LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale

Tim Dettmers[λ] [*] & $MikeLewis^{\dagger}$ & $YounesBelkada^{\S\mp}$ & $LukeZettlemoyer^{\dagger\lambda}$
University of Washington[λ] Facebook AI Research[†] Hugging Face[§] ENS Paris-Saclay[∓]

June 28, 2025

Abstract

Large language models have been widely adopted but require significant GPU memory for inference. We develop a procedure for Int8 matrix multiplication for feed-forward and attention projection layers in transformers, which cut the memory needed for inference by half while retaining full precision performance. With our method, a 175B parameter 16/32-bit checkpoint can be loaded, converted to Int8, and used immediately without performance degradation. This is made possible by understanding and working around properties of highly systematic emergent features in transformer language models that dominate attention and transformer predictive performance. To cope with these features, we develop a two-part quantization procedure, LLM.int8(). We first use vector-wise quantization with separate normalization constants for each inner product in the matrix multiplication, to quantize most of the features. However, for the emergent outliers, we also include a new mixed-precision decomposition scheme, which isolates the outlier feature dimensions into a 16-bit matrix multiplication while still more than 99.9% of values are multiplied in 8-bit. Using LLM.int8(), we show empirically it is possible to perform inference in LLMs with up to 175B parameters without any performance degradation. This result makes such models much more accessible, for example making it possible to use OPT-175B/BLOOM on a single server with consumer GPUs.We open source our software.
大型语言模型已被广泛采用，但需要大量 GPU 内存进行推理。我们为 Transformer 中的前馈和注意力投影层开发了一种 Int8 矩阵乘法程序，将推理所需的内存减少了一半，同时保持了全精度性能。使用我们的方法，可以加载 175B 参数的 16/32 位检查点，将其转换为 Int8，并立即使用而不会降低性能。这是通过理解和解决 Transformer 语言模型中主导注意力和 Transformer 预测性能的高度系统化的新兴特征的属性而实现的。为了应对这些特征，我们开发了一个两部分量化程序 LLM.int8()。我们首先使用矢量量化，对矩阵乘法中的每个内积使用单独的规范化常数，以量化大多数特征。然而，对于出现的异常值，我们还引入了一种新的混合精度分解方案，该方案将异常值特征维度隔离为 16 位矩阵乘法，同时仍有超过 99.9% 的值以 8 位相乘。使用 LLM.int8()，我们通过经验证明，可以在具有多达 175B 个参数的 LLM 中执行推理而不会降低性能。这一结果使此类模型更易于访问，例如，可以在具有消费级 GPU 的单个服务器上使用 OPT-175B/BLOOM。我们开源我们的软件。

1 Introduction

Large pretrained language models are widely adopted in NLP [48, 39, 4, 59] but require significant memory for inference. For large transformer language models at and beyond 6.7B parameters, the feed-forward and attention projection layers and their matrix multiplication operations are responsible for 95%[1] of consumed parameters and 65-85% of all computation [23]. One way to reduce the size of the parameters is to quantize them to less bits and use low-bit-precision matrix multiplication. With this goal in mind, 8-bit quantization methods for transformers have been developed [6, 28, 56, 44]. While these methods reduce memory use, they degrade performance, usually require tuning quantization further after training, and have only been studied for models with less than 350M parameters. Degradation-free quantization up to 350M parameters is poorly understood, and multi-billion parameter quantization remains an open challenge.
大型预训练语言模型在 NLP [48, 39, 4, 59] 中被广泛采用，但需要大量内存进行推理。对于参数超过 6.7B 的大型 Transformer 语言模型，前馈和注意投影层及其矩阵乘法运算占消耗参数的 95% 和所有计算的 65-85% [23]。减小参数大小的一种方法是将它们量化为更少的位并使用低位精度矩阵乘法。出于这一目标，已经开发了 Transformer 的 8 位量化方法 [6, 28, 56, 44]。虽然这些方法减少了内存使用，但它们会降低性能，通常需要在训练后进一步调整量化，并且仅针对参数少于 350M 的模型进行了研究。高达 350M 参数的无降级量化尚不明确，数十亿参数量化仍然是一个悬而未决的挑战。

[*]Majority of research done as a visiting researcher at Facebook AI Research.
[1]Other parameters come mostly from the embedding layer. A tiny amount comes from norms and biases.
其他参数主要来自嵌入层。极少量来自规范和偏差。

In this paper, we present the first multi-billion-scale Int8 quantization procedure for transformers that does not incur any performance degradation. Our procedure makes it possible to load a 175B parameter transformer with 16 or 32-bit weights, convert the feed-forward and attention projection layers to 8-bit, and use the resulting model immediately for inference without any performance degradation. We achieve this result by solving two key challenges: the need for higher quantization precision at scales beyond 1B parameters and the need to explicitly represent the sparse but systematic large magnitude outlier features that ruin quantization precision once they emerge in all transformer layers starting at scales of 6.7B parameters. This loss of precision is reflected in C4 evaluation perplexity (Section 3) as well as zeroshot accuracy as soon as these outlier features emerge, as shown in Figure 1.
在本文中，我们介绍了第一个不会导致任何性能下降的数十亿级 Int8 Transformer 量化程序。我们的程序可以加载具有 16 或 32 位权重的 175B 参数 Transformer，将前馈和注意力投影层转换为 8 位，并立即使用生成的模型进行推理，而不会导致任何性能下降。我们通过解决两个关键挑战实现了这一结果：需要在超过 1B 参数的规模上提高量化精度，以及需要明确表示稀疏但系统性的大幅度异常特征，这些特征一旦出现在 6.7B 参数的规模开始的 all Transformer 层中，就会破坏量化精度。一旦这些异常特征出现，这种精度损失就会反映在 C4 评估困惑度（第 3 节）以及零样本准确度中，如图 1 所示。

We show that with the first part of our method, vector-wise quantization, it is possible to retain performance at scales up to 2.7B parameters. For vector-wise quantization, matrix multiplication can be seen as a sequence of independent inner products of row and column vectors. As such, we can use a separate quantization normalization constant for each inner product to improve quantization precision. We can recover the output of the matrix multiplication by denormalizing by the outer product of column and row normalization constants before we perform the next operation.
我们表明，使用我们方法的第一部分，即矢量量化，可以在高达 2.7B 参数的规模下保持性能。对于矢量量化，矩阵乘法可以看作是行和列向量的独立内积序列。因此，我们可以为每个内积使用单独的量化归一化常数来提高量化精度。在执行下一个操作之前，我们可以通过使用列和行归一化常数的外积进行反归一化来恢复矩阵乘法的输出。

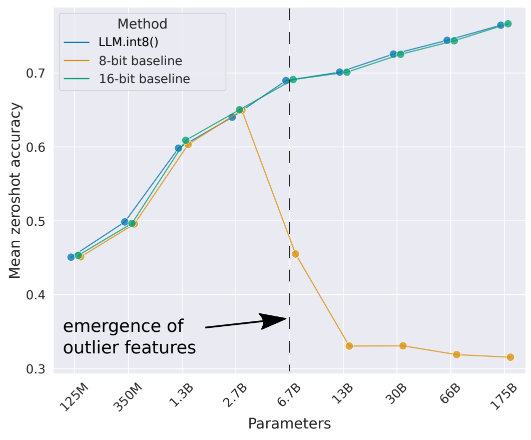

Figure 1: OPT model mean zeroshot accuracy for WinoGrande, HellaSwag, PIQA, and LAMBADA datasets. Shown is the 16-bit baseline, the most precise previous 8-bit quantization method as a baseline, and our new 8-bit quantization method, LLM.int8(). We can see once systematic outliers occur at a scale of 6.7B parameters, regular quantization methods fail, while LLM.int8() maintains 16-bit accuracy.
OPT 模型对 WinoGrande、HellaSwag、PIQA 和 LAMBADA 数据集的平均零样本准确率。图中显示的是 16 位基线、之前最精确的 8 位量化方法作为基线，以及我们新的 8 位量化方法 LLM.int8()。我们可以看到，一旦在 6.7B 参数规模上出现系统性异常值，常规量化方法就会失效，而 LLM.int8() 则保持了 16 位准确率。

To scale beyond 6.7B parameters without performance degradation, it is critical to understand the emergence of extreme outliers in the feature dimensions of the hidden states during inference. To this end, we provide a new descriptive analysis which shows that large features with magnitudes up to 20x larger than in other dimensions first appear in about 25% of all transformer layers and then gradually spread to other layers as we scale transformers to 6B parameters. At around 6.7B parameters, a phase shift occurs, and all transformer layers and 75% of all sequence dimensions are affected by extreme magnitude features.
为了在不降低性能的情况下扩展超过 6.7B 参数，了解在推理过程中隐藏状态的特征维度中极端异常值的出现至关重要。为此，我们提供了一个新的描述性分析，该分析表明，幅度比其他维度大 20 倍的大特征首先出现在所有 Transformer 层的约 25% 中，然后随着我们将 Transformer 扩展到 6B 参数而逐渐扩散到其他层。在约 6.7B 参数时，会发生相移，并且所有 Transformer 层和 75% 的所有序列维度都受到极端幅度特征的影响。

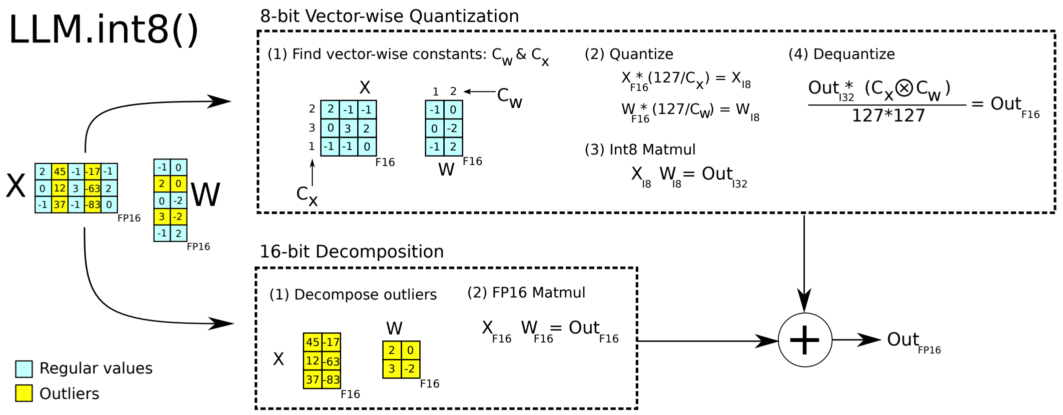

Figure 2: Schematic of LLM.int8(). Given 16-bit floating-point inputs X_{f16} and weights W_{f16}, the features and weights are decomposed into sub-matrices of large magnitude features and other values. The outlier feature matrices are multiplied in 16-bit. All other values are multiplied in 8-bit. We perform 8-bit vector-wise multiplication by scaling by row and column-wise absolute maximum of C_x and C_w and then quantizing the outputs to Int8. The Int32 matrix multiplication outputs Out_{i32} are dequantization by the outer product of the normalization constants $C_x \otimes C_w$. Finally, both outlier and regular outputs are accumulated in 16-bit floating point outputs.
LLM.int8() 的示意图。给定 16 位浮点输入 X_{f16} 和权重 W_{f16}，特征和权重被分解为大幅度特征和其他值的子矩阵。异常特征矩阵以 16 位相乘。所有其他值以 8 位相乘。我们通过按行和列的 C_x 和 C_w 的绝对最大值缩放来执行 8 位向量乘法，然后将输出量化为 Int8。Int32 矩阵乘法输出 Out_{i32} 由规范化常数 $C_x \otimes C_w$ 的外积进行反量化。最后，异常值和常规输出都累积在 16 位浮点输出中。

These outliers are highly systematic: at the 6.7B scale, 150,000 outliers occur per sequence, but they are concentrated in only 6 feature dimensions across the entire transformer. Setting these outlier feature dimensions to zero decreases top-1 attention softmax probability mass by more than 20% and degrades validation perplexity by 600-1000% despite them only making up about 0.1% of all input features. In contrast, removing the same amount of random features decreases the probability by a maximum of 0.3% and degrades perplexity by about 0.1%.
这些异常值具有高度系统性：在 6.7B 规模下，每个序列会出现 150,000 个异常值，但它们仅集中在整个 Transformer 的 6 个特征维度中。将这些异常特征维度设置为零会使 top-1 注意力 softmax 概率质量降低 20% 以上，并使验证困惑度降低 600-1000%，尽管它们仅占所有输入特征的 0.1% 左右。相比之下，删除相同数量的随机特征最多会使概率降低 0.3%，并使困惑度降低约 0.1%。

To support effective quantization with such extreme outliers, we develop mixed-precision decomposition, the second part of our method. We perform 16-bit matrix multiplication for the outlier feature dimensions and 8-bit matrix multiplication for the other 99.9% of the dimensions. We name the combination of vector-wise quantization and mixed precision decomposition, LLM.int8(). We show that by using LLM.int8(), we can perform inference in LLMs with up to 175B parameters without any performance degradation. Our method not only provides new insights into the effects of these outliers on model performance but also makes it possible for the first time to use very large models, for example, OPT-175B/BLOOM, on a single server with consumer GPUs. While our work focuses on making large language models accessible without degradation, we also show in Appendix D that we maintain end-to-end inference runtime performance for large models, such as BLOOM-176B and provide modest matrix multiplication speedups for GPT-3 models of size 6.7B parameters or larger. We open-source our software[2] and release a Hugging Face Transformers [52] integration making our method available to all hosted Hugging Face Models that have linear layers.
为了支持对这种极端异常值进行有效量化，我们开发了混合精度分解，这是我们方法的第二部分。我们对异常值特征维度执行 16 位矩阵乘法，对其他 99.9% 的维度执行 8 位矩阵乘法。我们将向量量化和混合精度分解的组合命名为 LLM.int8()。我们表明，通过使用 LLM.int8()，我们可以在具有多达 175B 个参数的 LLM 中执行推理，而不会降低性能。我们的方法不仅提供了有关这些异常值对模型性能的影响的新见解，而且还首次使在具有消费者 GPU 的单个服务器上使用非常大的模型（例如 OPT-175B/BLOOM）成为可能。虽然我们的工作重点是让大型语言模型在不降低性能的情况下实现可访问性，但我们还在附录 D 中展示了我们为大型模型（例如 BLOOM-176B）保持端到端推理运行时性能，并为大小为 6.7B 参数或更大的 GPT-3 模型提供适度的矩阵乘法加速。我们开源了我们的软件, 并发布了 Hugging Face Transformers [52] 集成，使我们的方法可用于所有具有线性层的托管 Hugging Face 模型。

[2]https://github.com/TimDettmers/bitsandbytes

2 Background

In this work, push quantization techniques to their breaking point by scaling transformer models. We are interested in two questions: at which scale and why do quantization techniques fail and how does this related to quantization precision? To answer these questions we study high-precision asymmetric quantization (zeropoint quantization) and symmetric quantization (absolute maximum quantization). While zeropoint quantization offers high precision by using the full bit-range of the datatype, it is rarely used due to practical constraints. Absolute maximum quantization is the most commonly used technique.
在这项工作中，通过扩展 transformer 模型将量化技术推向极限。我们感兴趣的两个问题是：量化技术在什么规模下会失败以及失败的原因是什么，以及这与量化精度有何关系？为了回答这些问题，我们研究了高精度非对称量化（零点量化）和对称量化（绝对最大量化）。虽然零点量化通过使用数据类型的全位范围来提供高精度，但由于实际限制，它很少使用。绝对最大量化是最常用的技术。

2.1 8-bit Data Types and Quantization

Absmax quantization scales inputs into the 8-bit range $[-127, 127]$ by multiplying with $s_{x_{f16}}$ which is 127 divided by the absolute maximum of the entire tensor. This is equivalent to dividing by the infinity norm and multiplying by 127. As such, for an FP16 input matrix $\mathrm{X}_{f16} \in \mathbb{R}^{s \times h}$ Int8 absmax quantization is given by:
绝对最大量化将输入缩放到 8 位范围 $[-127, 127]$，方法是乘以 $s_{x_{f16}}$，即 127 除以整个张量的绝对最大值。这相当于除以无穷大范数并乘以 127。因此，对于 FP16 输入矩阵 $\mathrm{X}_{f16} \in \mathbb{R}^{s \times h}$，Int8 绝对最大量化由以下公式给出：

$$\mathrm{X}_{i8} = \left\lfloor \frac{127 \cdot \mathrm{X}_{f16}}{\max\limits_{ij}(|\mathrm{X}_{f16_{ij}}|)} \right\rceil = \left\lfloor \frac{127}{\|\mathrm{X}_{f16}\|_{\infty}} \mathrm{X}_{f16} \right\rceil = \left\lfloor s_{x_{f16}} \mathrm{X}_{f16} \right\rceil,$$

where $\lfloor \rceil$ indicates rounding to the nearest integer.
其中 $\lfloor \rceil$ 表示四舍五入到最接近的整数。

Zeropoint quantization shifts the input distribution into the full range $[-127, 127]$ by scaling with the normalized dynamic range nd_x and then shifting by the zeropoint zp_x. With this affine transformation, any input tensors will use all bits of the data type, thus reducing the quantization error for asymmetric distributions. For example, for ReLU outputs, in absmax quantization all values in $[-127, 0)$ go unused, whereas in zeropoint quantization the full $[-127, 127]$ range is used. Zeropoint quantization is given by the following equations:
零点量化通过缩放归一化动态范围 nd_x，然后移动零点 zp_x，将输入分布移至完整范围 $[-127, 127]$。通过这种仿射变换，任何输入张量都将使用数据类型的所有位，从而减少非对称分布的量化误差。例如，对于 ReLU 输出，在绝对量化中，$[-127, 0)$ 中的所有值都未使用，而在零点量化中，则使用完整范围 $[-127, 127]$。零点量化由以下公式给出：

$$nd_{x_{f16}} = \frac{2 \cdot 127}{\max\limits_{ij}(\mathrm{X}^{ij}_{f16}) - \min\limits_{ij}(\mathrm{X}^{ij}_{f16})} \tag{1}$$

$$zp_{x_{i16}} = \left\lfloor \mathrm{X}_{f16} \cdot \min_{ij}(\mathrm{X}^{ij}_{f16}) \right\rceil \tag{2}$$

$$\mathrm{X}_{i8} = \left\lfloor nd_{x_{f16}} \mathrm{X}_{f16} \right\rceil \tag{3}$$

To use zeropoint quantization in an operation we feed both the tensor X_{i8} and the zeropoint $zp_{x_{i16}}$ into a special instruction[3] which adds $zp_{x_{i16}}$ to each element of X_{i8} before performing a 16-bit integer operation. For example, to multiply two zeropoint quantized numbers A_{i8} and B_{i8} along with their zeropoints $zp_{a_{i16}}$ and $zp_{b_{i16}}$ we calculate:
要在运算中使用零点量化，我们将张量 X_{i8} 和零点 $zp_{x_{i16}}$ 都输入到特殊指令中，该指令在执行 16 位整数运算之前将 $zp_{x_{i16}}$ 添加到 X_{i8} 的每个元素中。例如，要将两个零点量化数 A_{i8} 和 B_{i8} 与其零点 $zp_{a_{i16}}$ 和 $zp_{b_{i16}}$ 相乘，我们计算：

$$C_{i32} = \mathrm{multiply}_{i16}(A_{zp_{a_{i16}}}, B_{zp_{b_{i16}}}) = (A_{i8} + zp_{a_{i16}})(B_{i8} + zp_{b_{i16}}) \tag{4}$$

[3] https://www.felixcloutier.com/x86/pmaddubsw

where unrolling is required if the instruction multiply$_{i16}$ is not available such as on GPUs or TPUs:
如果指令 multiply$_{i16}$ 不可用（例如在 GPU 或 TPU 上），则需要展开：

$$C_{i32} = A_{i8}B_{i8} + A_{i8}zp_{b_{i16}} + B_{i8}zp_{a_{i16}} + zp_{a_{i16}}zp_{b_{i16}}, \tag{5}$$

where $A_{i8}B_{i8}$ is computed with Int8 precision while the rest is computed in Int16/32 precision. As such, zeropoint quantization can be slow if the multiply$_{i16}$ instruction is not available. In both cases, the outputs are accumulated as a 32-bit integer C_{i32}. To dequantize C_{i32}, we divide by the scaling constants $nd_{a_{f16}}$ and $nd_{b_{f16}}$.
其中 $A_{i8}B_{i8}$ 以 Int8 精度计算，其余则以 Int16/32 精度计算。因此，如果没有乘法 $_{i16}$ 指令，零点量化可能会很慢。在这两种情况下，输出都累积为 32 位整数 C_{i32}。为了反量化 C_{i32}，我们除以缩放常数 $nd_{a_{f16}}$ 和 $nd_{b_{f16}}$。

Int8 Matrix Multiplication with 16-bit Float Inputs and Outputs. Given hidden states $\mathrm{X}_{f16} \in \mathbb{R}^{s\times h}$ and weights $\mathrm{W}_{f16} \in \mathbb{R}^{h\times o}$ with sequence dimension s, feature dimension h, and output dimension o we perform 8-bit matrix multiplication with 16-bit inputs and outputs as follows:
给定隐藏状态 $\mathrm{X}_{f16} \in \mathbb{R}^{s\times h}$ 和权重 $\mathrm{W}_{f16} \in \mathbb{R}^{h\times o}$，其序列维度为 s，特征维度为 h，输出维度为 o，我们使用 16 位输入和输出执行 8 位矩阵乘法，如下所示：

$$\begin{aligned} \mathrm{X}_{f16}\mathrm{W}_{f16} = \mathrm{C}_{f16} &\approx \frac{1}{c_{x_{f16}}c_{w_{f16}}}\mathrm{C}_{i32} = S_{f16}\cdot \mathrm{C}_{i32} \\ &\approx S_{f16}\cdot \mathrm{A}_{i8}\mathrm{B}_{i8} = S_{f16}\cdot Q(\mathrm{A}_{f16})\, Q(\mathrm{B}_{f16}), \end{aligned} \tag{6}$$

Where $Q(\cdot)$ is either absmax or zeropoint quantization and $c_{x_{f16}}$ and $c_{w_{f16}}$ are the respective tensor-wise scaling constants s_x and s_w for absmax or nd_x and nd_w for zeropoint quantization.
其中 $Q(\cdot)$ 是绝对最大或零点量化，而 $c_{x_{f16}}$ 和 $c_{w_{f16}}$ 分别是绝对最大时的张量缩放常数 s_x 和 s_w，或零点量化时的 nd_x 和 nd_w。

3 Int8 Matrix Multiplication at Scale

The main challenge with quantization methods that use a single scaling constant per tensor is that a single outlier can reduce the quantization precision of all other values. As such, it is desirable to have multiple scaling constants per tensor, such as block-wise constants [11], so that the effect of that outliers is confined to each block. We improve upon one of the most common ways of blocking quantization, row-wise quantization [25], by using vector-wise quantization, as described in more detail below.
每个张量使用单个缩放常数的量化方法的主要挑战是单个异常值会降低所有其他值的量化精度。因此，最好每个张量有多个缩放常数，例如逐块常数 [11]，以便将异常值的影响限制在每个块中。我们通过使用逐向量量化改进了最常见的分块量化方法之一，即逐行量化 [25]，如下文所述。

To handle the large magnitude outlier features that occur in all transformer layers beyond the 6.7B scale, vector-wise quantization is no longer sufficient. For this purpose, we develop mixed-precision decomposition, where the small number of large magnitude feature dimensions (≈0.1%) are represented in 16-bit precision while the other 99.9% of values are multiplied in 8-bit. Since most entries are still represented in low-precision, we retain about 50% memory reduction compared to 16-bit. For example, for BLOOM-176B, we reduce the memory footprint of the model by 1.96x.
为了处理 6.7B 规模之外的所有 Transformer 层中出现的大量异常特征，向量量化已不再足够。为此，我们开发了混合精度分解，其中少量大量特征维度（约 0.1%）以 16 位精度表示，而其他 99.9% 的值以 8 位相乘。由于大多数条目仍以低精度表示，因此与 16 位相比，我们保留了约 50% 的内存减少。例如，对于 BLOOM-176B，我们将模型的内存占用减少了 1.96 倍。

Vector-wise quantization and mixed-precision decomposition are shown in Figure 2. The LLM.int8() method is the combination of absmax vector-wise quantization and mixed precision decomposition.
矢量量化和混合精度分解如图 2 所示。LLM.int8() 方法是 absmax 矢量量化和混合精度分解的组合。

3.1 Vector-wise Quantization

One way to increase the number of scaling constants for matrix multiplication is to view matrix multiplication as a sequence of independent inner products. Given the hidden states $\mathrm{X}_{f16} \in \mathbb{R}^{b\times h}$ and weight matrix $\mathrm{W}_{f16} \in \mathbb{R}^{h\times o}$, we can assign a different scaling constant $c_{x_{f16}}$to each row of X_{f16} and c_w to each column of W_{f16}. To dequantize, we denormalize each inner product result by $1/(c_{x_{f16}} c_{w_{f16}})$. For the whole matrix multiplication this is equivalent to denormalization by the outer product $\mathrm{c}_{x_{f16}} \otimes \mathrm{c}_{w_{f16}}$, where $\mathrm{c}_x \in \mathbb{R}^s$ and $\mathrm{c}_w \in \mathbb{R}^o$. As such the full equation for matrix multiplication with row and column constants is given by:
增加矩阵乘法缩放常数数量的一种方法是将矩阵乘法视为一系列独立的内积。给定隐藏状态 $\mathrm{X}_{f16} \in \mathbb{R}^{b\times h}$ 和权重矩阵 $\mathrm{W}_{f16} \in \mathbb{R}^{h\times o}$，我们可以为 X_{f16} 的每一行分配不同的缩放常数 $c_{x_{f16}}$，为 W_{f16} 的每一列分配不同的缩放常数 $c_{x_{f16}}$。为了反量化，我们将每个内积结果反规范化为 $1/(c_{x_{f16}} c_{w_{f16}})$。对于整个矩阵乘法，这等效于通过外积 $\mathrm{c}_{x_{f16}} \otimes \mathrm{c}_{w_{f16}}$ 进行非规范化，其中 $\mathrm{c}_x \in \mathbb{R}^s$ 和 $\mathrm{c}_w \in \mathbb{R}^o$。因此，具有行和列常数的矩阵乘法的完整方程由以下公式给出：

$$\mathrm{C}_{f_{16}} \approx \frac{1}{\mathrm{c}_{x_{f16}} \otimes \mathrm{c}_{w_{f16}}} \mathrm{C}_{i32} = S \cdot \mathrm{C}_{i32} = \mathrm{S} \cdot \mathrm{A}_{i8}\mathrm{B}_{i8} = \mathrm{S} \cdot Q(\mathrm{A}_{f16})\, Q(\mathrm{B}_{f16}), \tag{7}$$

which we term vector-wise quantization for matrix multiplication.

3.2 The Core of LLM.int8(): Mixed-precision Decomposition

In our analysis, we demonstrate that a significant problem for billion-scale 8-bit transformers is that they have large magnitude features (columns), which are important for transformer performance and require high precision quantization. However, vector-wise quantization, our best quantization technique, quantizes each row for the hidden state, which is ineffective for outlier features. Luckily, we see that these outlier features are both incredibly sparse and systematic in practice, making up only about 0.1% of all feature dimensions, thus allowing us to develop a new decomposition technique that focuses on high precision multiplication for these particular dimensions.
在我们的分析中，我们证明了十亿级 8 位 Transformer 的一个重大问题是它们具有大量特征（列），这对于 Transformer 性能至关重要，并且需要高精度量化。然而，我们最好的量化技术——向量量化，会量化隐藏状态的每一行，这对于异常特征无效。幸运的是，我们发现这些异常特征在实践中非常稀疏且系统化，仅占所有特征维度的约 0.1

We find that given input matrix $\mathrm{X}_{f16} \in \mathbb{R}^{s\times h}$, these outliers occur systematically for almost all sequence dimensions s but are limited to specific feature/hidden dimensions h. As such, we propose mixed-precision decomposition for matrix multiplication where we separate outlier feature dimensions into the set $O = \{i|i \in \mathbb{Z}, 0 \leq i \leq h\}$, which contains all dimensions of h which have at least one outlier with a magnitude larger than the threshold α. In our work, we find that $\alpha = 6.0$ is sufficient to reduce transformer performance degradation close to zero. Using Einstein notation where all indices are superscripts, given the weight matrix $\mathrm{W}_{f16} \in \mathbb{R}^{h\times o}$, mixed-precision decomposition for matrix multiplication is defined as follows:
我们发现，给定输入矩阵 $\mathrm{X}_{f16} \in \mathbb{R}^{s\times h}$，这些异常值几乎在所有序列维度 s 中都会系统地出现，但仅限于特定特征/隐藏维度 h。因此，我们提出了矩阵乘法的混合精度分解，其中我们将异常特征维度分离到集合 $O = \{i|i \in \mathbb{Z}, 0 \leq i \leq h\}$ 中，它包含 h 中所有至少有一个异常值幅度大于阈值 α 的维度。在我们的工作中，我们发现 $\alpha = 6.0$ 足以将 Transformer 性能下降降低到接近零。使用所有指标都是上标的爱因斯坦符号，给定权重矩阵 $\mathrm{W}_{f16} \in \mathbb{R}^{h\times o}$，矩阵乘法的混合精度分解定义如下：

$$\mathrm{C}_{f16} \approx \sum_{h\in O} \mathrm{X}^h_{f16}\mathrm{W}^h_{f16} + \mathrm{S}_{f16} \cdot \sum_{h\notin O} \mathrm{X}^h_{i8}\mathrm{W}^h_{i8} \tag{8}$$

where S_{f16} is the denormalization term for the Int8 inputs and weight matrices X_{i8} and W_{i8}.
其中 S_{f16} 是 Int8 输入和权重矩阵 X_{i8} 和 W_{i8} 的非规范化项。

This separation into 8-bit and 16-bit allows for high-precision multiplication of outliers while using memory-efficient matrix multiplication with 8-bit weights of more than 99.9% of values. Since the number of outlier feature dimensions is not larger than 7 ($|O| \leq 7$) for transformers up to 13B parameters, this decomposition operation only consumes about 0.1% additional memory.
这种 8 位和 16 位分离允许对异常值进行高精度乘法，同时使用内存高效的矩阵乘法，其中 8 位权重的值超过 99.9%。由于对于最多 13B 个参数的 transformers，异常值特征维度的数量不大于 7 ($|O| \leq 7$)，因此此分解操作仅消耗约 0.1% 的额外内存。

3.3 Experimental Setup

We measure the robustness of quantization methods as we scale the size of several publicly available pre-trained language models up to 175B parameters. The key question is not how well a quantization method performs for a particular model but the trend of how such a method performs as we scale.
我们将几个公开可用的预训练语言模型的规模扩大到 1750 亿个参数，以此衡量量化方法的稳健性。关键问题不在于量化方法对特定模型的表现如何，而是这种方法在规模扩大后的表现趋势。

We use two setups for our experiments. One is based on language modeling perplexity, which we find to be a highly robust measure that is very sensitive to quantization degradation. We use this setup to compare different quantization baselines. Additionally, we evaluate zeroshot accuracy degradation on OPT models for a range of different end tasks, where we compare our methods with a 16-bit baseline.
我们在实验中使用了两种设置。一种是基于语言建模困惑度，我们发现这是一种非常稳健的测量方法，对量化退化非常敏感。我们使用此设置比较不同的量化基线。此外，我们评估了一系列不同最终任务的 OPT 模型的零样本准确度退化情况，其中我们将我们的方法与 16 位基线进行了比较。

For the language modeling setup, we use dense autoregressive transformers pretrained in fairseq [35] ranging between 125M and 13B parameters. These transformers have been pretrained on Books [63], English Wikipedia, CC-News [33], OpenWebText [19], CC-Stories [47], and English CC100 [51]. For more information on how these pretrained models are trained, see [1].
对于语言建模设置，我们使用在 fairseq [35] 中预训练的密集自回归变换器，范围在 125M 到 13B 参数之间。这些变换器已在 Books [63]、English Wikipedia、CC-News [33]、OpenWebText [19]、CC-Stories [47] 和 English CC100 [51] 上进行预训练。有关如何训练这些预训练模型的更多信息，请参阅 [1]。

To evaluate the language modeling degradation after Int8 quantization, we evaluate the perplexity of the 8-bit transformer on validation data of the C4 corpus [40] which is a subset of the Common Crawl corpus.[4] We use NVIDIA A40 GPUs for this evaluation.
为了评估 Int8 量化后的语言建模退化，我们评估了 8 位变换器在 C4 语料库 [40] 的验证数据上的困惑度，该语料库是 Common Crawl 语料库的一个子集。我们使用 NVIDIA A40 GPU 进行此次评估。

To measure degradation in zeroshot performance, we use OPT models [59], and we evaluate these models on the EleutherAI language model evaluation harness [17].
为了衡量零样本性能的下降，我们使用 OPT 模型 [?]，并在 EleutherAI 语言模型评估工具 [17] 上评估这些模型。

3.4 Main Results

The main language modeling perplexity results on the 125M to 13B Int8 models evaluated on the C4 corpus can be seen in Table 1. We see that absmax, row-wise, and zeropoint quantization fail as we scale, where models after 2.7B parameters perform worse than smaller models. Zeropoint quantization fails instead beyond 6.7B parameters. Our method, LLM.int8(), is the only method that preserves perplexity. As such, LLM.int8() is the only method with a favorable scaling trend.
在 C4 语料库上评估的 125M 到 13B Int8 模型的主要语言建模困惑度结果可以在 Table 1 中看到。我们发现，随着规模的扩大，absmax、按行和零点量化会失效，其中 2.7B 参数之后的模型表现比较小的模型更差。而零点量化则会在超过 6.7B 参数时失效。我们的方法 LLM.int8() 是唯一保留困惑度的方法。因此，LLM.int8() 是唯一具有良好扩展趋势的方法。

When we look at the scaling trends of zeroshot performance of OPT models on the EleutherAI language model evaluation harness in Figure 1, we see that LLM.int8() maintains full 16-bit performance as we scale from 125M to 175B parameters. On the other hand, the baseline, 8-bit absmax vector-wise quantization, scales poorly and degenerates into random performance.
当我们查看图 1中 EleutherAI 语言模型评估工具上 OPT 模型的零样本性能扩展趋势时，我们发现 LLM.int8() 在参数从 125M 扩展到 175B 时保持了完整的 16 位性能。另一方面，基线 8 位绝对最大矢量量化扩展性较差，并退化为随机性能。

Although our primary focus is on saving memory, we also measured the run time of LLM.int8(). The quantization overhead can slow inference for models with less than 6.7B parameters, as compared to a FP16 baseline. However, models of 6.7B parameters or less fit on most GPUs and quantization is less needed in practice. LLM.int8() run times is about two times faster for large matrix multiplications equivalent to those in 175B models. Appendix D provides more details on these experiments.
虽然我们主要关注节省内存，但我们也测量了 LLM.int8() 的运行时间。与 FP16 基线相比，量化开销可能会减慢参数少于 6.7B 的模型的推理速度。但是，6.7B 或更少参数的模型适合大多数 GPU，并且实践中量化的需要较少。对于相当于 175B 的大型矩阵乘法，LLM.int8() 的运行时间大约快两倍。

[4] https://commoncrawl.org/

Table 1: C4 validation perplexities of quantization methods for different transformer sizes ranging from 125M to 13B parameters. We see that absmax, row-wise, zeropoint, and vector-wise quantization leads to significant performance degradation as we scale, particularly at the 13B mark where 8-bit 13B perplexity is worse than 8-bit 6.7B perplexity. If we use LLM.int8(), we recover full perplexity as we scale. Zeropoint quantization shows an advantage due to asymmetric quantization but is no longer advantageous when used with mixed-precision decomposition.
不同 Transformer 大小（从 125M 到 13B 参数）的量化方法的 C4 验证困惑度。我们发现，随着规模的扩大，绝对量化、逐行量化、零点量化和逐向量量化会导致性能显著下降，尤其是在 13B token 处，8 位 13B 困惑度比 8 位 6.7B 困惑度更差。如果我们使用 LLM.int8()，我们会在规模扩大时恢复完全困惑度。零点量化由于非对称量化而显示出优势，但与混合精度分解一起使用时不再具有优势。

Parameters	125M	1.3B	2.7B	6.7B	13B
32-bit Float	25.65	15.91	14.43	13.30	12.45
Int8 absmax	87.76	16.55	15.11	14.59	19.08
Int8 zeropoint	56.66	16.24	14.76	13.49	13.94
Int8 absmax row-wise	30.93	17.08	15.24	14.13	16.49
Int8 absmax vector-wise	35.84	16.82	14.98	14.13	16.48
Int8 zeropoint vector-wise	25.72	15.94	14.36	13.38	13.47
Int8 absmax row-wise + decomposition	30.76	16.19	14.65	13.25	12.46
Absmax LLM.int8() (vector-wise + decomp)	25.83	15.93	14.44	13.24	12.45
Zeropoint LLM.int8() (vector-wise + decomp)	25.69	15.92	14.43	13.24	12.45

4 Emergent Large Magnitude Features in Transformers at Scale

As we scale transformers, outlier features with large magnitudes emerge and strongly affect all layers and their quantization. Given a hidden state $X \in \mathbb{R}^{s\times h}$ where s is the sequence/token dimension and h the hidden/feature dimension, we define a feature to be a particular dimension h_i. Our analysis looks at a particular feature dimension h_i across all layers of a given transformer.
随着我们扩展 Transformer，会出现大量异常特征，并严重影响 all 层及其量化。给定一个隐藏状态 $X \in \mathbb{R}^{s\times h}$，其中 s 是序列/token 维度，h 是隐藏/特征维度，我们将特征定义为特定维度 h_i。我们的分析着眼于给定 Transformer 所有层中的特定特征维度 h_i。

We find that outlier features strongly affect attention and the overall predictive performance of transformers. While up to 150k outliers exist per 2048 token sequence for a 13B model, these outlier features are highly systematic and only representing at most 7 unique feature dimensions h_i. Insights from this analysis were critical to developing mixed-precision decomposition. Our analysis explains the advantages of zeropoint quantization and why they disappear with the use of mixed-precision decomposition and the quantization performance of small vs. large models.
我们发现异常特征会严重影响注意力和 Transformer 的整体预测性能。虽然 13B 模型中每 2048 个 token 序列最多有 15 万个异常值，但这些异常特征具有高度系统性，最多仅代表 7 个唯一特征维度 h_i。从此分析中获得的见解对于开发混合精度分解至关重要。我们的分析解释了零点量化的优势以及为什么它们会随着混合精度分解的使用而消失，以及小型模型与大型模型的量化性能。

4.1 Finding Outlier Features

The difficulty with the quantitative analysis of emergent phenomena is two-fold. We aim to select a small subset of features for analysis such that the results are intelligible and not to complex while also capturing important probabilistic and structured patterns. We use an empirical approach to find these constraints. We define outliers according to the following criteria: the magnitude of the feature is at least 6.0, affects at least 25% of layers, and affects at least 6% of the sequence dimensions.
对突发现象进行定量分析的困难有两个方面。我们的目标是选择一小部分特征进行分析，以便结果易于理解且不太复杂，同时还能捕捉重要的概率和结构化模式。我们使用经验方法来找到这些约束。我们根据以下标准定义异常值：特征的量级至少为 6.0，影响至少 25% 的层，并影响至少 6% 的序列维度。

More formally, given a transformer with L layers and hidden state $X_l \in \mathbb{R}^{s\times h}, l = 0...L$ where s is the sequence dimension and h the feature dimension, we define a feature to be a particular dimension h_i in any of the hidden states X_{l_i}.
更正式地讲，给定一个具有 L 层和隐藏状态 $X_l \in \mathbb{R}^{s\times h}, l = 0...L$ 的 Transformer，其中 s 是序列维度，h 是特征维度，我们将特征定义为任何隐藏状态 X_{l_i} 中的特定维度 h_i。

We track dimensions $h_i, 0 \leq i \leq h$, which have at least one value with a magnitude of $\alpha \geq 6$ and we only collect statistics if these outliers occur in the same feature dimension h_i in at least 25% of transformer layers $0...L$ and appear in at least 6% of all sequence dimensions s across all hidden states X_l. Since feature outliers only occur in attention projection (key/query/value/output) and the feedforward network expansion layer (first sub-layer), we ignore the attention function and the FFN contraction layer (second sub-layer) for this analysis.
我们跟踪维度 $h_i, 0 \leq i \leq h$，这些维度至少有一个幅度为 $\alpha \geq 6$ 的值，并且仅当这些异常值出现在至少 25% 的 Transformer 层 $0...L$ 中的特征维度 h_i 中，并出现在所有隐藏状态 X_l 中所有序列维度 s 的至少 6% 中时，我们才会收集统计数据。由于特征异常值仅出现在注意力投射（键/查询/值/输出）和前馈网络扩展层（第一个子层）中，因此我们忽略注意力函数和 FFN 收缩层（第二个子层）进行分析。

Our reasoning for these thresholds is as follows. We find that using mixed-precision decomposition, perplexity degradation stops if we treat any feature with a magnitude 6 or larger as an outlier feature. For the number of layers affected by outliers, we find that outlier features are systematic in large models: they either occur in most layers or not at all. On the other hand, they are probabilistic in small models: they occur sometimes in some layers for each sequence. As such, we set our threshold for how many layers need to be affected to detect an outlier feature in such a way as to limit detection to a single outlier in our smallest model with 125M parameters. This threshold corresponds to that at least 25% of transformer layers are affected by an outlier in the same feature dimension. The second most common outlier occurs in only a single layer (2% of layers), indicating that this is a reasonable threshold. We use the same procedure to find the threshold for how many sequence dimensions are affected by outlier features in our 125M model: outliers occur in at least 6% of sequence dimensions.
我们对这些阈值的推理如下。我们发现，使用混合精度分解，如果我们将任何幅度为 6 或更大的特征视为异常特征，困惑度下降就会停止。对于受异常值影响的层数，我们发现异常值特征在大型模型中是系统性的：它们要么出现在大多数层中，要么根本不出现。另一方面，它们在小型模型中是概率性的：它们有时出现在每个序列的某些层中。因此，我们设置了需要影响多少层才能检测到异常特征的阈值，以便在具有 125M 个参数的最小模型中将检测限制为单个异常值。此阈值对应于至少 25% 的 Transformer 层受到同一特征维度中异常值的影响。第二常见的异常值仅出现在单个层中（ 2% 的层），表明这是一个合理的阈值。我们使用相同的程序来找到 125M 模型中受异常值特征影响的序列维度数量的阈值：异常值出现在至少 6% 的序列维度中。

We test models up to a scale of 13B parameters. To make sure that the observed phenomena are not due to bugs in software, we evaluate transformers that were trained in three different software frameworks. We evaluate four GPT-2 models which use OpenAI software, five Meta AI models that use Fairseq [35], and one EleutherAI model GPT-J that uses Tensorflow-Mesh [43]. More details can be found in Appendix C. We also perform our analysis in two different inference software frameworks: Fairseq and Hugging Face Transformers [52].
我们测试的模型规模高达 13B 个参数。为了确保观察到的现象不是由于软件中的错误造成的，我们评估了在三个不同的软件框架中训练的 Transformer。我们评估了四个使用 OpenAI 软件的 GPT-2 模型、五个使用 Fairseq [35] 的 Meta AI 模型和一个使用 Tensorflow-Mesh [43] 的 EleutherAI 模型 GPT-J。更多详细信息可在附录 C 中找到。我们还在两个不同的推理软件框架中执行分析：Fairseq 和 Hugging Face Transformers [52]。

4.2 Measuring the Effect of Outlier Features

To demonstrate that the outlier features are essential for attention and predictive performance, we set the outlier features to zero before feeding the hidden states X_l into the attention projection layers and then compare the top-1 softmax probability with the regular softmax probability with outliers. We do this for all layers independently, meaning we forward the regular softmax probabilities values to avoid cascading errors and isolate the effects due to the outlier features. We also report the perplexity degradation if we remove the outlier feature dimension (setting them to zero) and propagate these altered, hidden states through the transformer. As a control, we apply the same procedure for random non-outlier feature dimensions and note attention and perplexity degradation.
为了证明异常特征对于注意力和预测性能至关重要，我们在将隐藏状态 X_l 馈入注意力投影层之前将异常特征设置为零，然后将 top-1 softmax 概率与具有异常值的常规 softmax 概率进行比较。我们对所有层独立执行此操作，这意味着我们转发常规 softmax 概率值以避免级联错误并隔离由于异常特征造成的影响。如果我们删除异常特征维度（将其设置为零）并通过 Transformer 传播这些改变的隐藏状态，我们还会报告困惑度下降。作为对照，我们对随机非异常特征维度应用相同的程序，并注意注意力和困惑度下降。

Our main quantitative results can be summarized as four main points.
我们的主要定量结果可以概括为四个要点。

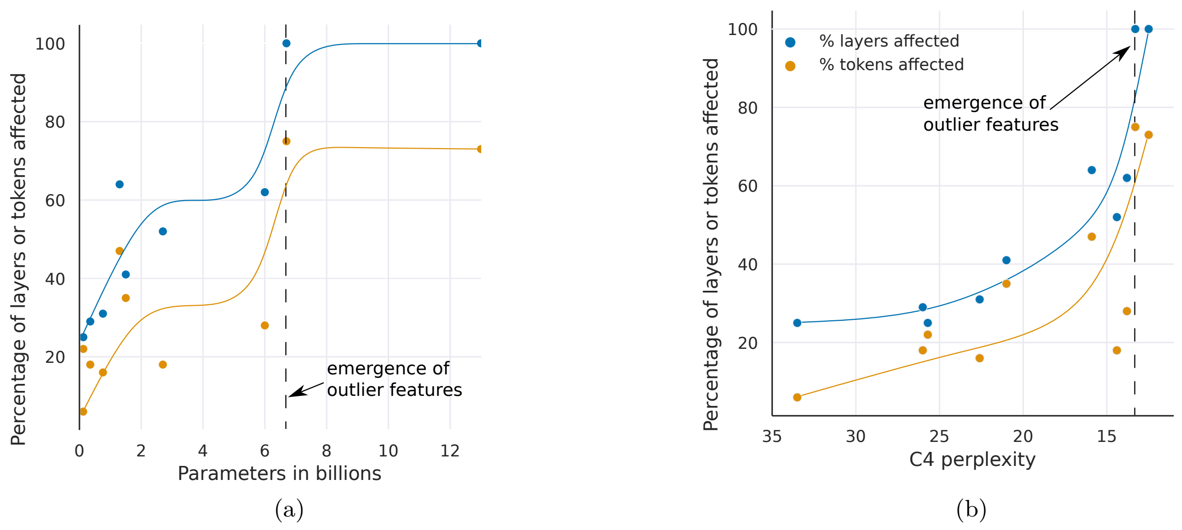

Figure 3: Percentage of layers and all sequence dimensions affected by large magnitude outlier features across the transformer by (a) model size or (b) C4 perplexity. Lines are B-spline interpolations of 4 and 9 linear segments for (a) and (b). Once the phase shift occurs, outliers are present in all layers and in about 75% of all sequence dimensions. While (a) suggest a sudden phase shift in parameter size, (b) suggests a gradual exponential phase shift as perplexity decreases. The stark shift in (a) co-occurs with the sudden degradation of performance in quantization methods.
Transformer 中受 (a) 模型大小或 (b) C4 困惑度影响的层和所有序列维度的百分比。线条是 (a) 和 (b) 的 4 和 9 个线性段的 B 样条插值。一旦发生相移，所有层和约 75% 的所有序列维度中都会出现异常值。虽然 (a) 表明参数大小的突然相移，但 (b) 表明随着困惑度的降低，相移呈逐渐指数增长。(a) 中的明显转变与量化方法性能的突然下降同时发生。

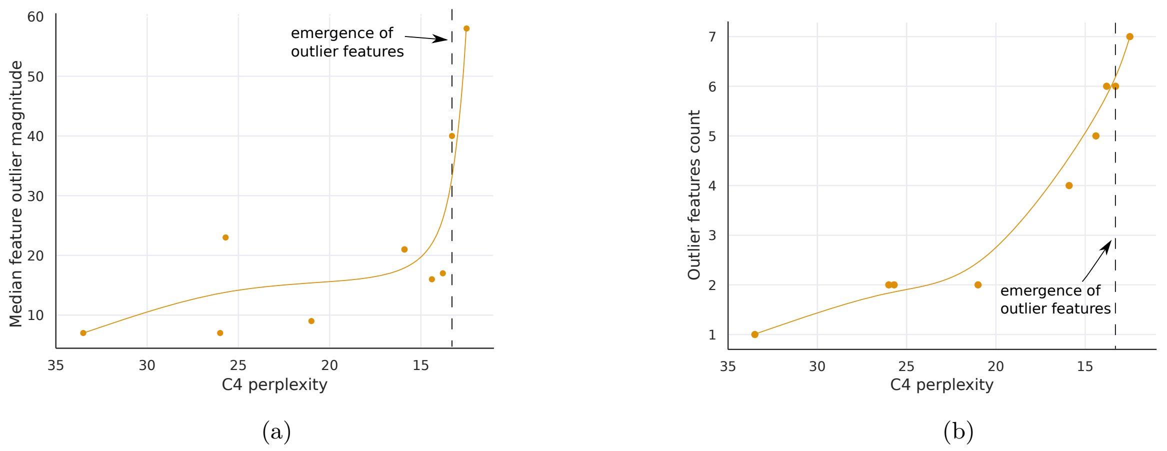

Figure 4: The median magnitude of the largest outlier feature in (a) indicates a sudden shift in outlier size. This appears to be the prime reason why quantization methods fail after emergence. While the number of outlier feature dimensions is only roughly proportional to model size, (b) shows that the number of outliers is strictly monotonic with respect to perplexity across all models analyzed. Lines are B-spline interpolations of 9 linear segments.
(a) 中最大异常特征的中值幅度表明异常值大小突然发生变化。这似乎是量化方法在出现后失败的主要原因。虽然异常特征维度的数量仅与模型大小大致成正比，但 (b) 表明，在所有分析的模型中，异常值的数量与困惑度的关系是严格单调的。线是 9 个线性段的 B 样条插值。

(1) When measured by the number of parameters, the emergence of large magnitude features across all layers of a transformer occurs suddenly between 6B and 6.7B parameters as shown in Figure 3a as the percentage of layers affected increases from 65% to 100%. The number of sequence dimensions affected increases rapidly from 35% to 75%. This sudden shift co-occurs with the point where quantization begins to fail.
以参数数量来衡量，当受影响的层百分比从 65% 增加到 100% 时，Transformer 的所有层中突然出现了大量特征，如图 3a 所示，在 6B 和 6.7B 参数之间突然出现。受影响的序列维度数量从 35% 迅速增加到 75%。这种突然转变与量化开始失败的点同时发生。

(2) Alternatively, when measured by perplexity, the emergence of large magnitude features across all layers of the transformer can be seen as emerging smoothly according to an exponential function of decreasing perplexity, as seen in Figure 3b. This indicates that there is nothing sudden about emergence and that we might be able to detect emergent features before a phase shift occurs by studying exponential trends in smaller models. This also suggests that emergence is not only about model size but about perplexity, which is related to multiple additional factors such as the amount of training data used, and data quality [22, 21].
或者，当用困惑度来衡量时，Transformer 所有层中大量特征的出现可以看作是根据困惑度降低的指数函数平稳出现的，如图 3b 所示。这表明出现并不突然，我们可能能够通过研究较小模型中的指数趋势在相移发生之前检测到出现的特征。这也表明出现不仅与模型大小有关，还与困惑度有关，困惑度与多个其他因素有关，例如使用的训练数据量和数据质量 [22, 21]。

(3) Median outlier feature magnitude rapidly increases once outlier features occur in all layers of the transformer, as shown in Figure 4a. The large magnitude of outliers features and their asymmetric distribution disrupts Int8 quantization precision. This is the core reason why quantization methods fail starting at the 6.7B scale – the range of the quantization distribution is too large so that most quantization bins are empty and small quantization values are quantized to zero, essentially extinguishing information. We hypothesize that besides Int8 inference, regular 16-bit floating point training becomes unstable due to outliers beyond the 6.7B scale – it is easy to exceed the maximum 16-bit value 65535 by chance if you multiply by vectors filled with values of magnitude 60.
一旦异常特征出现在 Transformer 的所有层中，中位数异常特征幅度就会迅速增加，如图 4a 所示。异常特征的大量级及其不对称分布会破坏 Int8 量化精度。这是量化方法从 6.7B 规模开始失败的核心原因——量化分布的范围太大，以至于大多数量化箱都为空，小量化值被量化为零，从本质上消灭了信息。我们假设，除了 Int8 推理之外，常规的 16 位浮点训练由于 6.7B 规模以外的异常值而变得不稳定——如果乘以填充幅度为 60 的值的向量，很容易偶然超过最大 16 位值 65535。

(4) The number of outliers features increases strictly monotonically with respect to decreasing C4 perplexity as shown in Figure 4b, while a relationship with model size is non-monotonic. This indicates that model perplexity rather than mere model size determines the phase shift. We hypothesize that model size is only one important covariate among many that are required to reach emergence.
如图 4b 所示，异常值特征的数量随着 C4 困惑度的降低而严格单调增加，而与模型大小的关系则非单调。这表明，模型困惑度而非单纯的模型大小决定了相移。我们假设模型大小只是实现突现所需的众多协变量中的一个重要协变量。

These outliers features are highly systematic after the phase shift occurred. For example, for a 6.7B transformer with a sequence length of 2048, we find about 150k outlier features per sequence for the entire transformer, but these features are concentrated in only 6 different hidden dimensions.
这些异常特征在相移发生后具有高度系统性。例如，对于序列长度为 2048 的 6.7B transformer，我们发现整个 transformer 每个序列大约有 150k 个异常特征，但这些特征仅集中在 6 个不同的隐藏维度中。

These outliers are critical for transformer performance. If the outliers are removed, the mean top-1 softmax probability is reduced from about 40% to about 20%, and validation perplexity increases by 600-1000% even though there are at most 7 outlier feature dimensions. When we remove 7 random feature dimensions instead, the top-1 probability decreases only between 0.02-0.3%, and perplexity increases by 0.1%. This highlights the critical nature of these feature dimensions. Quantization precision for these outlier features is paramount as even tiny errors greatly impact model performance.
这些异常值对于 Transformer 的性能至关重要。如果移除异常值，平均 top-1 softmax 概率将从约 40% 降至约 20%，即使异常特征维度最多只有 7 个，验证困惑度也会增加 600-1000%。当我们移除 7 个随机特征维度时，top-1 概率仅下降 0.02-0.3%，困惑度增加 0.1%。这凸显了这些特征维度的关键性。这些异常特征的量化精度至关重要，因为即使是微小的错误也会极大地影响模型性能。

4.3 Interpretation of Quantization Performance

Our analysis shows that outliers in particular feature dimensions are ubiquitous in large transformers, and these feature dimensions are critical for transformer performance. Since row-wise and vector-wise quantization scale each hidden state sequence dimension s (rows) and because outliers occur in the feature dimension h (columns), both methods cannot deal with these outliers effectively. This is why absmax quantization methods fail quickly after emergence.
我们的分析表明，特定特征维度的异常值在大型 Transformer 中普遍存在，而这些特征维度对于 Transformer 的性能至关重要。由于行式和向量式量化会缩放每个隐藏状态序列维度 s（行），并且异常值出现在特征维度 h（列）中，因此这两种方法都无法有效处理这些异常值。这就是 absmax 量化方法在出现后很快失败的原因。

However, almost all outliers have a strict asymmetric distribution: they are either solely positive or negative (see Appendix C). This makes zeropoint quantization particularly effective for these outliers, as zeropoint quantization is an asymmetric quantization method that scales these outliers into the full $[-127, 127]$ range. This explains the strong performance in our quantization scaling benchmark in Table 1. However, at the 13B scale, even zeropoint quantization fails due to accumulated quantization errors and the quick growth of outlier magnitudes, as seen in Figure 4a.
然而，几乎所有的异常值都具有严格的非对称分布：它们要么完全为正，要么完全为负（参见附录 C）。这使得零点量化对这些异常值特别有效，因为零点量化是一种非对称量化方法，可将这些异常值缩放到完整的 $[-127, 127]$ 范围内。这解释了我们在表 1 中的量化缩放基准测试中的强劲表现。然而，在 13B 规模下，即使是零点量化也会因累积量化误差和异常值幅度的快速增长而失败，如图 4a 所示。

If we use our full LLM.int8() method with mixed-precision decomposition, the advantage of zeropoint quantization disappears indicating that the remaining decomposed features are symmetric. However, vector-wise still has an advantage over row-wise quantization, indicating that the enhanced quantization precision of the model weights is needed to retain full precision predictive performance.
如果我们使用完整的 LLM.int8() 方法进行混合精度分解，零点量化的优势就会消失，表明剩余的分解特征是对称的。然而，向量量化仍然比行量化有优势，这表明需要提高模型权重的量化精度才能保持全精度预测性能。

5 Related work

There is closely related work on quantization data types and quantization of transformers, as described below. Appendix B provides further related work on quantization of convolutional networks.
量化数据类型和 Transformer 量化方面有密切相关的工作，如下所述。附录 B 提供了有关卷积网络量化的进一步相关工作。

8-bit Data Types. Our work studies quantization techniques surrounding the Int8 data type, since it is currently the only 8-bit data type supported by GPUs. Other common data types are fixed point or floating point 8-bit data types (FP8). These data types usually have a sign bit and different exponent and fraction bit combinations. For example, a common variant of this data type has 5 bits for the exponent and 2 bits for the fraction [49, 45, 5, 32] and uses either no scaling constants or zeropoint scaling. These data types have large errors for large magnitude values since they have only 2 bits for the fraction but provide high accuracy for small magnitude values. [24] provide an excellent analysis of when certain fixed point exponent/fraction bit widths are optimal for inputs with a particular standard deviation. We believe FP8 data types offer superior performance compared to the Int8 data type, but currently, neither GPUs nor TPUs support this data type.
8 **位数据类型**。我们的工作研究了围绕 Int8 数据类型的量化技术，因为它目前是 GPU 支持的唯一 8 位数据类型。其他常见数据类型是定点或浮点 8 位数据类型 (FP8)。这些数据类型通常具有符号位和不同的指数和小数位组合。例如，此数据类型的常见变体有 5 位指数和 2 位小数，并且不使用缩放常数或零点缩放。这些数据类型对于大幅度值具有较大的误差，因为它们只有 2 位小数，但对于小幅度值具有高精度。[24] 提供了对某些定点指数/小数位宽何时对于具有特定标准偏差的输入最佳的出色分析。我们相信 FP8 数据类型比 Int8 数据类型具有更优异的性能，但目前 GPU 和 TPU 都不支持这种数据类型。

Outlier Features in Language Models. Large magnitude outlier features in language models have been studied before [46, 3, 50, 30]. Previous work proved the theoretical relationship between outlier appearance in transformers and how it relates to layer normalization and the token frequency distribution [16]. Similarly, [26] attribute the appearance of outliers in BERT model family to LayerNorm, and [37] show empirically that outlier emergence is related to the frequency of tokens in the training distribution. We extend this work further by showing how the scale of autoregressive models relates to the emergent properties of these outlier features, and showing how appropriately modeling outliers is critical to effective quantization.
语言模型中的异常特征。在 [46, 3, 50, 30] 之前，已经对语言模型中的大量异常特征进行了研究。先前的工作证明了 Transformer 中异常值出现与层规范化和 token 频率分布 [16] 的关系之间的理论关系。同样，[26] 将 BERT 模型系列中异常值的出现归因于 LayerNorm，而 [37] 通过经验表明，异常值的出现与训练分布中 token 的频率有关。我们通过展示自回归模型的规模与这些异常值特征的出现属性之间的关系，以及展示适当地建模异常值对于有效量化的重要性，进一步扩展了这项工作。

Multi-billion Scale Transformer Quantization. There are two methods that were developed in parallel to ours: nuQmm [36] and ZeroQuant [55]. Both use the same quantization scheme: group-w2ise quantization, which has even finer quantization normalization constant granularity than vector-wise quantization. This scheme offers higher quantization precision but also requires custom CUDA kernels. Both nuQmm and ZeroQuant aim to accelerate inference and reduce the memory footprint while we focus on preserving predictive performance under an 8-bit memory footprint. The largest models that nuQmm and ZeroQuant evaluate are 2.7B and 20B parameter transformers, respectively. ZeroQuant achieves zero-degradation performance for 8-bit quantization of a 20B model. We show that our method allows for zero-degradation quantization of models up to 176B parameters. Both nuQmm and ZeroQuant suggest that finer quantization granularity can be an effective means to quantize large models. These methods are complementary with LLM.int8(). Another parallel work is GLM-130B which uses insights from our work to achieve zero-degradation 8-bit quantization [57]. GLM-130B performs full 16-bit precision matrix multiplication with 8-bit weight storage.
数十亿级 transformer **量化**。有两种方法与我们的方法同时开发：nuQmm [36] 和 ZeroQuant [55]。两者都使用相同的量化方案：分组量化，其量化规范化常数粒度甚至比矢量量化更细。该方案提供更高的量化精度，但也需要自定义 CUDA 内核。nuQmm 和 ZeroQuant 都旨在加速推理并减少内存占用，而我们专注于在 8 位内存占用下保持预测性能。nuQmm 和 ZeroQuant 评估的最大模型分别是 2.7B 和 20B 参数 transformer。ZeroQuant 对 20B 模型的 8 位量化实现了零降级性能。我们表明，我们的方法允许对多达 176B 参数的模型进行零降级量化。nuQmm 和 ZeroQuant 都表明，更精细的量化粒度可以成为量化大型模型的有效手段。这些方法与 LLM.int8() 相辅相成。另一项并行工作是 GLM-130B，它利用我们工作中的见解实现零降级 8 位量化 [57]。GLM-130B 执行完整的 16 位精度矩阵乘法，并存储 8 位权重。

6 Discussion and Limitations

We have demonstrated for the first time that multi-billion parameter transformers can be quantized to Int8 and used immediately for inference without performance degradation. We achieve this by using our insights from analyzing emergent large magnitude features at scale to develop mixed-precision decomposition to isolate outlier features in a separate 16-bit matrix multiplication. In conjunction with vector-wise quantization that yields our method, LLM.int8(), which we show empirically can recover the full inference performance of models with up to 175B parameters.
我们首次证明，数十亿参数 transformers 可以量化为 Int8，并立即用于推理，而不会降低性能。我们利用从大规模分析新兴大规模特征中获得的见解，开发混合精度分解，以在单独的 16 位矩阵乘法中隔离异常特征，从而实现这一目标。结合向量量化，我们的方法 LLM.int8() 得到了结果，我们通过实证证明，该方法可以恢复具有多达 175B 个参数的模型的全部推理性能。

The main limitation of our work is that our analysis is solely on the Int8 data type, and we do not study 8-bit floating-point (FP8) data types. Since current GPUs and TPUs do not support this data type, we believe this is best left for future work. However, we also believe many insights from Int8 data types will directly translate to FP8 data types. Another limitation is that we only study models with up to 175B parameters. While we quantize a 175B model to Int8 without performance degradation, additional emergent properties might disrupt our quantization methods at larger scales.
我们工作的主要限制在于，我们的分析仅限于 Int8 数据类型，而我们没有研究 8 位浮点 (FP8) 数据类型。由于当前的 GPU 和 TPU 不支持此数据类型，我们认为这最好留待将来研究。但是，我们也相信，许多来自 Int8 数据类型的见解将直接转化为 FP8 数据类型。另一个限制是我们只研究最多 175B 个参数的模型。虽然我们将 175B 模型量化为 Int8 而不会降低性能，但额外的新兴属性可能会破坏我们在更大规模上的量化方法。

A third limitation is that we do not use Int8 multiplication for the attention function. Since our focus is on reducing the memory footprint and the attention function does not use any parameters, it was not strictly needed. However, an initial exploration of this problem indicated that a solution required additional quantization methods beyond those we developed here, and we leave this for future work.
第三个限制是我们没有对注意力函数使用 Int8 乘法。由于我们的重点是减少内存占用，并且注意力函数不使用任何参数，因此这并不是必需的。然而，对这个问题的初步探索表明，解决方案需要我们在此开发的量化方法以外的其他量化方法，我们将此留待将来研究。

A final limitation is that we focus on inference but do not study training or finetuning. We provide an initial analysis of Int8 finetuning and training at scale in Appendix E. Int8 training at scale requires complex trade-offs between quantization precision, training speed, and engineering complexity and represents a very difficult problem. We again leave this to future work.
最后一个限制是我们专注于推理，但不研究训练或微调。我们在附录 E 中提供了 Int8 微调和大规模训练的初步分析。大规模 Int8 训练需要在量化精度、训练速度和工程复杂性之间进行复杂的权衡，这是一个非常困难的问题。我们再次将此留给未来的工作。

Table 2: Different hardware setups and which methods can be run in 16-bit vs. 8-bit precision. We can see that our 8-bit method makes many models accessible that were not accessible before, in particular, OPT-175B/BLOOM.
不同的硬件设置以及哪些方法可以在 16 位和 8 位精度下运行。我们可以看到，我们的 8 位方法使许多以前无法访问的模型变得可用，尤其是 OPT-175B/BLOOM。

			Largest Model that can be run	
Class	Hardware	GPU Memory	8-bit	16-bit
Enterprise	8x A100	80 GB	OPT-175B / BLOOM	OPT-175B / BLOOM
Enterprise	8x A100	40 GB	OPT-175B / BLOOM	OPT-66B
Academic server	8x RTX 3090	24 GB	OPT-175B / BLOOM	OPT-66B
Academic desktop	4x RTX 3090	24 GB	OPT-66B	OPT-30B
Paid Cloud	Colab Pro	15 GB	OPT-13B	GPT-J-6B
Free Cloud	Colab	12 GB	T0/T5-11B	GPT-2 1.3B

7 Broader Impacts

The main impact of our work is enabling access to large models that previously could not fit into GPU memory. This enables research and applications which were not possible before due to limited GPU memory, in particular for researchers with the least resources. See Table 3 for model/GPU combinations which are now accessible without performance degradation. However, our work also enables resource-rich organizations with many GPUs to serve more models on the same number of GPUs, which might increase the disparities between resource-rich and poor organizations.
我们的工作的主要影响是能够访问以前无法放入 GPU 内存的大型模型。这使得以前由于 GPU 内存有限而无法进行的研究和应用成为可能，特别是对于资源最少的研究人员而言。请参阅表 3，了解现在可以访问且性能不会降低的模型/GPU 组合。然而，我们的工作也使拥有许多 GPU 的资源丰富的组织能够在相同数量的 GPU 上提供更多模型，这可能会增加资源丰富和资源匮乏的组织之间的差距。

In particular, we believe that the public release of large pretrained models, for example, the recent Open Pretrained Transformers (OPT) [59], along with our new Int8 inference for zero- and few-shot prompting, will enable new research for academic institutions that was not possible before due to resource constraints. The widespread accessibility of such large-scale models will likely have both beneficial and detrimental effects on society that are difficult to predict.
具体而言，我们认为，大型预训练模型的公开发布，例如最近的 Open Pretrained Transformers (OPT) [59]，以及我们针对零样本和小样本提示的新 Int8 推理，将使学术机构能够开展新的研究，而这在以前由于资源限制而无法实现。这种大规模模型的广泛普及可能会对社会产生难以预测的利弊影响。

Acknowledgments We thank Ofir Press, Gabriel Ilharco, Daniel Jiang, Mitchell Wortsman, Ari Holtzman, Mitchell Gordon for their feedback on drafts of this work. We thank JustHeuristic (Yozh) and Titus von Köller for help with Hugging Face Transformers integration.

References

[1] M. Artetxe, S. Bhosale, N. Goyal, T. Mihaylov, M. Ott, S. Shleifer, X. V. Lin, J. Du, S. Iyer, R. Pasunuru, et al. Efficient large scale language modeling with mixtures of experts. arXiv preprint arXiv:2112.10684, 2021.

[2] H. Bai, W. Zhang, L. Hou, L. Shang, J. Jin, X. Jiang, Q. Liu, M. R. Lyu, and I. King. Binarybert: Pushing the limit of bert quantization. ArXiv, abs/2012.15701, 2021.

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

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020.

[5] L. Cambier, A. Bhiwandiwalla, T. Gong, O. H. Elibol, M. Nekuii, and H. Tang. Shifted and squeezed 8-bit floating point format for low-precision training of deep neural networks. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020. URL https://openreview.net/forum?id=Bkxe2AVtPS.

[6] J. Chen, Y. Gai, Z. Yao, M. W. Mahoney, and J. E. Gonzalez. A statistical framework for low-bitwidth training of deep neural networks. Advances in Neural Information Processing Systems, 33:883–894, 2020.

[7] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang. Accurate and efficient 2-bit quantized neural networks. In A. Talwalkar, V. Smith, and M. Zaharia, editors, Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019. mlsys.org, 2019. URL https://proceedings.mlsys.org/book/268.pdf.

[8] M. Courbariaux and Y. Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. CoRR, abs/1602.02830, 2016. URL http://arxiv.org/abs/1602.02830.

[9] M. Courbariaux, Y. Bengio, and J.-P. David. Training deep neural networks with low precision multiplications. arXiv preprint arXiv:1412.7024, 2014.

[10] M. Courbariaux, Y. Bengio, and J. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pages 3123–3131, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/3e15cc11f979ed25912dff5b0669f2cd-Abstract.html.

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

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[13] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 293–302, 2019.

[14] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha. Learned step size quantization. arXiv preprint arXiv:1902.08153, 2019.

[15] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin. Training with quantization noise for extreme model compression. arXiv preprint arXiv:2004.07320, 2020.

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

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

[18] 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.

[19] A. Gokaslan and V. Cohen. Openwebtext corpus. urlhttp://Skylion007. github. io/OpenWebTextCorpus, 2019.

[20] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 4851–4860. IEEE, 2019. doi: 10.1109/ICCV.2019.00495. URL https://doi.org/10.1109/ICCV.2019.00495.

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

[22] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. arXiv preprint arXiv:2203.15556, 2022.

[23] G. Ilharco, C. Ilharco, I. Turc, T. Dettmers, F. Ferreira, and K. Lee. High performance natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts, pages 24–27, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-tutorials.4. URL https://aclanthology.org/2020.emnlp-tutorials.4.

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

[25] D. Khudia, J. Huang, P. Basu, S. Deng, H. Liu, J. Park, and M. Smelyanskiy. Fbgemm: Enabling high-performance low-precision deep learning inference. arXiv preprint arXiv:2101.05615, 2021.

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

[27] R. Li, Y. Wang, F. Liang, H. Qin, J. Yan, and R. Fan. Fully quantized network for object detection. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 2810–2819. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00292. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Li_Fully_Quantized_Network_for_Object_Detection_CVPR_2019_paper.html.

[28] 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.

[29] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[30] Z. Luo, A. Kulmizev, and X. Mao. 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), pages 5312–5327, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.413. URL https://aclanthology.org/2021.acl-long.413.

[31] M. Macháček and O. Bojar. Results of the wmt14 metrics shared task. In Proceedings of the Ninth Workshop on Statistical Machine Translation, pages 293–301, 2014.

[32] N. Mellempudi, S. Srinivasan, D. Das, and B. Kaul. Mixed precision training with 8-bit floating point. CoRR, abs/1905.12334, 2019. URL http://arxiv.org/abs/1905.12334.

[33] S. Nagel. Cc-news, 2016.

[34] M. Ott, S. Edunov, D. Grangier, and M. Auli. Scaling neural machine translation. arXiv preprint arXiv:1806.00187, 2018.

[35] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling. arXiv preprint arXiv:1904.01038, 2019.

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

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

[38] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe. Binary neural networks: A survey. CoRR, abs/2004.03333, 2020. URL https://arxiv.org/abs/2004.03333.

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

[40] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683, 2019.

[41] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, 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, pages 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.

[42] R. Sennrich, B. Haddow, and A. Birch. Edinburgh neural machine translation systems for wmt 16. arXiv preprint arXiv:1606.02891, 2016.

[43] N. Shazeer, Y. Cheng, N. Parmar, D. Tran, A. Vaswani, P. Koanantakool, P. Hawkins, H. Lee, M. Hong, C. Young, et al. Mesh-tensorflow: Deep learning for supercomputers. Advances in neural information processing systems, 31, 2018.

[44] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8815–8821, 2020.

[45] X. Sun, J. Choi, C. Chen, N. Wang, S. Venkataramani, V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan. Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 4901–4910, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/65fc9fb4897a89789352e211ca2d398f-Abstract.html.

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

[47] T. H. Trinh and Q. V. Le. A simple method for commonsense reasoning. arXiv preprint arXiv:1806.02847, 2018.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.

[49] N. Wang, J. Choi, D. Brand, C. Chen, and K. Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 7686–7695, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/335d3d1cd7ef05ec77714a215134914c-Abstract.html.

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

[51] G. Wenzek, M.-A. Lachaux, A. Conneau, V. Chaudhary, F. Guzmán, A. Joulin, and E. Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 4003–4012, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://www.aclweb.org/anthology/2020.lrec-1.494.

[52] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771, 2019.

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

[54] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In International Conference on Machine Learning, pages 11875–11886. PMLR, 2021.

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

[56] 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.

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

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

[59] 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.

[60] W. Zhang, L. Hou, Y. Yin, L. Shang, X. Chen, X. Jiang, and Q. Liu. Ternarybert: Distillation-aware ultra-low bit bert. In EMNLP, 2020.

[61] C. Zhao, T. Hua, Y. Shen, Q. Lou, and H. Jin. Automatic mixed-precision quantization search of bert. arXiv preprint arXiv:2112.14938, 2021.

[62] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1_pAu9xl.

[63] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In Proceedings of the IEEE international conference on computer vision, pages 19–27, 2015.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default [TODO] to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section ??.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the limitation section
   (c) Did you discuss any potential negative societal impacts of your work?[Yes] See the Broader Impacts section
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them?[Yes] Yes, we believe our work conforms to these guidelines.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will include our code in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)?[Yes] See the experimental setup section
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Our experiments are deterministic for each model. Instead of running the same model multiple times, we run multiple models at different scales. We are unable to compute error bars for these experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the exper2imental setup section
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See experimental setup section
   (b) Did you mention the license of the assets? [No] The license is permissible for all the assets that we use. The individual licenses can easily be looked up.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] We only use existing datasets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Memory usage compared to 16-bit precision

Table 3 compares the memory footprint of 16-bit inference and LLM.int8() for different open source models. We can see, that LLM.int8() allows to run the largest open source models OPT-175B and BLOOM-176B on a single node equipped with consumer-grade GPUs.
表 3 比较了不同开源模型的 16 位推理和 LLM.int8() 的内存占用情况。我们可以看到，LLM.int8() 允许在配备消费级 GPU 的单个节点上运行最大的开源模型 OPT-175B 和 BLOOM-176B。

Table 3: Different hardware setups and which methods can be run in 16-bit vs. 8-bit precision. We can see that our 8-bit method makes many models accessible that were not accessible before, in particular, OPT-175B/BLOOM.

			Largest Model that can be run	
Class	Hardware	GPU Memory	8-bit	16-bit
Enterprise	8x A100	80 GB	OPT-175B / BLOOM	OPT-175B / BLOOM
Enterprise	8x A100	40 GB	OPT-175B / BLOOM	OPT-66B
Academic server	8x RTX 3090	24 GB	OPT-175B / BLOOM	OPT-66B
Academic desktop	4x RTX 3090	24 GB	OPT-66B	OPT-30B
Paid Cloud	Colab Pro	15 GB	OPT-13B	GPT-J-6B
Free Cloud	Colab	12 GB	T0/T5-11B	GPT-2 1.3B

B Additional Related Work

Quantization of Transformers with fewer than 1B Parameters Quantization of transformers has been focused on sub-billion parameter masked language model (MLMs), including BERT [12] and RoBERTa [29]. Versions of 8-bit BERT/RoBERTa include Q8BERT [56], QBERT [44], product quantization with quantization noise [15], TernaryBERT [60], and BinaryBERT [2]. Work by [61] performs both quantization and pruning. All these models require either quantization-aware finetuning or post-training quantization to make the model usable in low-precision. In contrast with our methods, the model can be used directly without performance degradation.
Transformer 的量化主要集中在十亿分之一参数掩码语言模型 (MLM) 上，包括 BERT [12] 和 RoBERTa [29]。8 位 BERT/RoBERTa 的版本包括 Q8BERT [56]、QBERT [44]、带量化噪声的乘积量化 [15]、TernaryBERT [60] 和 BinaryBERT [2]。[61] 的工作同时执行量化和修剪。所有这些模型都需要量化感知微调或训练后量化，以使模型可用于低精度。与我们的方法相比，该模型可以直接使用而不会降低性能。

If one views matrix multiplication as 1x1 convolution, vector-wise quantization is equivalent to channel-wise quantization for convolution combined with row quantization [25]. For matrix multiplication, this was used by [53] for BERT-sized transformers (350M parameters), while we are the first to study vector-wise quantization for autoregressive and large-scale models. The only other work that we are aware of that quantizes transformers other than BERT is [6], which uses post-training quantization with zeropoint quantization in the forward pass and zeropoint-row-wise quantization in the backward pass. However, this work is still for sub-billion parameter transformers. We compare with both zeropoint and row-wise quantization in our evaluations and do not require post-training quantization.
如果将矩阵乘法视为 1x1 卷积，则向量量化相当于卷积的通道量化与行量化 [25] 相结合。对于矩阵乘法，[53] 将其用于 BERT 大小的 Transformer（350M 参数），而我们是第一个研究自回归和大规模模型的向量量化的人。据我们所知，除了 BERT 之外，量化 Transformer 的唯一其他工作是 [6]，它使用训练后量化，在前向传递中使用零点量化，在后向传递中使用零点行量化。但是，这项工作仍然适用于十亿以下参数的 Transformer。我们在评估中与零点和行量化进行了比较，并且不需要训练后量化。

Low-bitwidth and Convolutional Network Quantization   Work that uses less than 8-bits for data types is usually for convolutional networks (CNNs) to reduce their memory footprint and increase inference speed for mobile devices while minimizing model degradation. Methods for different bit-widths have been studied: 1-bit methods [8, 41, 10], 2 to 3-bit [62, 7], 4-bits [27], more bits [9], or a variable amount of bits [20]. For additional related work, please see the survey of [38]. While we believe that lower than 8-bit width with some performance degradation is possible for billion-scale transformers, we focus on 8-bit transformers that do not degrade performance and that can benefit from commonly used GPUs that accelerates inference through Int8 tensor cores.
使用少于 8 位的数据类型的工作通常用于卷积网络 (CNN)，以减少其内存占用并提高移动设备的推理速度，同时最大限度地减少模型退化。已经研究了不同位宽的方法：1 位方法 [8, 41, 10]、2 到 3 位 [62, 7]、4 位 [27]、更多位 [9] 或可变数量的位 [20]。有关其他相关工作，请参阅 [38] 的调查。虽然我们相信十亿级 Transformer 的宽度低于 8 位可能会出现性能下降的情况，但我们关注的是不会降低性能的 8 位 Transformer，它可以从常用的 GPU 中受益，通过 Int8 张量核加速推理。

Another line of work that focuses on convolutional network quantization is to learn adjustments to the quantization procedure to improve quantization errors. For example, using Hessian information [13], step-size quantization [14], soft quantization [20], mixed-precision via linear programming optimization [54], and other learned quantization methods [58, 18].
另一项专注于卷积网络量化的研究是学习对量化程序的调整以改善量化误差。例如，使用 Hessian 信息 [13]、步长量化 [14]、软量化 [20]、通过线性规划优化实现的混合精度 [54] 以及其他学习到的量化方法 [58, 18]。

C Detailed Outlier Feature Data

Table 4 provides tabulated data from our outlier feature analysis. We provide the quartiles of the most common outlier in each transformer and the number of outliers that are one-sided, that is, which have asymmetric distributions which do not cross zero.
表 4 提供了我们异常值特征分析的表格数据。我们提供了每个 transformer 中最常见异常值的四分位数以及单侧异常值的数量，即具有不交叉零的非对称分布的异常值的数量。

D Inference Speedups and Slowdowns

D.1 Matrix Multiplication benchmarks

While our work focuses on memory efficiency to make models accessible, Int8 methods are also often used to accelerate inference. We find that the quantization and decomposition overhead is significant, and Int8 matrix multiplication itself only yields an advantage if the entire GPU is well saturated, which is only true for large matrix multiplication. This occurs only in LLMs with a model dimension of 4096 or larger.
虽然我们的工作重点是提高内存效率，使模型更易于访问，但 Int8 方法也经常用于加速推理。我们发现量化和分解开销很大，并且 Int8 矩阵乘法本身只有在整个 GPU 充分饱和的情况下才会产生优势，这仅适用于大型矩阵乘法。这仅发生在模型维度为 4096 或更大的 LLM 中。

Detailed benchmarks of raw matrix multiplication and quantization overheads are seen in Table 5. We see that raw Int8 matrix multiplication in cuBLASLt begins to be two times faster than cuBLAS at a model size of 5140 (hidden size 20560). If inputs need to be quantized and outputs dequantized – a strict requirement if not the entire transformer is done in Int8 – then the speedups compared to 16-bit is reduced to 1.6x at a model size of 5140. Models with model size 2560 or smaller are slowed down. Adding mixed precision decomposition slows inference further so that only the 13B and 175B models have speedups.
原始矩阵乘法和量化开销的详细基准测试见表 5。我们发现，在模型大小为 5140（隐藏大小为 20560）时，cuBLASLt 中的原始 Int8 矩阵乘法开始比 cuBLAS 快两倍。如果需要量化输入并反量化输出（如果不是整个 transformers 都以 Int8 完成，则这是一个严格要求），那么与 16 位相比，加速比在模型大小为 5140 时降低到 1.6 倍。模型大小为 2560 或更小的模型会变慢。添加混合精度分解会进一步减慢推理速度，因此只有 13B 和 175B 模型具有加速。

Table 4: Summary statistics of outliers with a magnitude of at least 6 that occur in at least 25% of all layers and at least 6% of all sequence dimensions. We can see that the lower the C4 validation perplexity, the more outliers are present. Outliers are usually one-sided, and their quartiles with maximum range show that the outlier magnitude is 3-20x larger than the largest magnitude of other feature dimensions, which usually have a range of [-3.5, 3.5]. With increasing scale, outliers become more and more common in all layers of the transformer, and they occur in almost all sequence dimensions. A phase transition occurs at 6.7B parameters when the same outlier occurs in all layers in the same feature dimension for about 75% of all sequence dimensions (SDim). Despite only making up about 0.1% of all features, the outliers are essential for large softmax probabilities. The mean top-1 softmax probability shrinks by about 20% if outliers are removed. Because the outliers have mostly asymmetric distributions across the sequence dimension s, these outlier dimensions disrupt symmetric absmax quantization and favor asymmetric zeropoint quantization. This explains the results in our validation perplexity analysis. These observations appear to be universal as they occur for models trained in different software frameworks (fairseq, OpenAI, Tensorflow-mesh), and they occur in different inference frameworks (fairseq, Hugging Face Transformers). These outliers also appear robust to slight variations of the transformer architecture (rotary embeddings, embedding norm, residual scaling, different initializations).
至少 25% 的所有层和至少 6% 的所有序列维度中出现的幅度至少为 6 的异常值的汇总统计数据。我们可以看到，C4 验证困惑度越低，异常值就越多。异常值通常是单侧的，其最大范围的四分位数表明，异常值幅度比其他特征维度的最大幅度大 3-20 倍，通常范围为 [-3.5, 3.5]。随着规模的增加，异常值在 Transformer 的所有层中变得越来越常见，并且它们几乎出现在所有序列维度中。当相同的异常值出现在相同特征维度的所有层中，占所有序列维度（SDim）的约 75% 时，在 6.7B 参数处发生相变。尽管仅占所有特征的约 0.1%，但异常值对于较大的 softmax 概率至关重要。如果删除异常值，平均 top-1 softmax 概率会缩小约 20%。由于异常值在序列维度 s 上大多具有非对称分布，这些异常值维度会破坏对称绝对量化并有利于非对称零点量化。这解释了我们验证困惑度分析中的结果。这些观察结果似乎是普遍的，因为它们出现在使用不同软件框架（fairseq、OpenAI、Tensorflow-mesh）训练的模型中，并且出现在不同的推理框架（fairseq、Hugging Face Transformers）中。这些异常值似乎对 Transformer 架构的轻微变化（旋转嵌入、嵌入范数、残差缩放、不同初始化）也具有鲁棒性。

			Outliers		Frequency			Top-1 softmax p	
Model	PPL↓	Params	Count	1-sided	Layers	SDims	Quartiles	w/ Outlier	No Outlier
GPT2	33.5	117M	1	1	25%	6%	(-8, -7, -6)	45%	19%
GPT2	26.0	345M	2	1	29%	18%	(6, 7, 8)	45%	19%
FSEQ	25.7	125M	2	2	25%	22%	(-40, -23, -11)	32%	24%
GPT2	22.6	762M	2	0	31%	16%	(-9, -6, 9)	41%	18%
GPT2	21.0	1.5B	2	1	41%	35%	(-11, -9, -7)	41%	25%
FSEQ	15.9	1.3B	4	3	64%	47%	(-33, -21, -11)	39%	15%
FSEQ	14.4	2.7B	5	5	52%	18%	(-25, -16, -9)	45%	13%
GPT-J	13.8	6.0B	6	6	62%	28%	(-21, -17, -14)	55%	10%
FSEQ	13.3	6.7B	6	6	100%	75%	(-44, -40, -35)	35%	13%
FSEQ	12.5	13B	7	6	100%	73%	(-63, -58, -45)	37%	16%

These numbers could be improved significantly with optimized CUDA kernels for the mixed precision decomposition. However, we also see that existing custom CUDA kernels are much faster than when we use default PyTorch and NVIDIA-provided kernels for quantization which slow down all matrix multiplications except for a 175B model.
使用针对混合精度分解优化的 CUDA 内核可以显著提高这些数字。但是，我们还发现，现有的自定义 CUDA 内核比使用默认 PyTorch 和 NVIDIA 提供的内核进行量化时的速度要快得多，后者会减慢除 175B 模型之外的所有矩阵乘法。

Table 5: Inference speedups compared to 16-bit matrix multiplication for the first hidden layer in the feed-forward of differently sized GPT-3 transformers. The hidden dimension is 4x the model dimension. The 8-bit without overhead speedups assumes that no quantization or dequantization is performed. Numbers small than 1.0x represent slowdowns. Int8 matrix multiplication speeds up inference only for models with large model and hidden dimensions.
与不同大小的 GPT-3 Transformer 的前馈中第一个隐藏层的 16 位矩阵乘法相比，推理速度有所提升。隐藏维度是模型维度的 4 倍。8 位无开销加速假设不执行量化或反量化。小于 1.0x 的数字表示速度变慢。Int8 矩阵乘法仅对具有较大模型和隐藏维度的模型加速推理。

GPT-3 Size Model dimension	Small 768	Medium 1024	Large 1536	XL 2048	2.7B 2560	6.7B 4096	13B 5140	175B 12288
FP16-bit baseline	1.00x	1.00x	1.00x	1.00x	1.00x	1.00x	1.00x	1.00x
Int8 without overhead	0.99x	1.08x	1.43x	1.61x	1.63x	1.67x	2.13x	2.29x
Absmax PyTorch+NVIDIA	0.25x	0.24x	0.36x	0.45x	0.53x	0.70x	0.96x	1.50x
Vector-wise PyTorch+NVIDIA	0.21x	0.22x	0.33x	0.41x	0.50x	0.65x	0.91x	1.50x
Vector-wise	0.43x	0.49x	0.74x	0.91x	0.94x	1.18x	1.59x	2.00x
LLM.int8() (vector-wise+decomp)	0.14x	0.20x	0.36x	0.51x	0.64x	0.86x	1.22x	1.81x

D.2 End-to-end benchmarks

Besides matrix multiplication benchmarks, we also test the end-to-end inference speed of BLOOM-176B in Hugging Face. Hugging Face uses an optimized implementation with cached attention values. Since this type of inference is distributed and, as such, communication dependent, we expect the overall speedup and slowdown due to Int8 inference to be smaller since a large part of the overall inference runtime is the fixed communication overhead.

We benchmark vs. 16-bit and try settings that use a larger batch size or fewer GPUs in the case of Int8 inference, since we can fit the larger model on fewer devices. We can see results for our benchmark in Table 6. Overall Int8 inference is slightly slower but close to the millisecond latency per token compared to 16-bit inference.

我们与 16 位进行基准测试，并在 Int8 推理的情况下尝试使用更大批量大小或更少 GPU 的设置，因为我们可以在更少的设备上安装更大的模型。我们可以在 Table 6 中看到基准测试的结果。与 16 位推理相比，总体而言，Int8 推理稍慢，但接近每个 token 的毫秒延迟。

除了矩阵乘法基准测试之外，我们还在 Hugging Face 中测试了 BLOOM-176B 的端到端推理速度。Hugging Face 使用具有缓存注意力值的优化实现。由于这种类型的推理是分布式的，因此依赖于通信，我们预计 Int8 推理导致的整体加速和减速会更小，因为整体推理运行时间的很大一部分是固定的通信开销。

Table 6: Ablation study on the number of GPUs used to run several types of inferences of BLOOM-176B model. We compare the number of GPUs used by our quantized BLOOM-176B model together with the native BLOOM-176B model. We also report the per-token generation speed in milliseconds for different batch sizes. We use our method integrated into transformers[52] powered by accelerate library from HuggingFace to deal with multi-GPU inference. Our method reaches a similar performance to the native model by fitting into fewer GPUs than the native model.

Batch Size	Hardware	1	8	32
bfloat16 baseline	8xA100 80GB	239	32	9.94
LLM.int8()	8xA100 80GB	253	34	10.44
LLM.int8()	4xA100 80GB	246	33	9.40
LLM.int8()	3xA100 80GB	247	33	9.11

E Training Results

We test Int8 training on a variety of training settings and compare to 32-bit baselines. We test separate settings for running the transformer with 8-bit feed-forward networks with and without 8-bit linear projections in the attention layer, as well at the attention iteself in 8-bit and compare against 32-bit performance. We test two tasks (1) language modeling on part of the RoBERTa corpus including Books [63], CC-News [33], OpenWebText [19], and CC-Stories [47]; and (2) neural machine translation (NMT) [34] on WMT14+WMT16 [31, 42].
我们在各种训练设置上测试了 Int8 训练，并与 32 位基线进行了比较。我们测试了使用 8 位前馈网络运行 Transformer 的单独设置，注意层中有和没有 8 位线性投影，以及 8 位注意本身，并与 32 位性能进行了比较。我们测试了两个任务 (1) 在 RoBERTa 语料库的一部分上进行语言建模，包括 Books [63]、CC-News [33]、OpenWebText [19] 和 CC-Stories [47]；(2) 在 WMT14+WMT16 [31, 42] 上进行神经机器翻译 (NMT) [34]。

The results are shown in Table 7 and Table 8. We can see that for training, using the attention linear projections with Int8 data types and vector-wise quantization leads to degradation for NMT and for 1.1B language model but not for 209M language modeling. The results improve slightly if mixed-precision decomposition is used but is not sufficient to recover full performance in most cases. These suggests that training with 8-bit FFN layers is straightforward while other layers require additional techniques or different data types than Int8 to do 8-bit training at scale without performance degradation.
结果显示在表 7 和表 8 中。我们可以看到，对于训练，使用 Int8 数据类型和矢量量化的注意力线性投影会导致 NMT 和 1.1B 语言模型的性能下降，但不会导致 209M 语言建模的性能下降。如果使用混合精度分解，结果会略有改善，但在大多数情况下不足以恢复全部性能。这表明使用 8 位 FFN 层进行训练很简单，而其他层需要额外的技术或不同于 Int8 的数据类型才能进行大规模 8 位训练而不会降低性能。

Table 7: Initial results on small and large-scale language modeling. Doing attention in 8-bit severely degrades performance and performance cannot fully recovered with mixed-precision decomposition. While small-scale language models is close to baseline performance for both 8-bit FFN and 8-bit linear projects in the attention layers performance degrades at the large scale.
小型和大型语言建模的初步结果。以 8 位进行注意会严重降低性能，并且性能无法通过混合精度分解完全恢复。虽然小规模语言模型在注意层中的 8 位 FFN 和 8 位线性项目都接近基线性能，但大规模性能会下降。

	Is 8-bit				
Params	FFN	Linear	Attention	Decomp	PPL
209M				0%	16.74
209M	✓			0%	16.77
209M	✓	✓.		0%	16.83
209M	✓	✓		2%	16.78
209M	✓	✓		5%	16.77
209M	✓	✓		10%	16.80
209M	✓	✓	✓	2%	24.33
209M	✓	✓	✓	5%	20.00
209M	✓	✓	✓	10%	19.00
1.1B				0%	9.99
1.1B	✓			0%	9.93
1.1B	✓	✓		0%	10.52
1.1B	✓	✓		1%	10.41

Table 8: Neural machine translation results for 8-bit FFN and linear attention layers for WMT14+16. Decomp indicates the percentage that is computed in 16-bit instead of 8-bit. The BLEU score is the median of three random seeds.
8 位 FFN 和 WMT14+16 的线性注意层的神经机器翻译结果。Decomp 表示以 16 位而不是 8 位计算的百分比。BLEU 分数是三个随机种子的中位数。

Is 8-bit			
FFN	Linear	Decomp	BLEU
		0%	28.9
✓		0%	28.8
✓	✓	0%	unstable
✓	✓	2%	28.0
✓	✓	5%	27.6
✓	✓	10%	27.5

F Fine-tuning Results

We also test 8-bit finetuning on RoBERTa-large finetuned on GLUE. We run two different setups: (1) we compare with other Int8 methods, and (2) we compare degradation of finetuning with 8-bit FFN layers as well as 8-bit attention projection layers comparel to 32-bit. We finetune with 5 random seeds and report median performance.
我们还在 GLUE 上对 RoBERTa-large 进行了 8 位微调测试。我们运行两种不同的设置：(1) 与其他 Int8 方法进行比较，(2) 比较 8 位 FFN 层以及 8 位注意力投影层与 32 位微调的退化。我们使用 5 个随机种子进行微调并报告平均性能。

Table 9 compares with different previous 8-bit methods for finetuning and shows that vector-wise quantization improves on other methods. Table 10 shows the performance of FFN and/or linear attention projections in 8-bit as well as improvements if mixed-precision decomposition is used. We find that 8-bit FFN layers lead to no degradation while 8-bit attention linear projections lead to degradation if not combined with mixed-precision decomposition where at least the top 2% magnitude dimensions are computed in 16-bit instead of 8-bit. These results highlight the critical role of mixed-precision decomposition for finetuning if one wants to not degrade performance.
表 9 与之前不同的 8 位微调方法进行了比较，并表明矢量量化优于其他方法。表 10 显示了 8 位 FFN 和/或线性注意力投影的性能以及使用混合精度分解时的改进。我们发现，8 位 FFN 层不会导致性能下降，而 8 位注意力线性投影如果不与混合精度分解结合使用则会导致性能下降，其中至少前 2% 幅度维度以 16 位而不是 8 位计算。这些结果强调了混合精度分解在微调中的关键作用，如果人们想不降低性能的话。

Table 9: GLUE finetuning results for quantization methods for the feedforward layer in 8-bit while the rest is in 16-bit. No mixed-precision decomposition is used. We can see that vector-wise quantization improve upon the baselines.
GLUE 微调结果显示前馈层的量化方法为 8 位，其余为 16 位。未使用混合精度分解。我们可以看到，矢量量化比基线有所改进。

Method	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Mean
32-bit Baseline	90.4	94.9	92.2	84.5	96.4	90.1	67.4	93.0	88.61
32-bit Replication	90.3	94.8	92.3	85.4	96.6	90.4	68.8	92.0	88.83
Q-BERT [44]	87.8	93.0	90.6	84.7	94.8	88.2	65.1	91.1	86.91
Q8BERT [56]	85.6	93.0	90.1	84.8	94.7	89.7	65.0	91.1	86.75
PSQ [6]	89.9	94.5	92.0	86.8	96.2	90.4	67.5	91.9	88.65
Vector-wise	90.2	94.7	92.3	85.4	96.4	91.0	68.6	91.9	88.81

Table 10: Breakdown for 8-bit feedforward network (FFN) and linear attention layers for GLUE. Scores are median of 5 random seeds. Decomp indicates the percentage that is decomposed into 16-bit matrix multplication. Compared to inference, fine-tuning appears to need a higher decomp percentage if the linear attention layers are also converted to 8-bit.
GLUE 的 8 位前馈网络 (FFN) 和线性注意层的细分。分数是 5 个随机种子的中位数。分解表示分解为 16 位矩阵乘法的百分比。与推理相比，如果线性注意层也转换为 8 位，则微调似乎需要更高的分解百分比。

Is 8-bit											
FFN	Linear	Decomp	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	MEAN
		0%	90.4	94.9	92.2	84.5	96.4	90.1	67.4	93.0	88.6
✓		0%	90.2	94.7	92.3	85.4	96.4	91.0	68.6	91.9	88.8
✓	✓	0%	90.2	94.4	92.2	84.1	96.2	89.7	63.6	91.6	87.7
✓	✓	1%	90.0	94.6	92.2	83.0	96.2	89.7	65.8	91.8	87.9
✓	✓	2%	90.0	94.5	92.2	85.9	96.7	90.4	68.0	91.9	88.7
✓	✓	3%	90.0	94.6	92.2	86.3	96.4	90.2	68.3	91.8	88.7