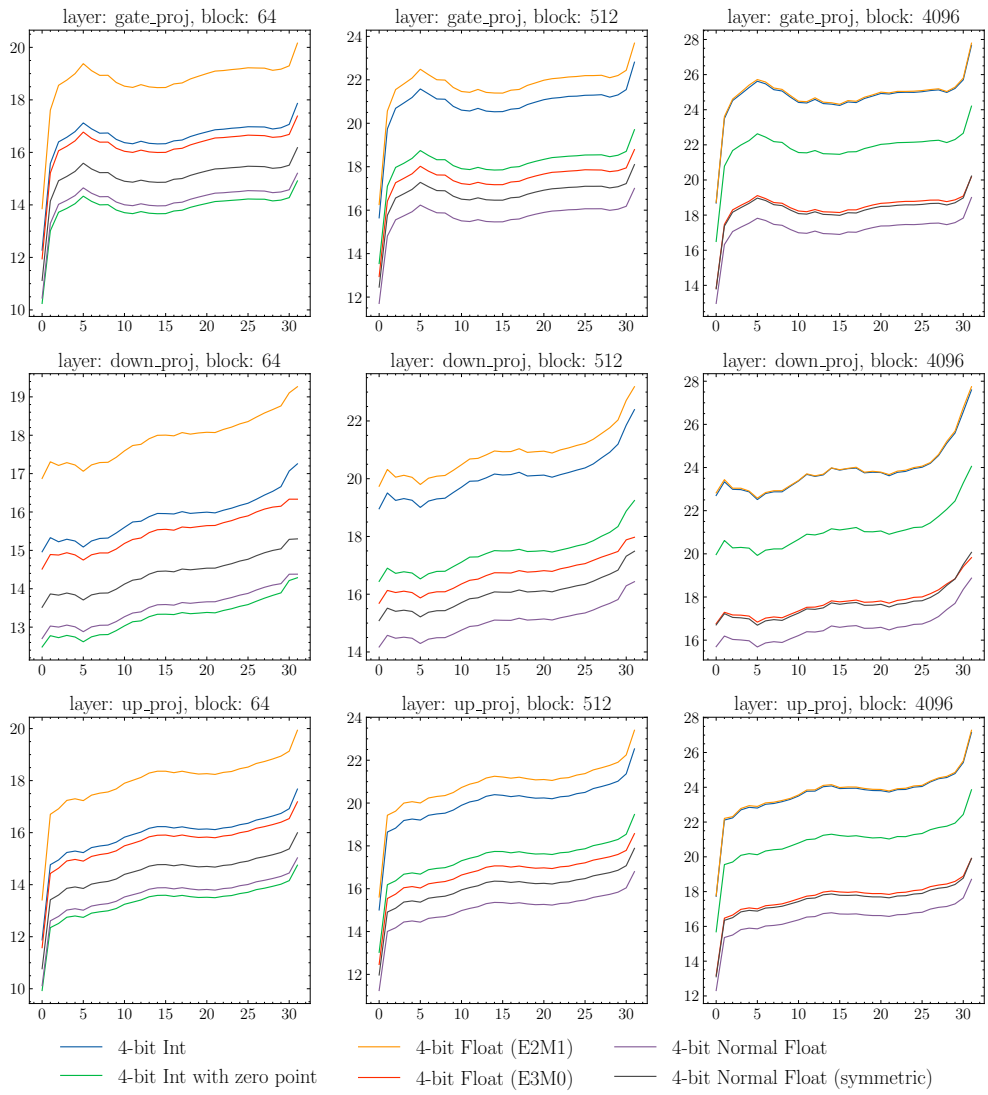

Figure 6: Linear quantization of the LLaMA feedforward layers with different data types.

For comparisons, we implemented zero-point, absmax, and quantile quantization with various underlying data types - 4-bit integer and 4-bit floating point (3 bits for exponent, 0 for mantissa; 2 bits for exponent, 1 for mantissa; normal float). Figure 5 and figure 6 show reconstruction error computed over the 32 layers of each layer type in the LLaMA model.

During the development of this project, an error was identified in the codebase of [13], specifically affecting quantization with block sizes exceeding 1024. This issue impacts both the `bitsandbytes` and Huggingface `transformers` libraries. Upon submitting a bug report along with our own implementation of FP4 and NF4 quantization for reproducibility, Tim Dettmers, the author of [13], confirmed the significance of the problem and acknowledged our debugging code snippets. We appreciate his response.

B Distribution of Weights in LLMs

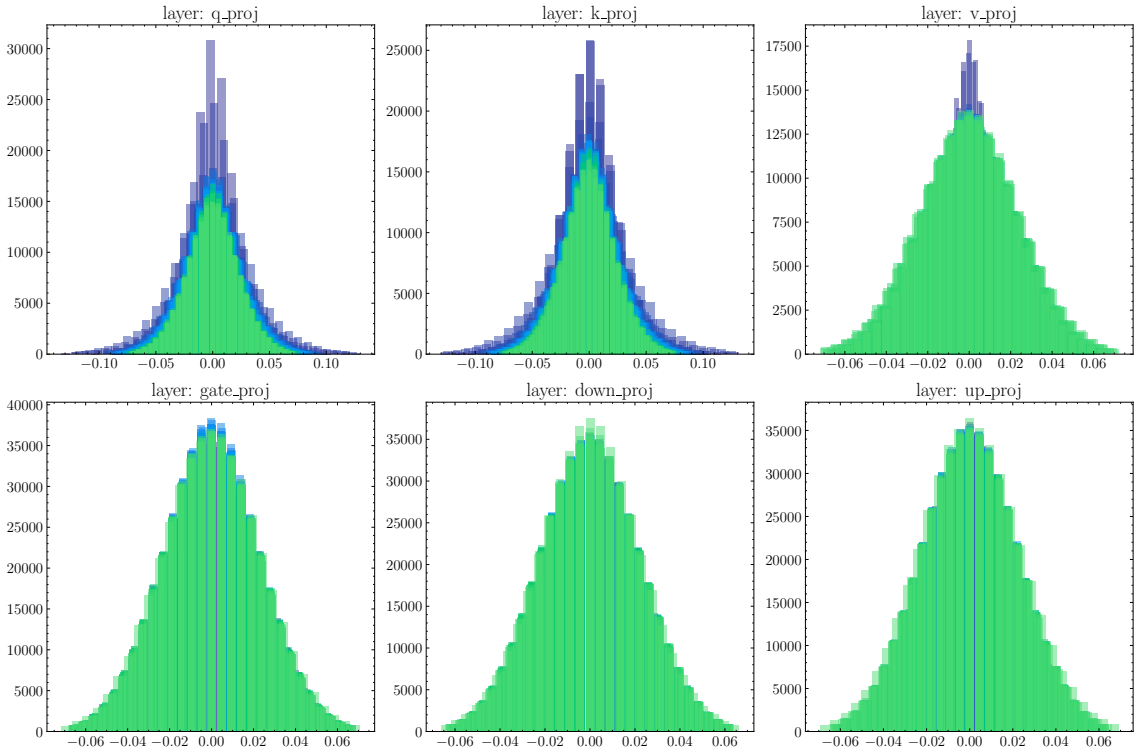

Figure 7: The weight distribution of a non-quantized LLaMA-7B [7] model.

C Alternative Gaussian Parameterization

During our work on minimizing the post-quantization reconstruction loss, we derived a way to optimize the offset δ instead of the standard deviation σ in the Gaussian case if the quantiles are rescaled in $[-1, 1]$. This retains the property of always having the absmax reconstructed exactly which turns out to be important in practice. From Eq. (7) we solve for δ having the standard deviation s of the weights also rescaled in $[-1, 1]$:

$$\frac{1}{\Phi^{-1}(1-\delta)} = s \quad \Leftrightarrow \quad \text{erf}^{-1}(1-2\delta) = \frac{1}{\sqrt{2}s}, \tag{15}$$

$$\text{erf}\left(\frac{1}{\sqrt{2}s}\right) = 1-2\delta \quad \Leftrightarrow \quad \delta = \frac{1}{2}\left(1-\text{erf}\left(\frac{1}{\sqrt{2}s}\right)\right). \tag{16}$$

D Nelder-Mead algorithm

Algorithm 1 Nelder-Mead Algorithm

Parameter Initialize:
$\rho = 1$ {Reflection parameter}, $\chi = 2$ {Expansion parameter}
$\gamma = 0.5$ {Contraction parameter}, $\psi = 0.5$ {Shrinkage parameter}
Initialize: a simplex with v_i vertices, $i = 1, 2, \ldots, d+1$
Calculate $f_i = f(v_i)$ for all i
while $\texttt{std}(f) > tol$ **do**
  Sort v_i's such that $f_1 \leq f_2 \leq \ldots \leq f_d \leq f_{d+1}$
  Compute the midpoint of the simplex excluding v_{d+1}: $v_{mid} = \frac{1}{d}\sum_{i=1}^{d} v_i$
  Reflection: $v_r = v_{mid} + \rho(v_{mid} - v_{d+1})$ and calculate f_r
  if $f_r < f_1$ **then**
    Expansion: $v_e = v_{mid} + \chi(v_r - v_{mid})$ and calculate f_e
    if $f_e < f_1$ **then**
      $v_{d+1} = v_e$
    else
      $v_{d+1} = v_r$
    end if
  else if $f_1 \leq f_r < f_d$ **then**
    $v_{d+1} = v_r$
  else if $f_d \leq f_r < f_{d+1}$ **then**
    Outside Contraction: $v_o = v_{mid} + \gamma(v_r - v_{mid})$ and calculate f_o
    if $f_o < f_{d+1}$ **then**
      $v_{d+1} = v_o$
    else
      for $i = 2, 3, \ldots, d+1$ **do**
        Shrinkage: $v_i = v_1 + \psi(v_i - v_1)$
      end for
    end if
  else
    Inside Contraction: $v_c = v_{mid} - \gamma(v_r - v_{mid})$ and calculate f_c
    if $f_c < f_{d+1}$ **then**
      $v_{d+1} = v_c$
    else
      for $i = 2, 3, \ldots, d+1$ **do**
        Shrinkage: $v_i = v_1 + \psi(v_i - v_1)$
      end for
    end if
  end if
end while

E Detailed Reconstruction Loss Results

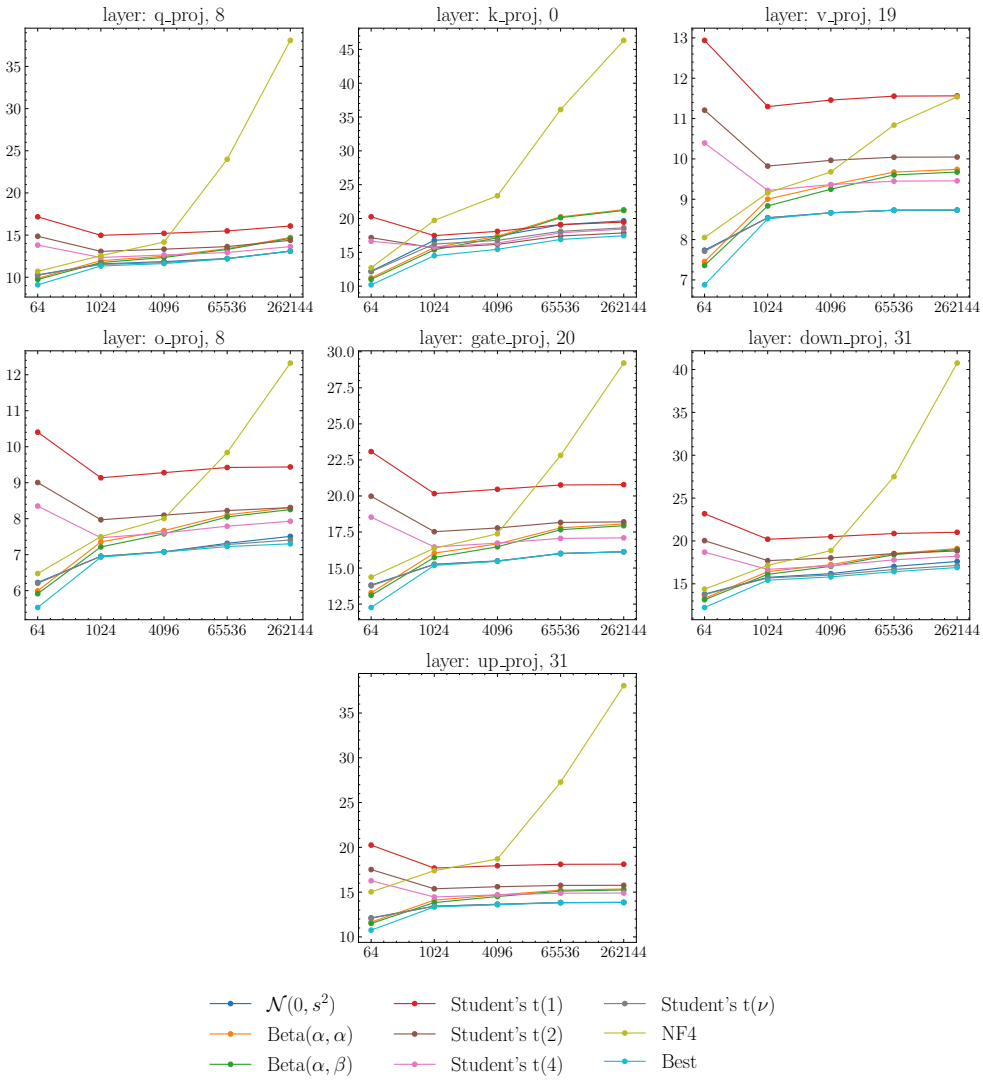

Figure 8: Reconstruction loss across various block sizes when LLaMA layers are quantized using quantiles from different distributions.

Reconstruction errors for NF4 increase substantially for large blocks due to rescaling into $[-1, 1]$, with a higher likelihood of the absolute maximum value being an outlier. Quantile quantization outperforms NF4 in all cases, especially for $B > 1024$. The Beta distribution is an optimal choice for $B = 64$, but $\alpha \neq \beta$ does not contribute further. Additionally, Student's t (ν) does not significantly exceed the simpler optimization of the Normal distribution.

F Detailed Breakdown of Distribution Usage

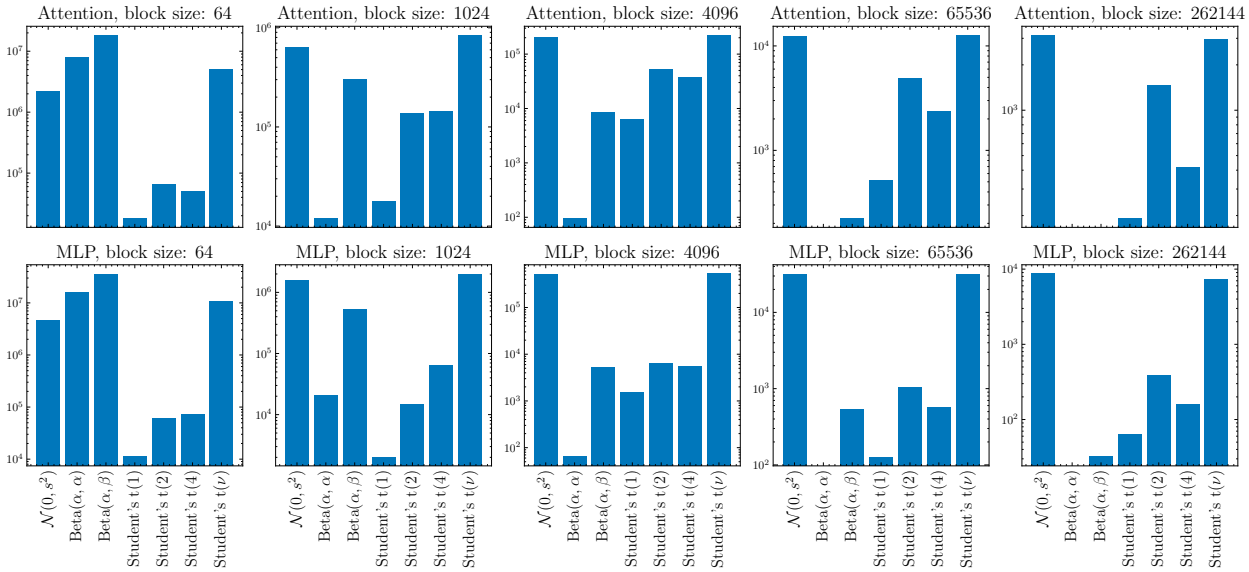

Figure 9: Number of blocks best represented by a distribution.

Figure 9 illustrates the number of blocks in each layer of LLaMA best reconstructed by each type of distribution. The *Attention* blocks are from `q_proj`, `k_proj`, `v_proj`, and `o_proj` layers. The *MLP* blocks are from the `up_proj`, `gate_proj`, and `down_proj` layers. We split the blocks into those groups to determine whether different parts of the model align better with different distributions. The data suggest that there is no substantial difference between quantizing the Transformer attention weights and the final MLP weights. Further, we observe that for smaller block sizes such as 64, the Beta distribution seems to contribute more to the minimization of the objective than other distributions. For larger block sizes, a normal distribution provides the best reconstruction. Theoretically, the Student's $t(\nu)$ should always be as effective as Gaussian or superior as it generalizes to it. Nonetheless, due to the optimizer's iteration limit, we do not typically observe this in practice, with some blocks yielding better reconstructions in the Gaussian. However, the differences in the error, as Appendix E details, are negligible.