The case for 4-bit precision: k-bit Inference Scaling Laws

Tim Dettmers [1] **Luke Zettlemoyer** [1]

Abstract

Quantization methods reduce the number of bits required to represent each parameter in a model, trading accuracy for smaller memory footprints and inference latencies. However, the final model size depends on both the number of parameters of the original model and the rate of compression. For example, a 30B 8-bit model and a 60B 4-bit model have the same number of bits but may have very different zero-shot accuracies. In this work, we study this trade-off by developing inference scaling laws of zero-shot performance in Large Language Models (LLMs) to determine the bit-precision and model size that maximizes zero-shot performance. We run more than 35,000 experiments with 16-bit inputs and k-bit parameters to examine which zero-shot quantization methods improve scaling for 3 to 8-bit precision at scales of 19M to 176B parameters across the LLM families BLOOM, OPT, NeoX/Pythia, and GPT-2. We find that it is challenging to improve the bit-level scaling trade-off, with the only improvements being the use of a small block size – splitting the parameters into small independently quantized blocks – and the quantization data type being used (e.g., Int vs Float). Overall, our findings show that 4-bit precision is almost universally optimal for total model bits and zero-shot accuracy.

1. Introduction

Large Language Models (LLMs) are widely adopted for zero/few-shot inference (Zhang et al., 2022; Black et al., 2022; Zeng et al., 2022; Scao et al., 2022), but they can be challenging to use both due to their large memory footprints – up to 352 GB of GPU memory for 175B models – and high latency. However, both the memory and latency are primarily determined by the total number of bits in the parameters. Therefore, if we reduce the model bits through quantization, we can expect the latency of the model to reduce proportionally, potentially at the expense of end task accuracy (Frantar et al., 2022; Park et al., 2022; Yao et al., 2022).

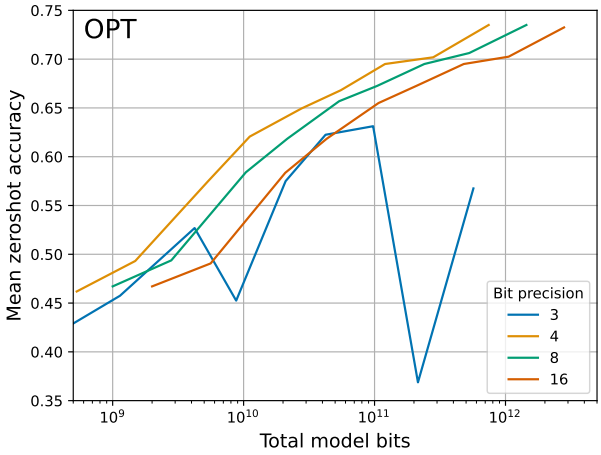

Figure 1. Bit-level scaling laws for mean zero-shot performance across four datasets for 125M to 176B parameter OPT models. Zero-shot performance increases steadily for fixed model bits as we reduce the quantization precision from 16 to 4 bits. At 3-bits, this relationship reverses, making 4-bit precision optimal.

Since we can quantize the parameters of a trained model to an arbitrary bit-precision, this raises the question of how many bits should be used to optimally trade off accuracy and total model bits, given our current base methods for model quantization. For example, **if we have a 60B LLM in 4-bit precision and a 30B LLM in 8-bit precision, which will achieve higher accuracy?** To study such trade-offs, it is helpful to take the perspective of scaling laws (Kaplan et al., 2020; Henighan et al., 2020), which evaluate the underlying trends of variables to make generalizations beyond individual data points.

In this paper, we study bit-level inference scaling laws for zero-shot quantization to determine the precision that maximizes zero-shot accuracy given a certain number of total bits in the model. **Our main finding is that 4-bit parameters yield optimal performance for a fixed number of model**

[1]University of Washington. Correspondence to: <dettmers@cs.washington.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

bits across all model scales and model families tested.

We study five different model families, OPT, Pythia/NeoX, GPT-2, BLOOM, and BLOOMZ (Zhang et al., 2022; Black et al., 2022; Radford et al., 2019; Scao et al., 2022), with 19M to 176B parameters, and for 3 to 16-bit precision. In addition, we run more than 35,000 zero-shot experiments to vary many of the recently studied advances in quantization precision, including the underlying data types and quantization block size. We find that reducing the precision steadily from 16 to 4 bits increases the zero-shot performance for a fixed number of model bits, while at 3-bit, the zero-shot performance decreases. This relationship holds across all models studied and did not change when scaling from 19M to 176B parameters, making 4-bit quantization universally optimal across all cases tested.

Beyond this, we analyze which quantization methods improve and which degrade bit-level scaling. We find that none of the quantization methods we test improve scaling behavior for 6 to 8-bit precision. For 4-bit precision, data types and a small quantization block size are the best ways to enhance bit-level scaling trends. We find that quantile quantization and floating point data types are the most effective, while integer and dynamic exponent data types generally yield worse scaling trends. For most 4-bit models, a block size between 64 to 128 is optimal in our study.

From our results, we can make straightforward **recommendations for using zero-shot quantized models for inference**: always use 4-bit models with a small block size and with a float data type. If trade-offs in total model bits or predictive performance are desired, keep the precision in 4-bit but vary the number of parameters of the model instead.

While earlier work has shown that it is possible to significantly improve the predictive performance of quantized models by using outlier-dependent quantization (Dettmers et al., 2022a; Xiao et al., 2022), we show that this is not effective for bit-level scaling. In particular, we analyze instabilities in 3-bit OPT and Pythia models and show that while these models can be stabilized through an outlier-dependent quantization which we term *proxy quantization*, this does not improve bit-level scaling compared to 4-bit precision. On the other hand, we highlight that one-shot quantization methods, that is, methods that optimize the quantization through a single mini-batch of data, potentially could be scaled to be bit-efficient below the 4-bit level. Overall, these findings suggest that the most promising directions to improve zero-shot bit-level scaling laws are to develop new data types and techniques that quantize outliers with high precision without requiring a significant amount of additional bits. We also highlight the potential for one-shot quantization methods as a way towards low-bit transformers if combined with our insights.

2. Background

It might be unintuitive that reducing the number of bits of a model is directly related to inference latency for LLMs. The following section gives the background to understand this relationship. Afterward, we provide a background on quantization data types and methods.

2.1. Relationship Between Inference Latency and Total Model Bits

While the main goal of our work is to find the best trade-offs between model bits and zero-shot accuracy for LLMs, the total model bits are also strongly related to inference latency. The overall computation latency – the time it takes from start to finish of a computation – is mainly determined by two factors: (1) how long does it take to load the data from main memory into caches and registers, (2) how long does it take to perform the computation. For example, for modern hardware like GPUs, it usually takes more than 100 times longer to load a number than to do an arithmetic operation with that number (Jia et al., 2019; Dongarra, 2022). Therefore, reducing the time spent loading data from main memory is often the best way to accelerate overall computation latency. Such reductions can be achieved mainly through caching and lower precision numbers.

Caching can reduce the overall latency of matrix multiplication by a factor of 10x or more (Jia et al., 2019), given that neither matrix entirely fits into the L1 cache of the device. In matrix multiplication, $\mathbf{bW} = \mathbf{h}$, with the batch $\mathbf{b}$ and parameter matrix $\mathbf{W}$, we load each row of $\mathbf{b}$ and each column of $\mathbf{W}$ and multiply them – thus multiple loads from global memory occur for each row/column which can be cached. If $\mathbf{b}$ entirely fits into the L1 cache, no reuse is possible because neither $\mathbf{W}$ nor $\mathbf{b}$ is loaded more than once from global memory. This occurs if the inference batch size is below 60 or 200 for an RTX 3090 or RTX 4090 GPU. As such, caching is ineffective below these batch sizes, and the inference latency is solely determined by the memory loaded from $\mathbf{W}$.

Thus in the case where the mini-batch fits into the L1 cache, inference latency can be reduced by using a smaller model with smaller $\mathbf{W}$ or by compressing an existing model to use a lower bit-precision per parameter in $\mathbf{W}$. For example, beyond their algorithmic innovation of improved rounding for quantization, Frantar et al. (2022) also developed inference CUDA kernels for 16-bit inputs and 3-bit integer weights, which yields inference latency improvements of up to 4.46x compared to 16-bit inputs and weights for OPT-175B – close to the 5.33x reduction in model bits. As such, reduction in the total model bits is strongly correlated with inference latency for small inference batch sizes. We provide additional data from a roofline model and preliminary data from unoptimized implementations in Appendix E.

2.2. Data types

Here we provide a brief overview of the data types that we study. Please see Appendix A for full specification of these data types.

We use four different data types. For **Integer** and **Float** data types, use IEEE standards. Our Float data type has an exponent bias of 2^{E-1}, where E is the number of exponent bits. We also use **quantile quantization**, a lossy maximum entropy quantization data type (Dettmers et al., 2022b), where each quantization bin holds an equal number of values. This ensures that each bit pattern occurs equally often. Finally, we use **dynamic exponent quantization** (Dettmers, 2016), which uses an indicator bit to separate an exponent bit region and a linear quantization region. The exponent can vary from value to value by shifting the indicator bit. This data type has low quantization error for tensors which have numbers that vary by many orders of magnitude.

2.3. Blocking / Grouping

Quantization precision is, in part, determined by whether all quantization bins are used equally. For example, a 4-bit data type has 16 bins, but if, on average, only 8 bins are used, it is equivalent to a 3-bit data type. As such, methods that help to increase the average use of all quantization bins in the data type can increase quantization precision. In this subsection, we introduce blocking. Blocking/grouping methods chunk the tensor into smaller pieces and quantize each block independently. This helps to confine outliers to particular blocks, which increases the average number of bins used across other blocks and thus increases the average quantization precision.

Blocking/Grouping. Blocking and grouping are similar concepts. In grouping, we sub-divide a tensor along a certain dimension into n parts and assign each sub-tensor, called group, its own normalization constant c, which is usually the absolute maximum of the group. In blocking, we view the tensor as a one-dimensional sequence of values and divide this sequence into parts of size n called blocks.

In our work, we use blocking because, unlike grouping, it provides a measure of additional bits per parameter independent of the hidden dimension. For example, using 16-bit normalization constants and a block size of 64 means, we have an extra 16 bits every 64 parameters or 16/64=0.25 bit-per-parameter additional cost for using block-wise quantization – this is true for every model regardless of hidden dimension. For grouping, the exact cost would depend on the size of the hidden dimension of each model.

For block-wise quantization, we use the notation of Dettmers et al. (2022b), which defines the block-wise quantization with k bits, block-size B, input tensor $\mathbf{T}$ with n elements, n/B blocks as follows. If $\mathbf{Q}_k^{\text{map}}(\cdot)$ maps the integer representation of a data type to the representative floating point value, for example, the bit representation of a 32-bit float to its real value, and if we define the index of each block in $0..n/B$ by index b, and we compute the normalization constant as $m_b = \max(|\mathbf{T}_b|)$, then block-wise quantization can be defined by finding the minimum distance to the value of the quantization map as follows:

$$\mathbf{T}_{bi}^{Q_k^{\text{map}}} = \underset{j=0}{\overset{n=2^k}{\arg\min}} |\mathbf{Q}_k^{\text{map}}(j) - \frac{\mathbf{T}_{bi}}{m_b}| \Big|_{0<i<B}, \qquad (1)$$

3. Outlier-dependent Quantization Through Proxy Quantization

Outlier features that emerge in large language models (Gao et al., 2019; Timkey & van Schijndel, 2021; Bondarenko et al., 2021; Wei et al., 2022; Luo et al., 2021; Kovaleva et al., 2021; Puccetti et al., 2022) can cause large quantization errors and severe performance degradation (Dettmers et al., 2022a; Zeng et al., 2022; Xiao et al., 2022). While it has been shown that it is sufficient to use 16-bit inputs and 8-bit weights to avoid this disruption (Zeng et al., 2022), it is unclear if outlier features can cause degradation if we use 16-bit inputs and weights below 8-bit precision.

To this end, we develop outlier-dependent quantization through proxy quantization, where we quantize weights to a higher precision for the corresponding outlier feature dimensions to test how much precision is needed for the weights. A significant challenge is that each model has a different number of outlier features, and outliers partially depend on inputs that are different for each zero-shot task. As such, we seek a model-independent model that has a constant memory footprint across all models and tasks.

In initial experiments, we noted that the criterion developed by Dettmers et al. (2022a), which thresholds the hidden states to detect outlier features, is unreliable as it depends on the standard deviation of the hidden state. This causes problems because, for models such as OPT, the standard deviation of the hidden states increases in later layers, which causes too many outliers to be detected. This also has been noted by Zeng et al. (2022). By inspecting this anomaly in OPT models, we find that a better measure of detecting outlier dimensions is the standard deviation of the weights of each hidden unit of the previous layer. The standard deviation of hidden unit weights that produce outliers are up to 20x larger than the standard deviation of other dimensions. We provide further data as a correlation analysis of the relationship between standard deviation and outlier size in Appendix D.

With this insight, we develop what we call **proxy quantization**. Proxy quantization is input-independent and, therefore

task-independent, as it uses the standard deviation of each layer's hidden unit weights as a proxy for which dimensions have outlier features. For example, given a transformer with n linear layers (FFN and attention projection layers) with weight matrices $\mathbf{W}_i \in \mathbb{R}^{h \times o}$ where h is the input dimension and o the output dimension (thus o hidden units), we define the set of indices J to be quantized in higher precision by:

$$J_{i+1} = \arg\max_{j=0}^{k} \text{std}(\mathbf{W}_i)\big|_{i..n} \tag{2}$$

where $\text{std}(\cdot)$ is the standard deviation of the output dimension o. We then quantize the input dimensions of the weight of the next layer in 16-bit if it is in set J and k-bit otherwise.

4. Experimental Setup

In our experiments, we use 16-bit inputs and k-bit quantized parameters for $3 \geq k \geq 8$. Attention matrices are not quantized since they do not contain parameters. We also use a 16-bit baseline that does not use any quantization (16-bit floats).

To measure inference performance for k-bit quantization methods, we use perplexity on the CommonCrawl subset of The Pile (Gao et al., 2020) and mean zero-shot performance on the EleutherAI LM Evaluation harness (Gao et al., 2021). In particular, for the zero-shot setting, we use the EleutherAI LM eval harness (Gao et al., 2021) in the GPT-2 setting on the tasks LAMBADA (Paperno et al., 2016), Winogrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), and PiQA (Bisk et al., 2020).

The choice of these particular zero-shot tasks was mainly motivated by previous work (Dettmers et al., 2022a; Yao et al., 2022; Xiao et al., 2022). However, in our evaluation, we find that perplexity is a superior metric since its continuous value per sample leads to less noisy evaluations. This has also been noted by Frantar et al. (2022). For example, when using perplexity to evaluate data types, quantile quantization is the best data type. Still, when we use zero-shot accuracy as an evaluation metric, the float data type is sometimes better because zero-shot accuracy is noisier. Furthermore, we find that across more than 35,000 zero-shot experiments, the **Pearson correlation coefficient between The Pile Common Crawl perplexity and zero-shot performance is -0.94**.

This highlights that perplexity is sufficient and preferable for evaluation purposes. A serious drawback is that perplexity is challenging to interpret. As such, we use zero-shot accuracies in the main paper for clarity but encourage the reader to use perplexity measures found in the appendix for replication and comparison purposes. Since perplexity evaluations are so reliable, it is possible to replicate our work by evaluating a small number of samples using the perplexity metric, which makes the construction of scaling laws less computationally intensive.

Scaling laws. We try to fit power laws to our data, but we find that bivariate power functions with respect to the number of parameters and the bit-precision provide a poor fit. However, when we fit linear interpolations to represent scaling curves, we find that different bit-precisions are almost parallel, indicating that the scaling trends of different precisions can be represented faithfully by a base function and an offset for each bit-precision. As such, we choose to use linear interpolations to represent scaling trends.

5. Results & Analysis

5.1. Bit-level Inference Scaling Laws

The main results are shown in Figure 2, which depicts the mean zero-shot accuracy over Lambada, PiQA, HellaSwag, and Windogrande, given the total number of bits for OPT, BLOOM, Pythia, and GPT-2 for 3 to 16-bit parameters.

We make the following observations:

1. For a given zero-shot performance, 4-bit precision yields optimal scaling for almost all model families and model scales. The only exception is BLOOM-176B where 3-bit is slightly but not significantly better.
2. Scaling curves are almost parallel, which indicates that bit-level scaling is mostly independent of scale. An exception to this is 3-bit quantization.
3. Pythia and OPT are unstable for 3-bit inference where performance is close to random (35%) for the largest Pythia/OPT models.

5.2. Improving Scaling Laws

Given the main scaling results in Figure 2, an important follow-up question is how we can improve the scaling trends further. To this end, we run more than 35,000 zero-shot experiments to vary many of the recently studied advances in quantization precision, including the underlying data types, quantization block size, and outlier-dependent quantization.

These methods usually improve the quantization error at a small cost of additional bits. For example, a block size of 64 with 16-bit quantization constants uses 16 extra bits for every 64 parameters, or 16/64 = 0.25 additional bits per parameter. Outlier-dependent quantization stores the top $p\%$ of weight vectors in 16-bit precision and increases the bits per parameter by $p(16-k)$, where k is the precision of the regular weights. For example, for $p = 0.02$ and $k = 4$, the additional memory footprint is 0.24 bits per parameter.

No scaling improvements for 6 to 8-bit precision. We combine all possible combinations of quantization methods (data types, blocking) with 6 to 8-bit quantization, and we

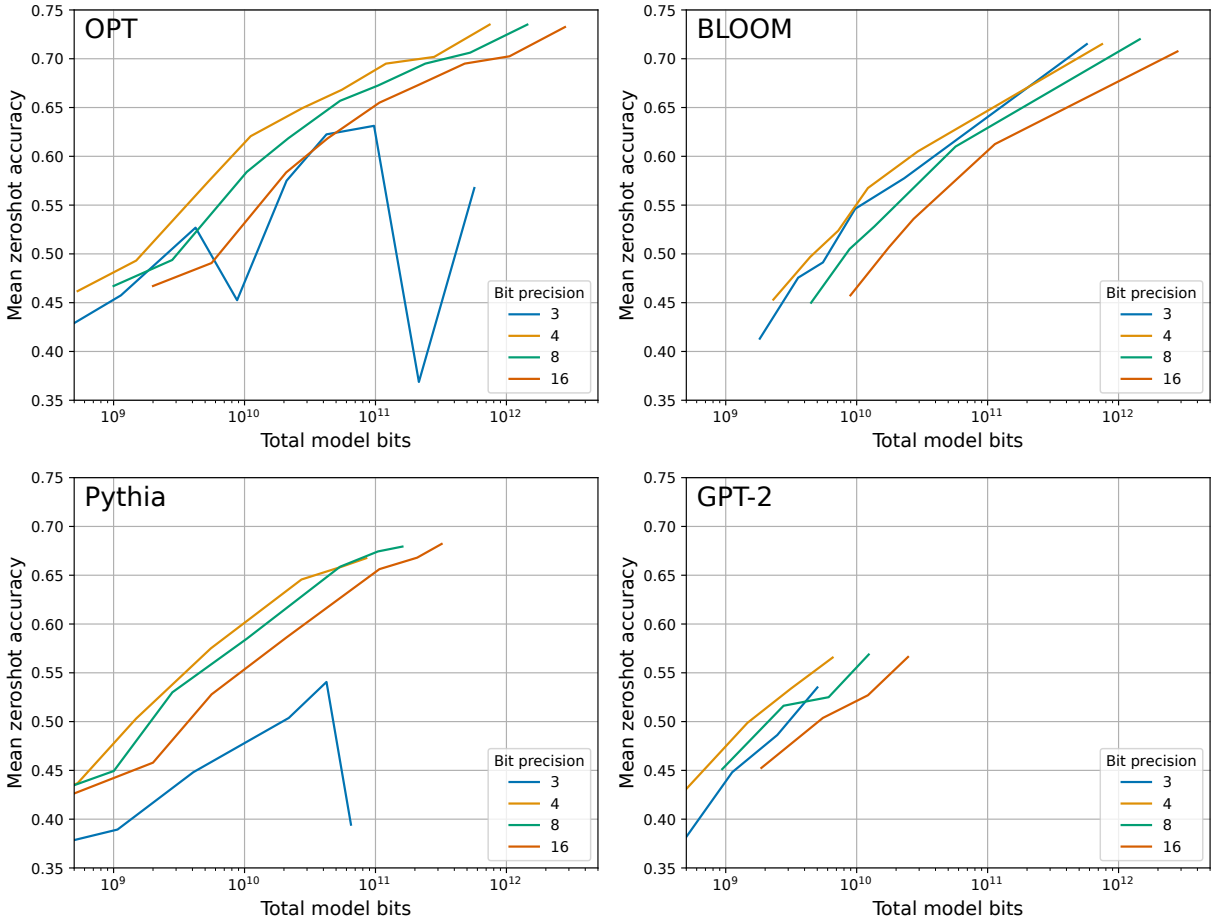

Figure 2. Bit-level scaling laws for mean zero-shot performance across Lambada, PiQA, Winogrande, and Hellaswag. 4-bit precision is optimal for almost all models at all scales, with few exceptions. While lowering the bit precision generally improves scaling, this trend stops across all models at 3-bit precision, where performance degrades. OPT and Pythia are unstable at 3-bit precision, while GPT-2 and BLOOM remain stable. Plots for all intermediate bit precisions can be found in the Appendix C.1.

find that none of these methods improve bit-level scaling (see Appendix C.3). It appears that for 6 to 8-bit precision, the model parameters have enough precision to not cause any significant performance degradation compared to 16-bit weights. As such, scaling behavior can only be improved by using less than 6-bit precision rather than enhancing the quantization precision through other means.

Small block size improves scaling. For 3 to 5-bit precision, we do see improvements in scaling for various quantization methods. Figure 3 shows that for 4-bit Pythia models, considerable improvements in bit-level scaling can be achieved by using a small block size. To put this improvement into perspective: Going from a block size of 1024 to 64 adds 0.24 bits per parameter but improves zero-shot accuracy almost as much as going from 4 to 5-bit precision. As such, using a small block size adds a few extra bits compared to improving zero-shot accuracy for 4-bit precision. Besides Pythia, GPT-2 models improve by a large degree. BLOOM, BLOOMZ, and OPT models improve significantly, but less in magnitude compared to Pythia and GPT-2 (see Appendix C.2) – this relationship likely arises from emergent outlier features. For 5-bit models, the improvement of using small block sizes is minor but still significant. Small block sizes improve 3-bit scaling considerably but still do not make it competitive with 4-bit precision scaling.

Data types improve scaling. From Figure 3, we see that data types improve scaling trends for 4-bit Pythia. In particular, the quantile quantization and float data types provide better scaling than integer and dynamic exponent quantization. We generally find that quantile quantization is the best data type across all models, scales, and precisions (see Appendix C.5). The float data type seems to be superior to Integer quantization with a few exceptions: Integer quantization is better than float quantization for 5-bit – it appears since the float data type is quite dependent on the balance between exponent and fraction bits, the float data type can

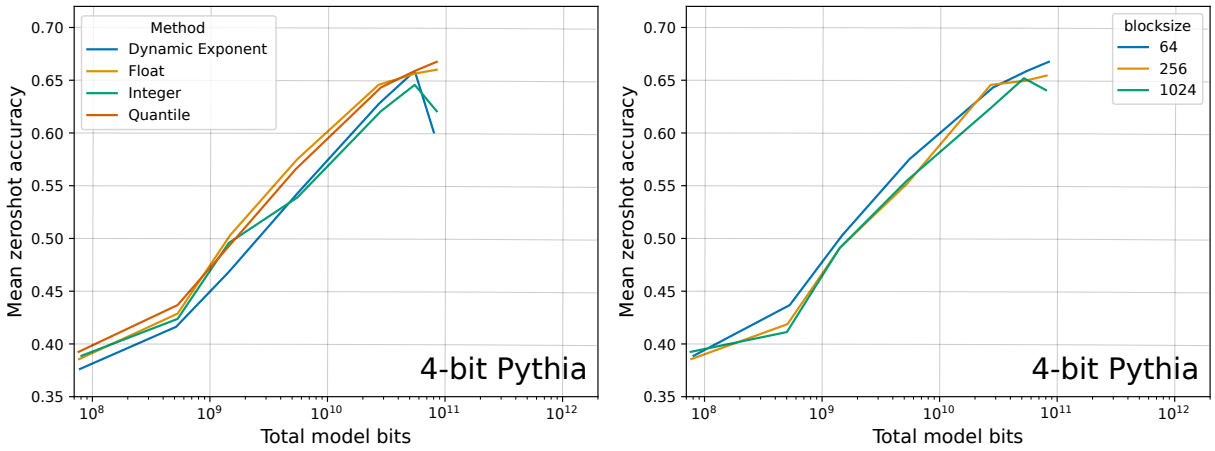

Figure 3. Bit-level mean zero-shot scaling laws for 4-bit Pythia models for different data types and block sizes. Block size and data types yield the largest improvements in bit-level scaling. For Pythia, a small block size adds only 0.25 bits per parameter but provides an improvement similar to going from 4-bit to 5-bit precision. In general, across all models, the float and quantile data types yield better scaling than int and dynamic exponent quantization.

be better or worse depending on the particular bit precision.

Outlier-dependent quantization improves stability, but not scaling. Finally, we noticed the 3-bit instabilities in OPT and Pythia, and we relate them to emergent outlier features (Dettmers et al., 2022a; Xiao et al., 2022). If we use proxy quantization to quantize the 2% most significant outlier dimensions to 16-bit instead of 3-bit, we can increase stability and overall scaling for 3-bit Pythia and OPT models. This is shown for OPT in Figure 4 (left). However, despite this improvement, 4-bit precision still provides better scaling. For 4-bit precision, outlier-dependent quantization has no scaling benefit, as shown in Figure 4 (right), which means 4-bit precision is optimal despite the considerable improvements for 3-bit precision when using proxy quantization. As such, it appears the outlier features do not require more than 4-bit precision weights to provide optimal bit-level scaling.

6. Related Work

Large language model quantization. The most closely related work is on large language model (LLM) quantization for models with more than a billion parameters. Compared to smaller models, LLM quantization poses some unique challenges, such as emergent outliers (Dettmers et al., 2022a; Zeng et al., 2022; Xiao et al., 2022) and optimized low-bit inference for LLMs (Frantar et al., 2022; Park et al., 2022; Yao et al., 2022). One major defining factor between approaches is zero-shot quantization methods that directly quantize a model without any additional information and one-shot quantization methods that need a mini-batch of data for quantization. While one-shot methods are more accurate, such as GPTQ, which optimizes the rounding during quantization via a mini-batch of data (Frantar et al., 2022), they are also more complex and may require hours of optimization before a model can be used. On the other hand, the advantage of zero-shot methods is that they can be used immediately, which makes them easy to use, but zero-shot quantization methods often fail at lower precisions.

Quantization methods. Another aspect related to our work are quantization methods which can be grouped into specific categories. For example, there are methods associated with blocking and grouping (Park et al., 2022; Wu et al., 2020; Jain et al., 2020; Nagel et al., 2019; Krishnamoorthi, 2018; Rusci et al., 2020), centering (Krishnamoorthi, 2018; Jacob et al., 2017), learned data types that are found through clustering (Gong et al., 2014; Han et al., 2015; Choi et al., 2016; Park et al., 2017), or direct codebook optimization (Rastegari et al., 2016; Hou et al., 2016; Leng et al., 2018; Zhang et al., 2018). While our work studies grouping and blocking, we only study one data type that groups similar weights through their quantiles of the entire input tensor (Dettmers et al., 2022b). While we do not study learned data types in depth, we are the first work that shows that these are critical for improving bit-level scaling for LLMs.

Scaling Laws for Inference. Early work in scaling laws highlighted the importance of studying how variables change with scale since scale is one of the best predictors of model performance (Kaplan et al., 2020; Rosenfeld et al., 2019; Hestness et al., 2017). Particularly, for inference, there has been work that studies scaling trends of zero-shot performance for 4-bit vs. 16-bit models (Zeng et al., 2022). We study precisions from 3 to 16-bit and disentangle the factors that improve scaling. Work by Pope et al. (2022) looks at scaling inference in a production setting where large batch sizes are common. While they only study quantization rudimentary, they disentangle factors that lead to better model FLOPS utilization (MFU). Since reducing the

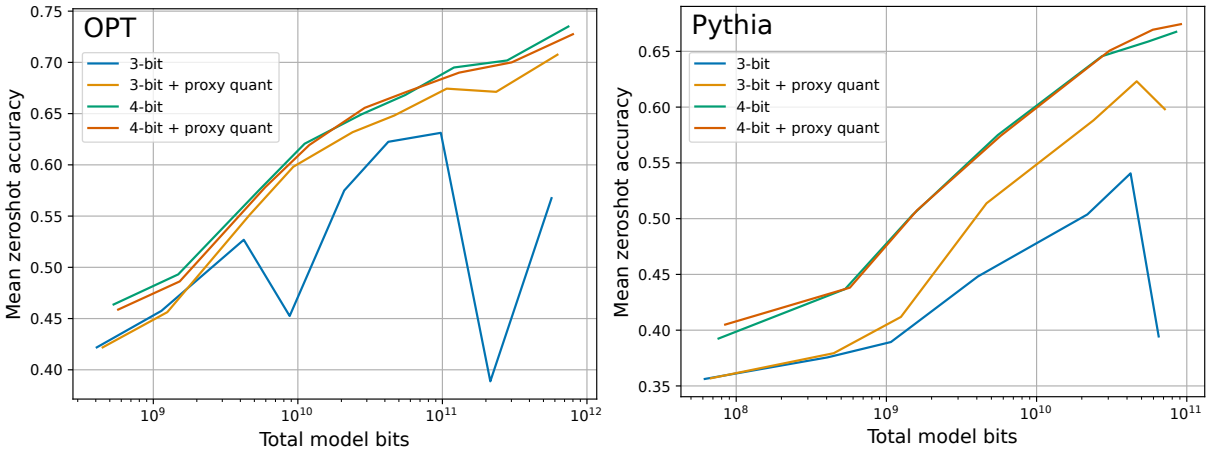

Figure 4. Bit-level scaling laws for outlier dependent quantization for OPT and Pythia. While proxy quantization removes instabilities and improves the 3-bit precision scaling of OPT and Pythia, it still scales worse than 4-bit precision. Proxy quantization is only useful for 3-bit precision weights. Proxy quantization is not useful for models that are relatively stable such as 3-bit BLOOM and GPT-2.

bit-precision of bits loaded leads to higher MFU, it is similar to our approach to studying bit-level scaling. The main difference is that we vary the bit-width of models and study small batch sizes that are common for consumers and small organizations.

7. Recommendations & Future Work

We make the following **recommendations**:

1. By default, use 4-bit quantization for LLM inference as it offers the total model bits and zero-shot accuracy trade-offs.
2. Use a block size of 128 or lower to stabilize 4-bit quantization and improve zero-shot performance.
3. Use a floating point or quantile quantization data type. In some cases, integer data types might be preferable to improve inference latency depending on the implementation and hardware support.

A case where a higher than 4-bit precision is desirable is when one works with a GPU with enough memory to hold a higher bit precision but not a larger model. For example, a 48 GB GPU has enough memory to use a 66B model in 5-bit precision but cannot fit a 175B model in 4-bit. Therefore, if maximal zero-shot accuracy is desired, 5-bit precision and a 66B model is preferable for this scenario.

Promising directions for future work. Our results highlight that 4-bit precision is currently bit-by-bit the most efficient precision, but we also show that 3-bit scaling can be significantly improved. As such, a promising research direction is to focus on low-bit precisions below 4-bit and improve their scaling trends. It has been shown that one-shot quantization methods, like GPTQ, which use an input sample for optimizing the quantization are more effective at

Table 1. WikiText-2 perplexity for 2-bit GPTQ and 3-bit Float with blocking. We can see that GPTQ is superior to 3-bit Float if blocking is used. As such, methods like GPTQ are a promising way to improve low-bit scaling. However, GPTQ requires blocking to provide good scaling (see Figure 5).

	WikiText-2 Perplexity	
Blocksize	2-bit GPTQ	3-bit Float
1024	**11.84**	13.26
256	**10.00**	10.38
64	**9.18**	9.99

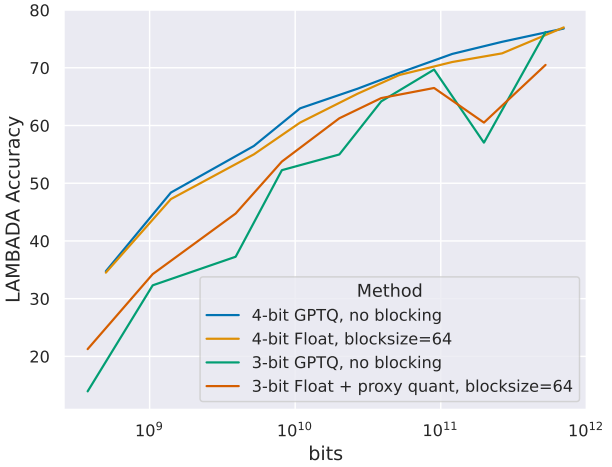

Figure 5. Bit-level scaling laws for LAMBADA zero-shot accuracy. We can see that GPTQ without blocking scales poorly at 3-bit. While one-shot quantization methods like GPTQ are superior to zero-shot methods like proxy quantization, other methods like blocking are required to make them bit-level efficient (see Table 1).

low-bit precisions (Frantar et al., 2022). Table 1 shows that 2-bit GPTQ with blocking yields better performance than zero-shot 3-bit Float. This highlights that one-shot methods are very promising for low-bit precisions.

On the other hand, Figure 5 also shows that 3-bit GPTQ without blocking scales worse compared to 3-bit Float with a blocksize of 64 and 4-bit GPTQ without blocking yields similar scaling compared to 4-bit Float with a blocksize of 64. From these results, it appears that the insights gained from our zero-shot scaling laws translate to one-shot quantization methods. This shows that zero-shot quantization research is well suited for disentangling and understanding individual factors in quantization scaling, while one-shot methods maximize performance. In contrast, current one-shot methods are expensive to study. For example, repeating our scaling experiments for GPTQ only for the OPT-175B and BLOOM-176B models would consume an estimated 5,120 GPU days of compute. Therefore, the combination of zero-shot quantization scaling insights and one-shot quantization methods may yield the best future methods.

One challenge unaddressed in our work is that the data type should also be able to be used efficiently on a hardware level. For example, while quantile quantization is the data type that shows the best scaling trends, it requires a small lookup table that is difficult to implement in a highly parallel setup where parallel memory access often leads to poor performance due to the serialization. This problem can be solved by designing new data types that are both bit-level scaling efficient and hardware efficient. A different approach to this problem would be hardware design: Can we design hardware to use data types such as quantile quantization efficiently?

Both block-size and outlier-dependent quantization improve the quantization precision of outliers. While outlier-dependent quantization does not offer improvements in scaling, it is reasonable that there are unknown quantization methods that help with outliers and improve scaling trends simultaneously. As such, another primary focus of future research should center around preserving outlier information while minimizing the use of additional bits.

8. Discussion & Limitations

While we ran more than 35,000 experiments, a main limitation is that we did not consider certain classes of quantization methods. For example, there are quantization methods where a data type is optimized with additional input data (Rastegari et al., 2016; Frantar et al., 2022) or from the weights of the model alone (Gong et al., 2014). Optimization from the weights alone is similar to quantile quantization which was the most effective data type in our study. As such, this hints that such quantization methods could improve scaling for inference and present a missed opportunity to be included in our study. However, our study is an essential step towards recognizing the importance of these methods for optimizing the model-bits-accuracy trade-off – a perspective that did not exist before.

Another limitation is the lack of optimized GPU implementations. It is unclear if other data types that rely on lookup tables can achieve significant speedups. However, efficient implementations for our Int/Float data types would be possible, and our results for other data types are still useful for the development of future data types, which yield strong scaling and efficient implementations.

While we only study the latency-optimal perspective indirectly through studying the model-bits-accuracy trade-off, a practical limitation of the latency-optimal perspective is that low-bit models with 16-bit inputs might be less latency efficient if such a model is deployed to be used by many users (Pope et al., 2022). Given a busy deployed API with thousands of requests per second, the large mini-batches would no longer fit into the cache. This means bit-level scaling laws would be increasingly unrelated to inference latency in this case. For such high throughput systems, scaling laws that model both low-bit weights *and* low-bit inputs are required to study optimal inference latency scaling. In short, our scaling laws are only valid for cases where the mini-batch does not fit into the L1 cache of the device, and beyond this, a new set of scaling laws is required.

A final limitation is that loading the weight matrix is only one part of inference latency which needs to be optimized to achieve fast inference. For example, without optimizations for the attention operations, multi-head attention can be a large chunk of the inference latency footprint (Jaszczur et al., 2021; Pope et al., 2022). However, the overall memory footprint of the model is still reduced, making large language models more easily usable when GPU memory is limited.

9. Conclusion

Here we presented a large-scale study of 35,000 zero-shot experiments on a wide variety of LLMs and parameter scales to analyze the scaling behavior and trade-offs between the number of parameters, quantization bit precision and zero-shot accuracy during inference. We find that 4-bit quantization is almost universally optimal to reduce the model bits and maximize zero-shot accuracy. We study the improvement of bit-level scaling behaviors and find that data types and block size are the most critical measures to improve bit-level scaling. Our analysis paves the way for the systematic study of inference scaling trends for LLMs.

Acknowledgements We would like to thank Aditya Kusupati, Gabriel Ilharco, Mitchell Wortsman, Andy Rock, and Ofir Press, for their helpful discussions and feedback.

References

Bisk, Y., Zellers, R., LeBras, R., Gao, J., and Choi, Y. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

Bondarenko, Y., Nagel, M., and Blankevoort, T. Understanding and overcoming the challenges of efficient transformer quantization. *arXiv preprint arXiv:2109.12948*, 2021.

Choi, Y., El-Khamy, M., and Lee, J. Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543*, 2016.

Dettmers, T. 8-bit approximations for parallelism in deep learning. *International Conference on Learning Representations (ICLR)*, 2016.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022a.

Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. 8-bit optimizers via block-wise quantization. *9th International Conference on Learning Representations, ICLR*, 2022b.

Dongarra, J. A not so simple matter of software. https://www.youtube.com/watch?v=cSO0Tc2w5Dg, November 2022.

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

Gao, J., He, D., Tan, X., Qin, T., Wang, L., and Liu, T.-Y. Representation degeneration problem in training natural language generation models. *arXiv preprint arXiv:1907.12009*, 2019.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gao, L., Tow, J., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., McDonell, K., Muennighoff, N., Phang, J., Reynolds, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

Gong, Y., Liu, L., Yang, M., and Bourdev, L. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M., Ali, M., Yang, Y., and Zhou, Y. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Hou, L., Yao, Q., and Kwok, J. T. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. arxiv e-prints, art. *arXiv preprint arXiv:1712.05877*, 2017.

Jain, S., Gural, A., Wu, M., and Dick, C. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. *Proceedings of Machine Learning and Systems*, 2:112–128, 2020.

Jaszczur, S., Chowdhery, A., Mohiuddin, A., Kaiser, L., Gajewski, W., Michalewski, H., and Kanerva, J. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907, 2021.

Jia, Z., Maggioni, M., Smith, J., and Scarpazza, D. P. Dissecting the nvidia turing t4 gpu via microbenchmarking. *arXiv preprint arXiv:1903.07486*, 2019.

Jin, Q., Ren, J., Zhuang, R., Hanumante, S., Li, Z., Chen, Z., Wang, Y., Yang, K., and Tulyakov, S. F8net: Fixed-point 8-bit only multiplication for network quantization. *arXiv preprint arXiv:2202.05239*, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kovaleva, O., Kulshreshtha, S., Rogers, A., and Rumshisky, A. Bert busters: Outlier dimensions that disrupt transformers. *arXiv preprint arXiv:2105.06990*, 2021.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Leng, C., Dou, Z., Li, H., Zhu, S., and Jin, R. Extremely low bit neural network: Squeeze the last bit out with admm. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Luo, Z., Kulmizev, A., and Mao, X. Positional artefacts propagate through masked language model embeddings. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5312–5327, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.413. URL https://aclanthology.org/2021.acl-long.413.

Micikevicius, P., Stosic, D., Burgess, N., Cornea, M., Dubey, P., Grisenthwaite, R., Ha, S., Heinecke, A., Judd, P., Kamalu, J., et al. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

Nagel, M., Baalen, M. v., Blankevoort, T., and Welling, M. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1325–1334, 2019.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, N. Q., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144.

Park, E., Ahn, J., and Yoo, S. Weighted-entropy-based quantization for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5456–5464, 2017.

Park, G., Park, B., Kwon, S. J., Kim, B., Lee, Y., and Lee, D. nuqmm: Quantized matmul for efficient inference of large-scale generative language models. *arXiv preprint arXiv:2206.09557*, 2022.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Levskaya, A., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. *arXiv preprint arXiv:2211.05102*, 2022.

Puccetti, G., Rogers, A., Drozd, A., and Dell'Orletta, F. Outliers dimensions that disrupt transformers are driven by frequency. *arXiv preprint arXiv:2205.11380*, 2022.

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

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pp. 525–542. Springer, 2016. doi: 10.1007/978-3-319-46493-0_32. URL https://doi.org/10.1007/978-3-319-46493-0_32.

Rosenfeld, J. S., Rosenfeld, A., Belinkov, Y., and Shavit, N. A constructive prediction of the generalization error across scales. *arXiv preprint arXiv:1909.12673*, 2019.

Rusci, M., Capotondi, A., and Benini, L. Memory-driven mixed low precision quantization for enabling deep network inference on microcontrollers. *Proceedings of Machine Learning and Systems*, 2:326–335, 2020.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, 2021. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Timkey, W. and van Schijndel, M. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. *arXiv preprint arXiv:2109.04404*, 2021.

Wei, X., Zhang, Y., Zhang, X., Gong, R., Zhang, S., Zhang, Q., Yu, F., and Liu, X. Outlier suppression: Pushing the limit of low-bit transformer language models. *arXiv preprint arXiv:2209.13325*, 2022.

Williams, S., Waterman, A., and Patterson, D. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

Wu, H., Judd, P., Zhang, X., Isaev, M., and Micikevicius, P. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

Xiao, G., Lin, J., Seznec, M., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*, 2022.

Yao, Z., Aminabadi, R. Y., Zhang, M., Wu, X., Li, C., and He, Y. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *arXiv preprint arXiv:2206.01861*, 2022.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1472. URL https://doi.org/10.18653/v1/p19-1472.

Zeng, A., Liu, X., Du, Z., Wang, Z., Lai, H., Ding, M., Yang, Z., Xu, Y., Zheng, W., Xia, X., et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

Zhang, D., Yang, J., Ye, D., and Hua, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 365–382, 2018.

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

A. Data Type Details

These sections provide the full details of all our data types that we used in our k-bit quantization experiments. First, we provide a general view of quantization that unifies data types under a common formalism so that data types can be more easily compared. We discuss the limitations of this view in our discussion section.

Quantization as a mapping from integers to values. While there are many ways to define quantization, we provide a general definition that unifies data types. We define quantization as a mapping from k-bit integers I to floating point values F in the range $[-1, 1]$. This definition has the advantage that a data type is fully specified by the set of floating point values F alone and the number of bits k in the set I.

More formally, we can describe k-bit quantization as a mapping from the set of k-bit integers I to the set F, that is, $\mathbf{Q}^{\text{map}} : I \mapsto F = [0, 2^k - 1] \mapsto F$. For example, the IEEE 32-bit floating point data type maps the indices $0...2^{32} - 1$ to the set S with domain [-3.4e38, +3.4e38]. Furthermore, to be able to compare data types more easily, we normalize the domain of the real values in set F to the range $[-1, 1]$ by dividing by the absolute maximum value of the set of possible values F. We use the following notation: $\mathbf{Q}_k^{\text{map}}(i) = q_i$, for example $\mathbf{Q}_8^{\text{uint}}(83) = 83/255 = 0.3255$ for an 8-bit unsigned integer data type. With this notation, the mapping index i represents the quantized value to be stored.

To quantize an arbitrary input, we normalize the input into the range $[-1, 1]$[1] and then do a binary search in set F to find the closest real value to the input and its associated mapping index i. Once the closest value is found, we store the quantized value as the mapped index. Formally, this can be described as:

$$\mathbf{T}_i^{Q_k^{\text{map}}} = \underset{j=0}{\overset{n=2^k}{\arg\min}} |\mathbf{Q}_k^{\text{map}}(j) - \mathbf{T}_i|, \tag{3}$$

where $\mathbf{T}$ is the normalized input tensor.

To dequantize the tensor $\mathbf{T}_i^{Q_k^{\text{map}}}$ back to a real-valued tensor $\mathbf{T}^F$, we perform a lookup:

$$\mathbf{T}_i^F = \mathbf{Q}^{\text{map}}(\mathbf{T}_i^Q) \cdot c, \tag{4}$$

where c is the normalization constant that normalized $\mathbf{T}$ into the range $[-1, 1]$.

To do computation, we dequantize the weight in the cache and perform a 16-bit floating point multiplication with the 16-bit input.

With this general definition, we can now define the following data types by defining their set of quantization values F, also called a codebook.

Integer data types. Integer quantization, also known as linear or uniform quantization, maps the integers to itself with an offset of 128 for a signed integer $\mathbf{Q}^{\text{int}} : I \mapsto F = [0, 2^k - 1] \mapsto [-(2^{k-1} - 1), 2^{k-1}]$. In practice, we truncate the set F to have an equal number of positive and negative values around zero. So, for example, for an Int8 data type, we have the values $[-127/c, 127/c]$ where $c = 127$ is the absolute maximum normalization constant.

Floating Point data types. Floating point data types are represented by a combination of exponent bits E (with base 2) and mantissa bits M (fraction). Since we use 3-8 bit precision in our work, we take the FP8 data type as a reference (Micikevicius et al., 2022). The only difference is that we do not allocate a value for NaN values. As such, the floating point data type we use is defined by the following equations:

[1]This range is used for storage and not computation.

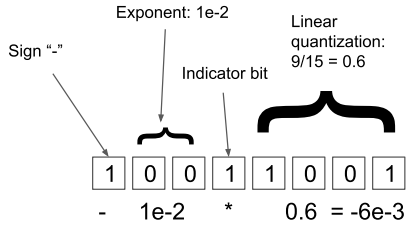

Figure 6. Schematic of dynamic exponent quantization.

$$\begin{gathered}\text{Subnormal numbers, if the exponent is zero:}\\ (-1)^{\text{signbit}} \times 2^{-\text{bias}} \times \sum_{i=1}^{\text{M}} b[\text{M}-i] \cdot 2^{-i}\\ \text{Normalized numbers, if the exponent is non-zero:}\\ (-1)^{\text{signbit}} \times 2^{-(\text{E}-1-\text{bias})} \times (1+\sum_{i=1}^{\text{M}} b[\text{M}-i] \cdot 2^{-i})\end{gathered}, \tag{5}$$

where bias=$2^{E-1}+1$, signbit=$\{0,1\}$, and $b[i]$ represents the binary mantissa bit value at index i in the bit-mask. To create the set F, we can iterate through all possible combinations of exponent, mantissa, and sign bits and apply the equations above. We evaluate different combinations of exponent and mantissa bits and found that a 2 to 3-bit exponent performs best in terms of zero-shot accuracy. We used a 3-bit exponent for 4 to 8-bit precision quantization, a 2-bit exponent for 3-bit quantization. As such our 4-bit and 5-bit Float results are slightly sub-optimal. For more on how different exponent bit combinations relate to performance see Appendix C.4.

Dynamic Exponent. Dynamic exponent data types (Dettmers, 2016; Dettmers et al., 2022b) use one bit for the sign, and the number of following zero bits represents the exponent with base 10. The first bit, which is one, serves as an indicator bit that separates the exponent and the unsigned linear fraction. The remaining bits represent an unsigned linear quantization. Alternatively, we can construct the set of floating point values F for the dynamic exponent data type by bisecting the interval $[0.1, 0.9]$ into n equal intervals where n is the number of fractional bits. We also define $00000000_2 = 0_{10}$ – which means we add the value of zero to F.

Quantile Quantization. The information-theoretically optimal data type will allocate an equal number of values to each quantization bin. For example, for a 1-bit quantization, this would be a quantization map with values F containing the lower and upper quartile so that 50% of values are allocated to each of the two bins. Quantile quantization (Dettmers, 2016) describes this information-theoretically optimal data type for a k-bit quantization. It can be defined as such:

$$q_i = \frac{Q_X\left(\frac{i}{2^k+1}\right) + Q_X\left(\frac{i+1}{2^k+1}\right)}{2}, \tag{6}$$

where Q_X is the quantile function which is the inverse of the cumulative distribution function $Q_X = F_X^{-1}$. We estimate the values of the quantile function by using the SRAM Quantiles algorithm (Dettmers et al., 2022b), which approximates the quantiles of an input tensor through the empirical cumulative distribution function of the tensor. The set F is defined by all q_i from $0 \ldots 2^k - 1$, and an additional 0 is added to the set F.

B. Further Negative Results: Distribution Centering

Here we use the notation introduced in Appendix A to define distribution centering and present negative results.

In **distribution centering**, we subtract the mean from the input tensor before quantization so that asymmetric distributions are approximately centered around zero.

$$
\begin{aligned}
m &= \sum_{i=0}^{n} \mathbf{T}_i / n \\
\mathbf{T}_i^{Q_k^{\text{map}}} &= \arg\min_{j=0}^{n=2^k} |\mathbf{Q}_k^{\text{map}}(j) - (\mathbf{T}_i - m)|
\end{aligned}
\tag{7}
$$

We add the mean as the final operation to dequantize a distribution-centered value.

$$
\mathbf{T}_i^F = \mathbf{Q}^{\text{map}}(\mathbf{T}_i^Q) \cdot c + m, \tag{8}
$$

Scaling results: Distribution centering is ineffective. While it has been shown that for activations that centering the distributions around zero can improve quantization errors due to asymmetric distributions (such as ReLU outputs), we find that distribution centering does not improve scaling for weight quantization in any scenario.

C. Detailed Scaling Results

C.1. Full Scaling Laws for 3-bit to 16-bit

Figure 7 shows bit-level scaling from 3 to 16-bit precision. Notable exceptions not found in the main paper scaling trends in these plots are as follow:

1. Pythia 5-bit as good as Pythoa 4-bit.
2. BLOOM and BLOOMZ show almost the same quantization behavior, indicating that fine-tuning an existing model does not change its quantization properties.

C.2. Details for scaling improvements at 4-bit precision

The main improvements that we see through quantization methods occur at 3-bit and 4-bit precision. Here we give full details for the 4-bit scenario. Figure 8 shows how bit-level scaling can be improved by using a smaller blocksize. Figure9 shows bit-level scaling for different data types. We did not perform a grid search on blocksize for 4-bits for BLOOM-176B and OPT-175B since we lacked the compute to complete these experiments.

C.3. No scaling improvements for 6 to 8-bit models through quantization methods

In this section we present data that shows that we cannot improve the bit-level scaling if we add quantization techniques that improve quantization precision if we use 6 to 8 bits per parameter. We hypothesize that this is because 6 to 8 bits per parameter is sufficient to model the weights with enough precision to not cause any major quantization precision problems. For example, any outliers in the weights might be sufficiently modelled by 6 to 8 bits and do not require additional techniques to prevent major quantization errors.

Figure 10 shows that we cannot improve bit-level scaling through data types when using 6-bit per paramters. We find similar results for 7 and 8-bit precision. Figure 11 shows a similar relationship for the block size variable.

C.4. Scaling of floating point exponent bits

It has been studied before how the standard deviation of the input relates to the optimal exponent bit configuration for the FP8 data type (Jin et al., 2022). However, this format assumes row-wise quantization without any blocking of the weight. Its also not studied how the overall performance is affected as we scale.

Here we present data on the scaling behavior of 3 to 8-bit precision quantization on transformer performance as we scale OPT from 125M to 176B. We use block-wise quantization with block wise 64 for the weight. Figure 12 shows that float data types with relatively many exponent bits do well if we have row-wise quantized inputs and block-wise quantized weights. A good heuristic is that for any bit-precision, the exponent bits should make up at least half the bits rounded up. This means, for 3, 4, 5, 6, 7, 8 bits we should use 2, 2, 3, 3, 4, 4 exponent bits. The only precision where this heuristic is not optimal is for 5-bit precision.

Table 2. Spearman rank correlation between the standard deviation of the weights of each hidden unit and the output layer magnitude. We see that the control has a rank correlation close to zero while the correlation between attention output projection standard deviation and first feedforward layer activation increases with scale from 0.55 to 0.776, showing a very strong correlation. This indicates that proxy quantization is effective at capturing outliers in the first FFN layer and moderately effective at capturing outliers in the attention output layer.

	Spearman rank correlation						
Layer / feature	350M	1.3B	2.7B	6.7B	13B	30B	66B
FFN1 / ReLU (control)	-0.05	-0.08	-0.07	0.00	0.08	0.06	0.05
KVQ / Attn output	0.12	0.20	0.23	0.29	0.21	0.28	0.41
Attn-output / FFN layer 1	0.55	0.66	0.69	0.73	0.72	0.74	0.78

C.5. Perplexity-based inference scaling laws

Since perplexity evaluation provides a continuous value with each token and zero-shot accuracy only a binary value, perplexity is the more reliable value to compare methods. In this section, we present data for evaluation on The Pile Common Crawl (Gao et al., 2020). We did not evaluate Pythia models since our codebase had a bug where evaluation crashed and only provide evaluation for BLOOM, BLOOM, GPT-2, and OPT. For OPT-175B we used less samples to evaluate since we did not have enough compute to produce these metrics before the conference deadline and this introduced noise which makes perplexity for OPT-175B worse than OPT-66B. In our graphs we limit the perplexity to 100 which indicates that the quantization was unstable and performed at random performance (perplexity > 100000). We also transform the perplexity to the cross entropy loss value to make curves more easily visibile. Figure 13 shows inference scaling laws for total bits, Figure 14 for data types, and Figure 15 for block size.

D. Analysis of Proxy Quantization: Correlation Between Standard deviation and Outliers

So provide further evidence that proxy quantization does the right thing, we provide a correlation analysis between the standard deviation of the outgoing weights – that is weights of each hidden unit – and the output magnitude. If the standard deviation is strongly related to outliers, the correlation should be large. Initial analysis with Pearson correlation reveals that there is a perfect correlation between standard deviation and output feature magnitudes for all layers $p = 0.00$. However, on further inspection, the standard deviation are not normally distributed with a few hidden units having much larger standard deviation and larger output features – which is as expected.

Because the Pearson correlation cofficient is not valid for non-normal residuals, we proceed to calculate the Spearman's rank correlation. This correlation underestimates the strength between individual standard-deviation-outlier pairs, but can measure if there is a *gradual* increase in outlier magnitude as the standard deviation increases.

As moted by Dettmers et al. (2022a) there are no outliers in the relu activation features after the first feedforward network (FFN) layer and we measure the correlation of this layer as a control. Results are shown in Table 2 and we can see that there is moderate to very large rank correlation between the standard deviation and outlier magnitude, indicating that proxy quantization works as expected. The control has a low correlation as expected.

E. Speedups Relative to 16-bit Float (FP16)

Throughout the paper we assume that the latency of inference is mostly dependent on the bits in the weight matrix. To test this claim we construct a roofline model (Williams et al., 2009) that is based on real matrix multiplication data as implemented by NVIDIA. A roofline model estimates the arithmetic intensity for a given size of matrix multiplication and thus estimates the processor is limited by memory bandwidth (loading the bits in the matrix) or by computation (multiplying and adding numbers). We benchmark memory bandwidth and tensor core utialization of 16-bit Float (FP16) matrix multiplications and construct a roofline model from these data.

Additionally, we implement some basic kernels for a batch size of 16. As of now, with the submission of the Camera Ready version for ICML, these kernels are not optimized and show lower performance than can be expected once they are optimized.

The results are shown in Figure 16. We can see that the roofline model indicates that large speedups of close to 4x are possible, but our unoptimized implementation do not yet reach these levels of speedup for a batch size of 16.

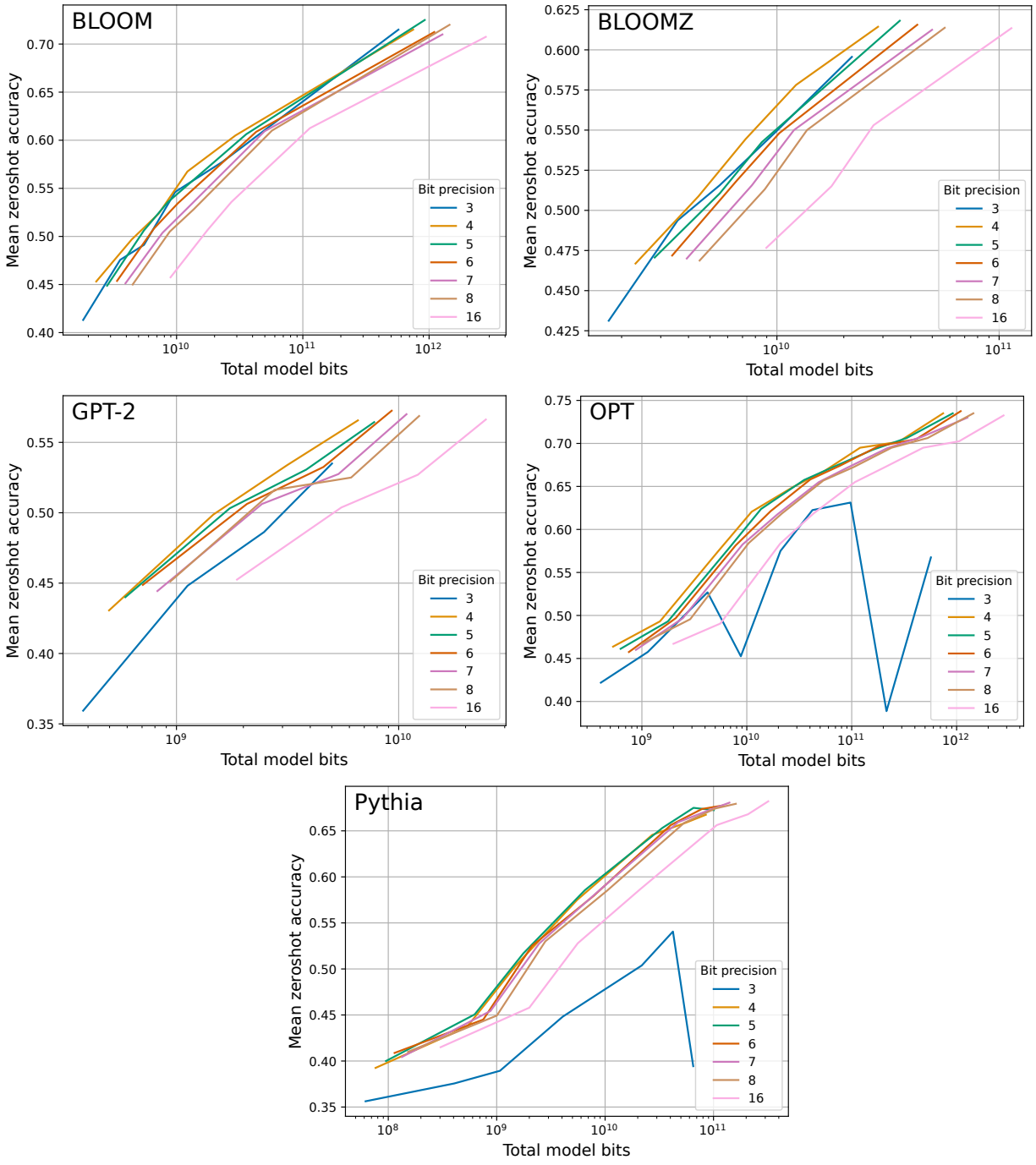

Figure 7. Bit-level scaling laws for mean zero-shot performance across Lambada, PiQA, Winogrande, and Hellaswag. 4-bit precision is optimal for all models at all scales with few exceptions. While lowering the precision generally improves scaling, this trend stops across all models at 3-bit precision where performance degrades. OPT and Pythia are unstable at 3-bit precision while GPT-2 and BLOOM remain stable. Stability is related to outlier features as explored in our analysis section.

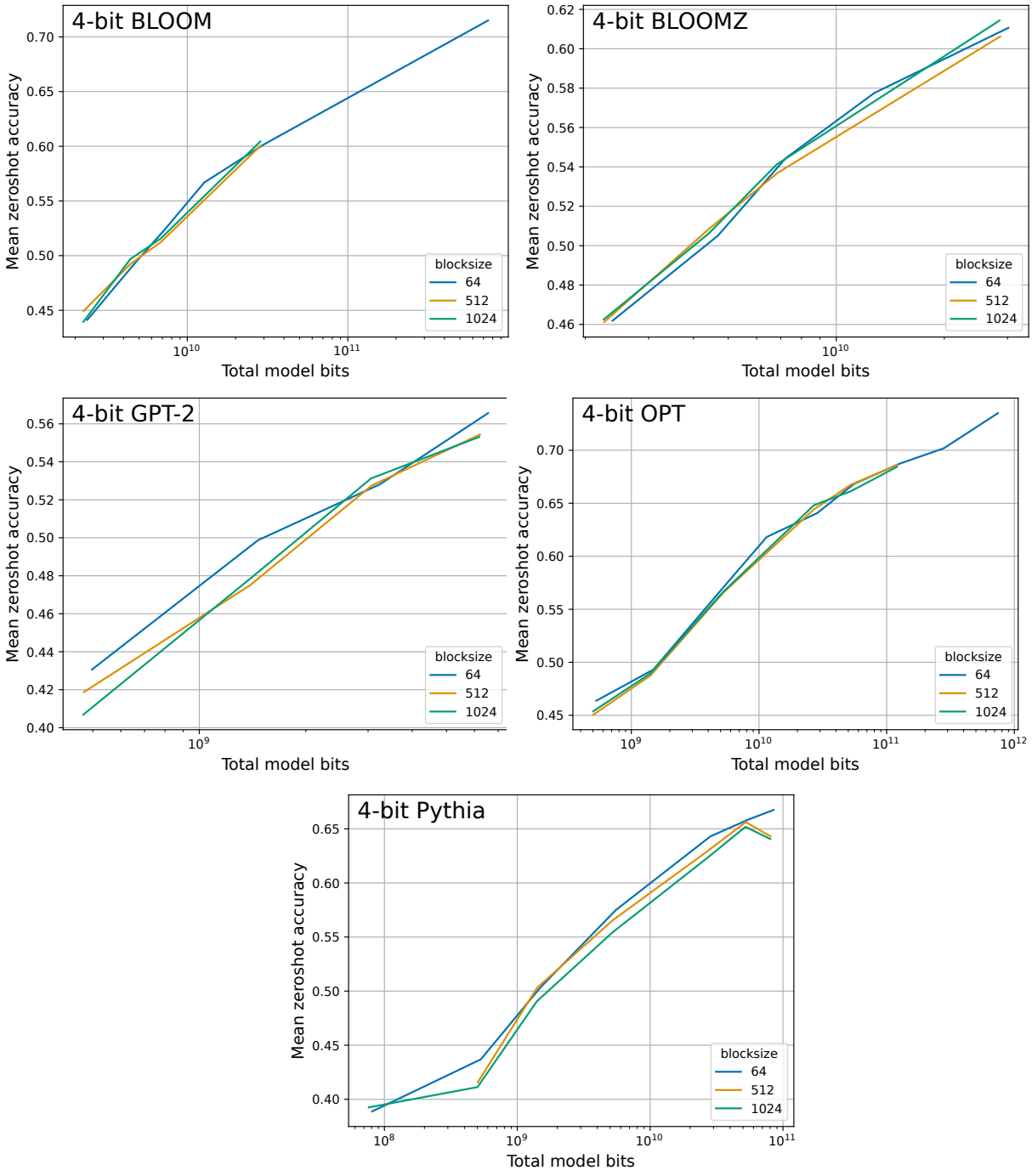

Figure 8. 4-bit scaling laws for mean zero-shot performance across Lambada, PiQA, Winogrande, and Hellaswag for different quantization data types. We see that the choice of block size improves bit-level scaling for most models at most scales. But this relationship is not always straightforward as in the case of OPT. We see more clear trends for 3-bit precision or when comparing methods using perplexity (not shown).

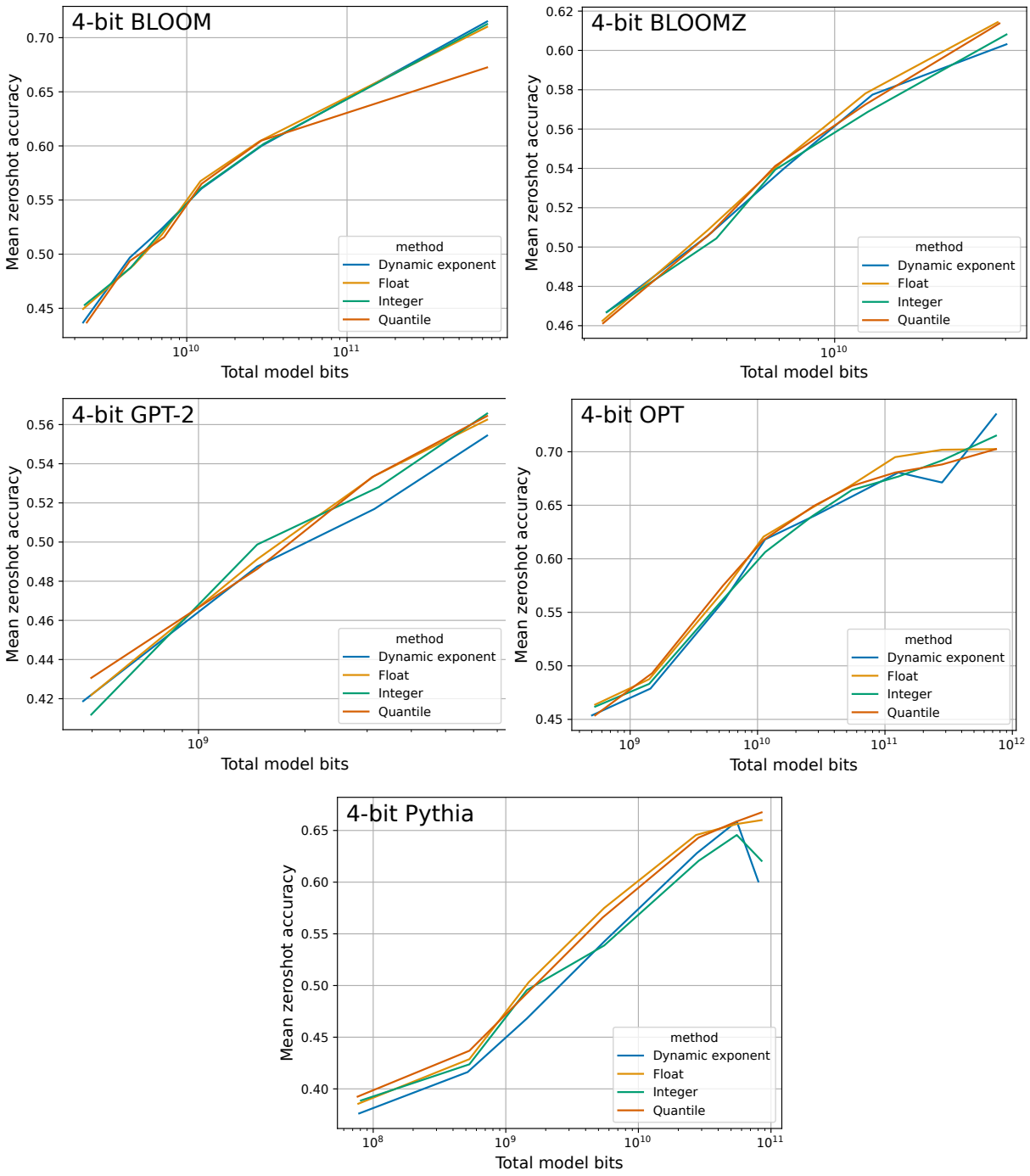

Figure 9. 4-bit scaling laws for mean zero-shot performance across Lambada, PiQA, Winogrande, and Hellaswag for different quantization data types. We see that the choice of data types improves bit-level scaling for most models at most scales with quantile quantization provided the best scaling on average.

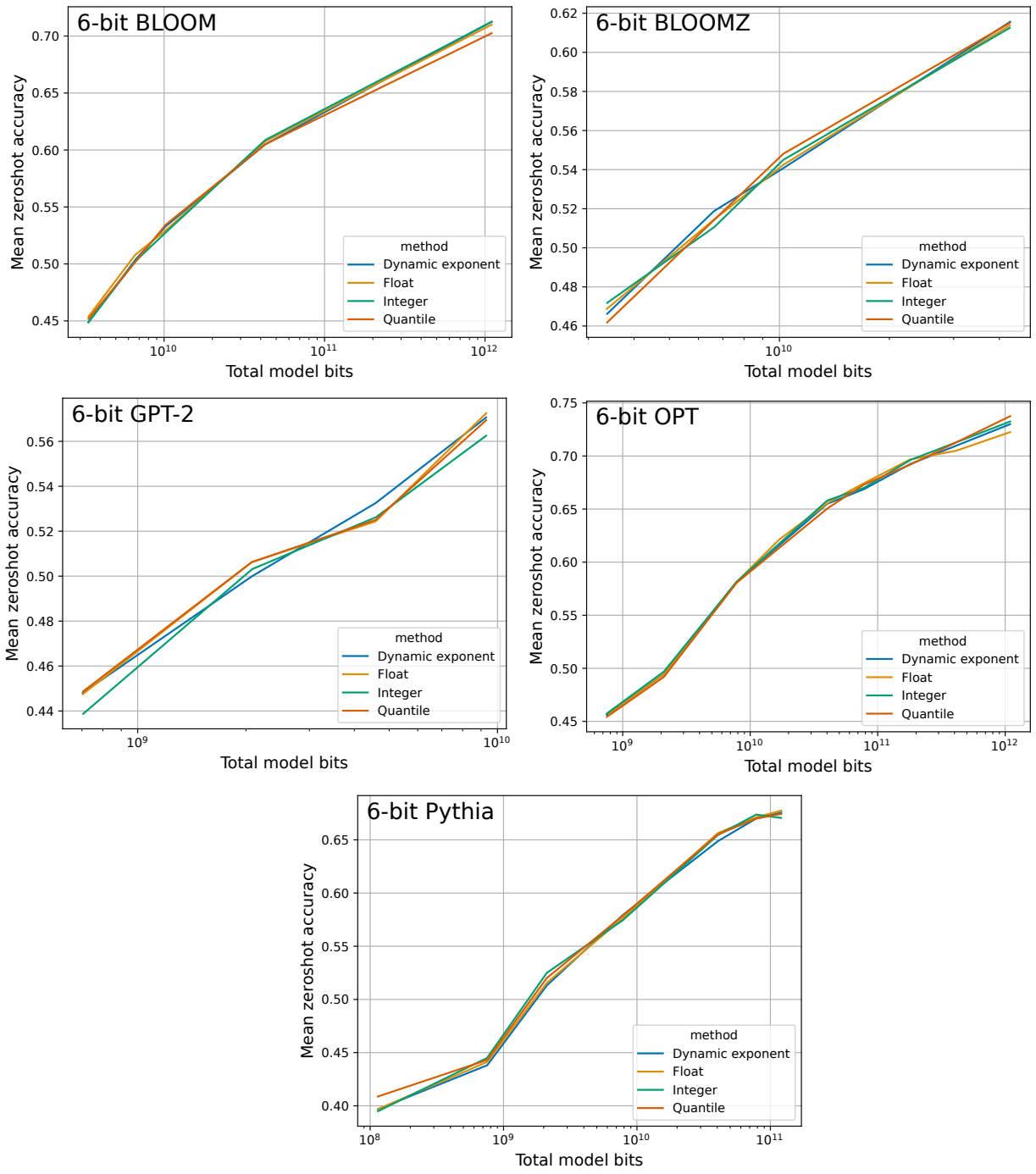

Figure 10. 6-bit scaling laws for mean zero-shot performance across Lambada, PiQA, Winogrande, and Hellaswag for different quantization data types. We see that the choice of data types does not affect scaling behavior at 6-bit precision. We results are similar tor 7 and 8-bit precision.

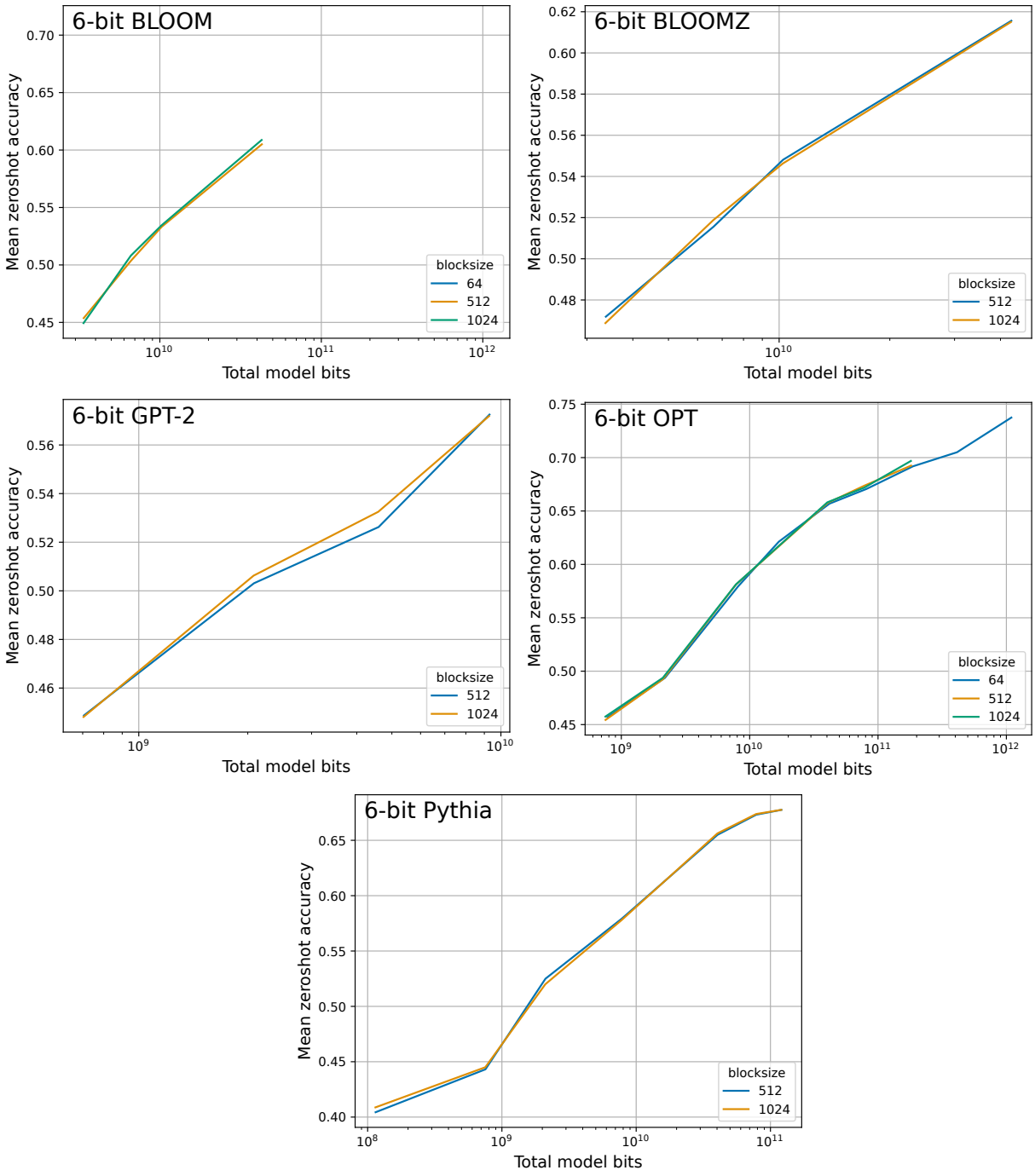

Figure 11. 6-bit scaling laws for mean zero-shot performance across Lambada, PiQA, Winogrande, and Hellaswag for different quantization data types. We see that the choice of the blocksize does not affect scaling behavior at 6-bit precision. We results are similar tor 7 and 8-bit precision.

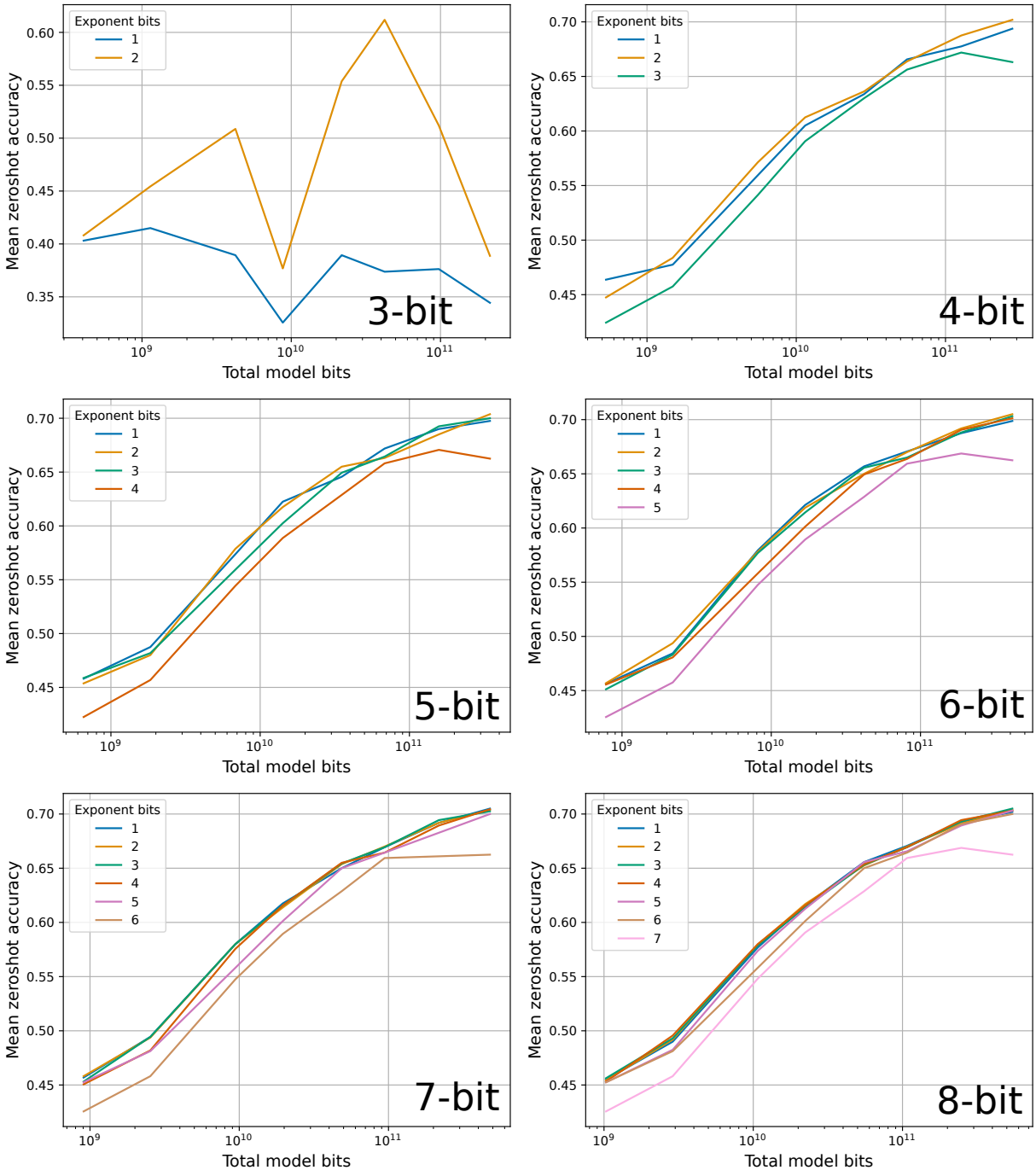

Figure 12. Inference scaling laws for mean zero-shot performance across Lambada, PiQA, Winogrande, and Hellaswag for different exponent bits for different bit precisions for the float data type and block-wise quantized weight. We see that a 2-bit exponent is the only one that performs well across all precisions.

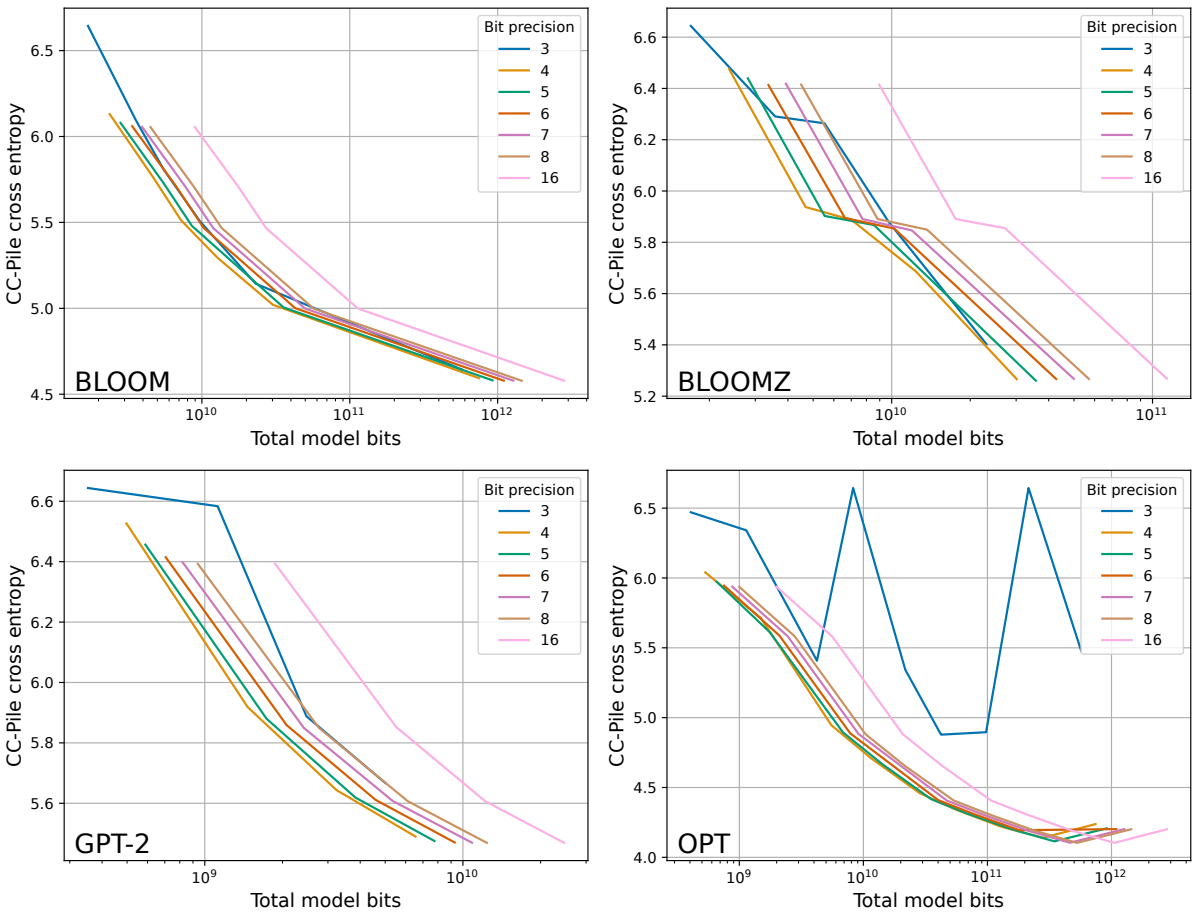

Figure 13. Inference scaling laws for cross entropy loss on CC-Pile for different bit precisions. We see that 4-bit quantization is bit-level optimal across all models.

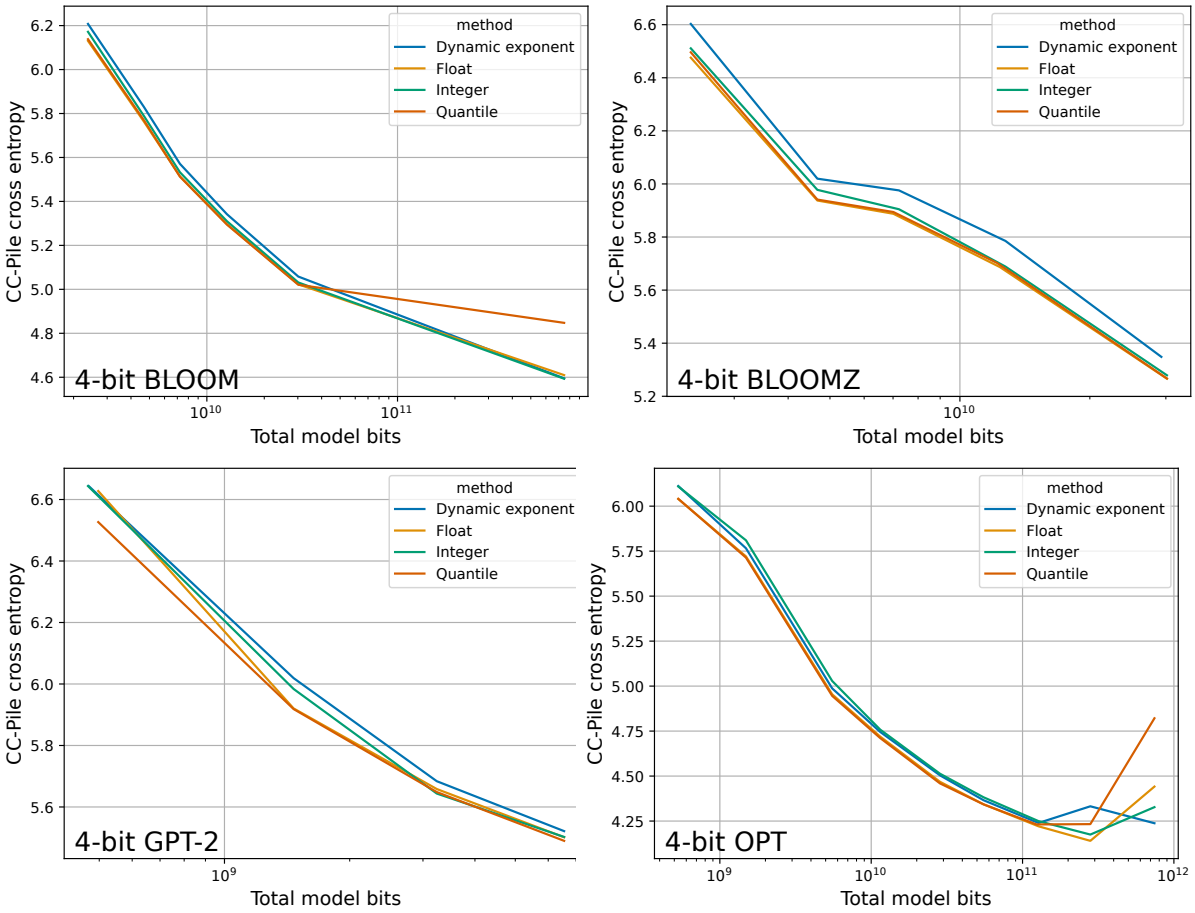

Figure 14. Scaling laws for cross entropy loss on CC-Pile for different quantization data types. We see that quantile quantization is the best data type on average.

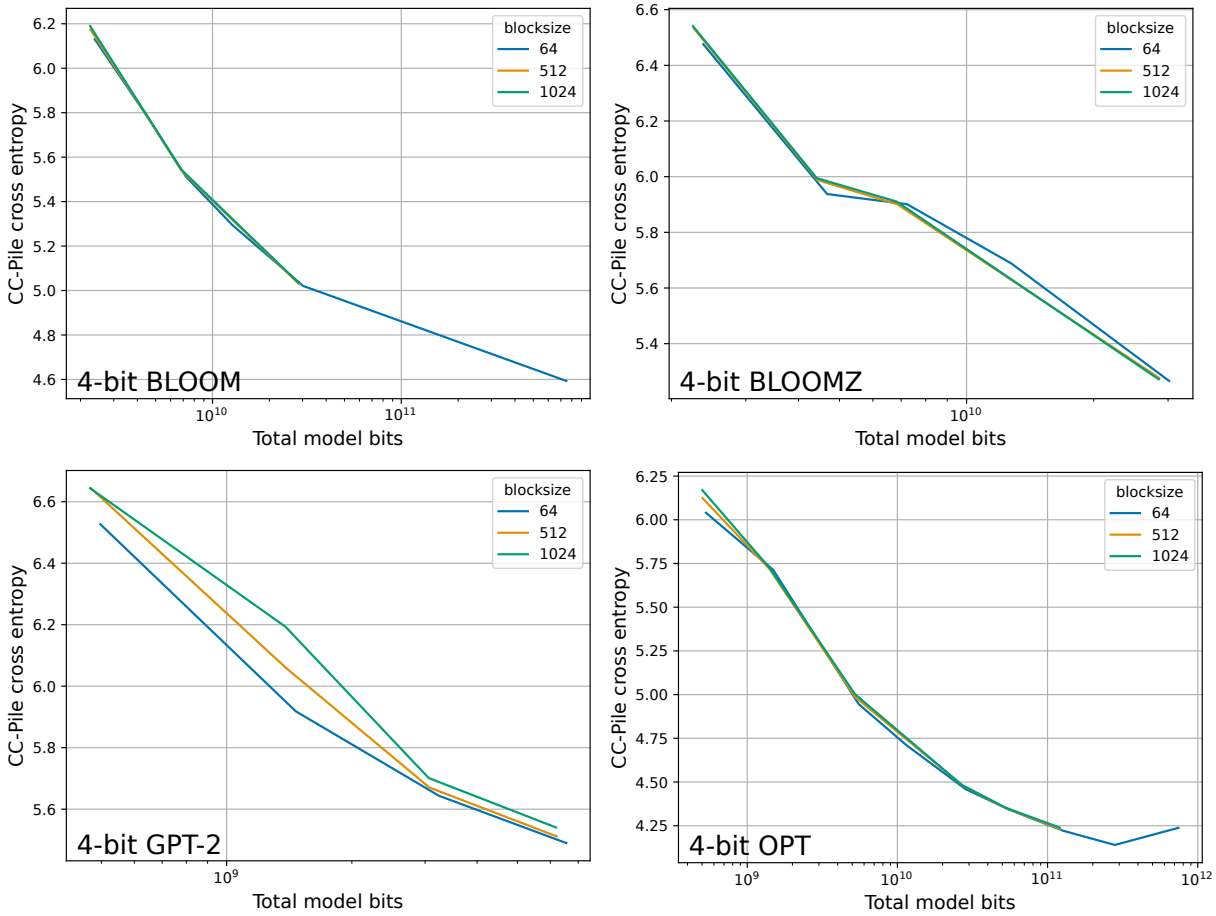

Figure 15. Inference scaling laws for cross entropy loss on CC-Pile for different block sizes. We see that a smaller block size is better than a larger one. While the difference is small across many models, small block sizes are consistently better, except for BLOOMZ.

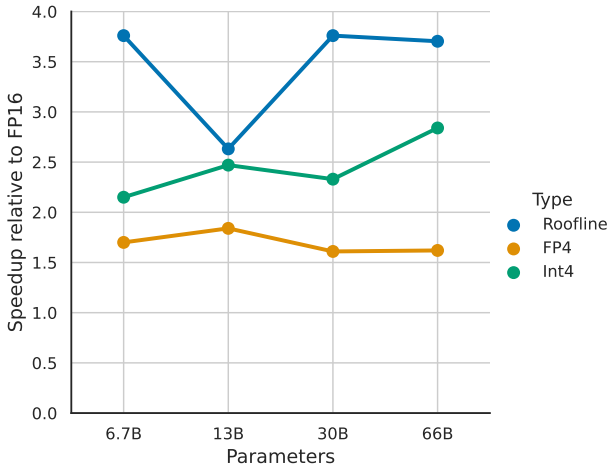

Figure 16. Speedups of unoptimized CUDA kernels and a roofline model of 4-bit matrix multplication compared to 16-bit Float (FP16) matrix multiplication throughput for LLM inference with batch size 16. We can see that the roofline model achieves close to 4x speedup. The unoptimized implementations do not quite reach these speedups in particular the FP4 kernel, which currently is bound by instruction throughput.