LPGEMM: 低精度通用矩阵乘法计算模拟框架研究

黄浩岚[1,2,3] 罗铁清[1] 文 梅[2,3] 曹亚松[2,3] 时 洋[2,3]
HUANG Haolan LUO Tieqing WEN Mei CAO Yasong SHI Yang

摘 要 通用矩阵乘（GEMM）算子是AI模型的核心计算，使用低精度数值格式加速GEMM对加速模型的推理和训练有重要影响。由于并不总是有合适的硬件可供选择，而且人们可能希望实验尚未在硬件中实现的新GEMM计算行为，但很难通过构建硬件的方式去进行不同计算行为的GEMM模拟，如何在算子内部进行细粒度模拟还没有被深入研究。通过提出LPGEMM——一个低精度GEMM计算模拟框架来模拟GEMM的计算过程，重新编写了GEMM算子，实现了可变分组累加长度以及低精度累加器，同时还实现了训练和推理全过程的GEMM相关数据统计，来支持用户探索模型精度的下限。实验结果证实了相较于此前的一些工作，所提出的方法模拟最高可减少56%的平均误差。

关键词 深度学习；用户探索模型；通用矩阵乘；低精度

doi：10.3969/j.issn.1672-9528.2024.02.023

0 引言

人们对神经网络表现能力的更高追求，推动了学界对大模型的探索，近年来的模型参数量已增长至千亿级。这同时促使了产业界对包括GPU和FPGA设备在内的高性能神经网络加速器，以及对降低模型的数值精度从而加速计算的追求。

1. 湖南中医药大学 湖南长沙 410208
2. 国防科技大学 湖南长沙 410073
3. 先进微处理器芯片与系统重点实验室 湖南长沙 410073

当前一代的深度神经网络（DNN），例如transformer和ResNet等，在很大程度上依赖于通用矩阵乘法（GEMM），GEMM在DNN模型中的计算占比高达95%以上[1]，GEMM的加速计算对整个模型计算的加速具有重要意义，低精度计算作为压缩与加速深度神经网络（DNN）中最成功的技术之一，被广泛应用于GEMM加速。

由于并不总是有合适的硬件可供选择，而且人们可能希望实验尚未在硬件中实现的新GEMM计算行为（例如用分组累加替代逐元素累加），人们很难通过构建硬件的方式去

[13] KENTON J D M W C, TOUTANOVA L K. BERT: Pre-training of deep bidirectional transformerrs for language understanding[C]//Proceedings of NAACL. Stroudsburg:Association for Computational Linguistics, 2019: 4171-4186.

[14] XU H, LIU B, SHU L, et al. Double embeddings and cnn-based sequence labeling for aspect extraction[C]//Proceedings of the ACL. Stroudsburg:Association for Computational Linguistics, 2018: 592-598.

[15] XU H, LIU B, SHU L, et al. BERT post-training for review reading comprehension and aspect-based sentiment analysis[C]//Proceedings of the NAACL. Stroudsburg:Association for Computational Linguistics, 2019: 2324-2335.

[16] CHEN Z, QIAN T. Enhancing aspect term extraction with soft prototypes[C]//Proceedings of the EMNLP.Stroudsburg:Association for Computational Linguistics, 2020: 2107-2117.

[17] MAO Y, SHEN Y, YU C, et al. A joint training dual-mrc framework for aspect based sentiment analysis[C]//Proceedings of the AAAI. Palo Alto:Association for the Advancement of Artificial Intelligence, 2021: 13543-13551.

[18] WANG Q, WEN Z, ZHAO Q, et al. Progressive self-training with discriminator for aspect term extraction[C]//Proceedings of the EMNLP.Stroudsburg:Association for Computational Linguistics, 2021: 257-268.

【作者简介】

石晓瑞（1993—），女，河南周口人，学士，研究方向：人工智能和实体提取。

（收稿日期：2023-11-09）

进行不同计算行为的 GEMM 模拟，因此人们对模拟低精度 GEMM 的软件越来越感兴趣。通过这种模拟，可以避免构建硬件的开销，研究不同设计下的 GEMM 计算行为。然而，最近提出的一些模拟框架通常只关注数值的模拟，忽略了算子内部的计算行为对结果的影响，这使得软件模拟的粒度较粗，无法精细的控制算子内部的数值计算行为，导致最终模拟结果过于理想，本文在第 4 节解释了为什么忽视内部计算行为会对结果造成影响。

此外，本文评估 LPGEMM 和此前一些工作的模拟可靠性，发现尽管一些工作宣称自己使用了低精度计算，但由于对 GEMM 内部计算行为的忽略，导致 GEMM 内部计算部分和累加格式仍为 FP32，这使得模拟的情况过于理想，降低精度的影响无法得到有效的评估。这个问题可能会使已发表文献中的某些结论无效。LPGEMM 重新设计了 GEMM 累加器，增加了逐操作量化模块，细粒度模拟了降低精度对累加带来的额外淹没现象。

为了促进相关研究，本文设计了 LPGEMM，一个低精度 GEMM 计算模拟模块。在设计 LPGEMM 之初，本文有以下几个目标。

（1）重视模拟的性能，对于许多模拟框架的目标来说，模拟性能十分重要，本文在第五节讨论了为缩小模拟计算和实际运算的性能差距所做的工作。

（2）重视模拟的准确性，本文的目的是专注支持对 GEMM 以及可被映射为 GEMM 运算的算子计算行为的细粒度模拟。如今的低精度训练研究通常不关注算子内部计算的模拟，导致模拟时 GEMM 使用理想累加器进行累加操作。这使得模拟的结果相较实际过于理想。

（3）研究者们通常还关心模型不同层间数值的统计学指标，LPGEMM 实现了对推理和训练过程中各项指标逐层的统计，如“淹没”数、模型各层的均值、方差与数据分布。通过提供充分的统计数据，来支持用户探索 GEMM 的最佳配置。

GEMM 的 GPU 实现十分复杂，内部优化是多维度、多层次的。要在算子内部实现分组累加、低精度模拟与数据统计是遇到的首要挑战。

本文的创新贡献如下。

（1）提出并实现了一种可分组，支持低于 FP32 精度任意位宽模拟的 GEMM 算子，内部计算格式可采用 FP32 或 FP16，并设计了算子内部量化模块对每次乘加操作量化，这相当于直接使用对应低精度累加器累加。

（2）LPGEMM 允许用户对各个层指定的不同的浮点数值格式来进行计算，并实现了推理和训练过程中以层为粒度的各项指标统计，如“淹没”数占比，各层的均值、方差及方差保留率（variance retention rate，VRR）。通过提供充分的统计数据，来支持用户以层为单位探索不同层不同配置 GEMM 对计算的影响。

（3）评估了 LPGEMM 在不同规模矩阵上的表现，实验证实了 LPGEMM 在低精度 GEMM 的模拟中有出色的表现。比社区流行的量化模拟框架最多降低约 56% 的平均误差。

文章的其余部分结构如下。

在第 2 节和第 3 节介绍了 LPGEMM 的背景和相关工作；在第 4 节中，分析了 GEMM 的误差来源和计算模拟，在第 5 节中详细解释了所提出的 LPGEMM 模块功能以及它是如何在 PyTorch 中工作的；在第 6 节中，展示并分析了实验结果；在第 7 节中，分享了本文的结论并展望了未来。图 1 中现象由于操作数 S_{i-1} 与 X_i 的指数差异而导致 X_i 的尾数位部分或完全截断，称为淹没。图 2 中 transformer 的核心多头注意力算子（multi-head attention）是由多个（c）自注意力（self-attention）算子组成，（c）自注意力（ self-attention）算子的计算核心是两个 GEMM 计算。

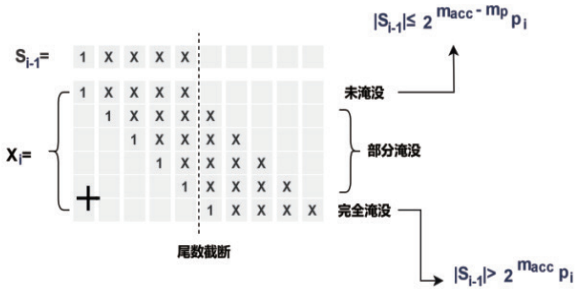


图 1 4 位数浮点数加法的淹没现象

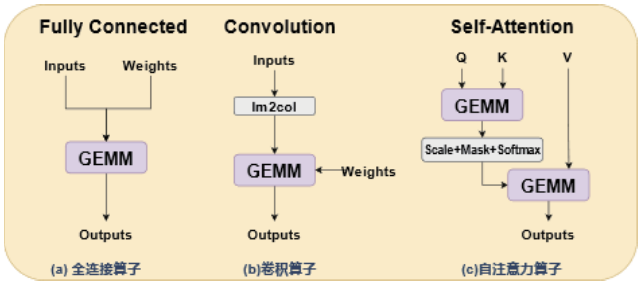


图 2 包含 GEMM 的算子结构

1 背景

本节首先介绍了浮点数的表示和“淹没”数的定义、GEMM 以及低精度计算。

低精度浮点数的表示：b 位浮点数 a 有 1 个符号位、e 个指数位和 m 个尾数位，因此 b=1+e+m。这样的数称为 (1, e, m) 浮点数[2]。缩写 FP8、BF16、TF32、FP16、FP32 和 FP64 分别表示格式 binary8、bfloat16、TensorFloat-32、binary16、binary32 和 binary64。它们的浮点格式表示如表 1 所示。

表 1 浮点精度的(1, *e*, *m*) 表示

浮点精度	浮点格式
FP64	(1,11,52)
FP32	(1,8,23)
TF32	(1,8,10)
FP16	(1,5,10)
BF16	(1,8,7)
FP8	(1,4,3) (1,5,2)

淹没数：构成 GEMM 的基本计算函数是向量点积。向量点积的计算方式是乘累加（multiply-accumulate，MAC），这需要乘法和加法两种浮点运算，理想浮点运算的实现需要输出处有一定的冗余位以避免信息丢失。但在实际 GEMM 计算时，所有操作数的精度一般相同，例如，在典型的 MAC 运算中，如果 $c=c+a\times b$，由于 c 的精度与 a、b 相同，因此需要在计算后立即舍入。这种舍入可能会导致操作数在加法中被完全或部分截断，图 1 以 4 位尾数的浮点数为例展示了这种现象，这种现象称为“淹没”[3]，发生“淹没”现象的数被称为淹没数。操作数的精度越低，“淹没”现象出现越频繁，在神经网络的训练中，梯度乘以学习率后通常是一个很小的数，这使得低精度训练中的“淹没”现象更为常见，这是误差累积的主要来源。

通用矩阵乘法（general matrix multiplication，GEMM）是神经网络的核心计算，如图 2 所示，全连接算子、卷积算子以及自注意力算子均包含 GEMM 计算。

2 相关工作

低精度浮点计算模拟：目前有很多通过软件来模拟低精度浮点计算的软件包。表 2 对这些软件包进行了总结，目前最全面的模拟任意精度浮点运算的软件是 GNUMPFR 库 [4]，它将 IEEE754 标准中的格式扩展到任何精度和任意大的指数范围。GNUMPFR 通常用于模拟需要高精度的计算。

表2 浮点模拟计算软件包

软件包	存储精度	内部计算精度	指定GEMM 行为
GNU MPFR	订制		×
NVIDIA Apex	FP16/FP32	FP32	×
CPFloat	FP32/FP64	FP16/BF16/TF32	×
chop	FP32/FP64		×
FASE	FP32		×
QPytorch	FP32	FP32	×
TensorQuant	FP32	FP32	×
LPGEMM	FP16/FP32	FP16/BF16/FP32	√

CPFloat[5] 是一个用 C 语言编写的低精度浮点计算模拟库，旨在高效且全面地进行浮点模拟，用于原型设计和测试混合精度算法，以及模拟定制精度硬件。CPFloat 并不直接支持神经网络及相关算子。

NICHOLAS J. 和 Higham 等提出了 Chop[6]，这是一个 MATLAB 函数，用于将 FP32 或 FP64 精度的数组舍入到 FP16，BF16 或其他低精度。该函数只能在 MATLAB 编程环境中使用，并且底层算法依赖于 MATLAB 所表示浮点数的数学运算，不一定适合神经网络及相关模拟。

Apex[12-13] 是 NVIDIA 公司推出的低精度训练软件包，用于简化 PyTorch 中的混合精度和分布式训练。Apex 只需几行代码即可将 FP32 模型代码修改为低精度或混合精度模型代码，Apex 只支持 FP16 与 FP32 格式，并且 Apex 不支持 GEMM 内部量化，GEMM 累加时仍使用 FP32 格式。

FASE[7] 利用动态二进制转换的方法来改变可执行文件的数值精度，而不需要对源代码进行任何修改。这种方法的缺点是，用户只能全局设置某一目标精度，不能细粒度地在每个操作上改变目标精度，这大大限制了模拟范围。例如不能模拟混合精度的操作，这在硬件设备中变得越来越普遍。

低精度模拟框架：QPyTorch 是一个基于 PyTorch 的低精度计算模拟框架，其主要目标是促进低精度神经网络的训练，而无需构建低精度硬件的开销 [8]。该库是目前低精度神经网络模拟计算中最全面的框架，其内部存储与计算精度是 FP32，所有低于 FP32 精度的低精度数会舍入到目标精度后用 FP32 存储及计算。这可能导致对如 FP8 等更低精度浮点数的模拟计算得出过于理想的结果。此外 QPyTorch 也不能指定 GEMM 内部的计算行为。

另一个基于 TensorFlow 设计的低精度模拟框架是 TensorQuant[9]，其允许在训练和推理过程中对现有 DNN 拓扑进行透明的量化模拟。TensorQuant 支持通用量化方法，并允许通过实验评估量化对单层以及整个拓扑的影响。与 QPyTorch 一样，TensorQuant 不能指定 GEMM 内部计算行为。

3 GEMM 计算模拟分析

当前一代的深度神经网络在很大程度上依赖于 GEMM，GEMM 在 DNN 模型中的计算占比高达 95% 以上，随着 transformer 的流行，该占比进一步提升，此前有许多研究致力于低精度浮点计算模拟，这些研究忽视了算子内部的计算行为，进而忽略了浮点误差的主要来源之一——GEMM 累加器 [10]。假设 GEMM 累加器中有一待累加的正数序列为 $X=\{x_1,x_2,\ldots,x_n|(x_i\geq 0)\}$，其累加和 $S_n=x_1+x_2+\ldots+x_n$。随着累加的进行，S_i 不断增大。如图 1 所示，m_{acc} 代表累加器精度尾数位，m_x 代表操作数精度尾数位，假设累加器精度尾数位总是大于操作数精度尾数位，当 S_i 与 x_i 满足时，就会出现淹没现象。

$$|S_i-1|>2^{m_{acc}-m_x}x_i \tag{1}$$

显然，m_{acc} 越接近 m_x，$2^{m_{acc}-m_x}x_i$ 越小，累加中的淹没

现象出现越频繁。对于使用 FP32 作为累加器精度的模拟框架，这相当于在模拟低精度 GEMM 时使用理想累加器，低精度数值格式对 GEMM 累加的损害并不会体现。为了模拟这种内部误差，LPGEMM 设计了一个累加器量化模块，具体设计将会在第 5 节论述。

$$V = \frac{Var(S_n)_{swamping}}{Var(S_n)_{ideal}} \tag{2}$$

如公式（2）所示，对于一组服从相同概率分布（例如正态分布）的独立随机变量，令$Var(Sn)_{swamping}$为实际求和序列的方差，$Var(Sn)_{idea}$为理想求和序列（即相当于理想累加器求和）的方差，它们之比称为方差保留率（variance retention rate，VRR），由于实际求和序列可能会出现淹没现象，因此有：

$$Var(Sn)swamping \leq Var(Sn)ideal \tag{3}$$

这使得 $V \leq 1$，当 V 越接近 1，实际累加结果越接近理想累加结果。

分组相乘后累加有益于减少淹没现象，令累加序列为 $\{X_i\}_{i=1}^{n}$，其中 $n=n_1 \times n_2$ 为分组数，n_2 为累加长度，这样就将累加序列分为 n_1 组长度为 n_2 的子累加序列。每一组子序列进行内部累加，得到的 n_2 个中间结果再累加得到最终结果 S_n，这种简单的技术可以极大地提高总和的稳定性，同时减少淹没现象。分组乘累加的公式为：

$$V = VRR(m_{acc}, m_p, n_1) \times VRR(m_{acc}, min(m_{acc}, m_p + log_2(n_1)), n_2) \tag{4}$$

式中：V 表示 VRR。

4 LPGEMM

此前有许多研究致力于低精度浮点计算模拟，本文研究将 GEMM 的内部行为模拟引入低精度浮点计算模拟中，使模拟的粒度更细。在本文中提出并实现了一种名为 LPGEMM 的低精度 GEMM 模拟计算框，如图 3 所示。

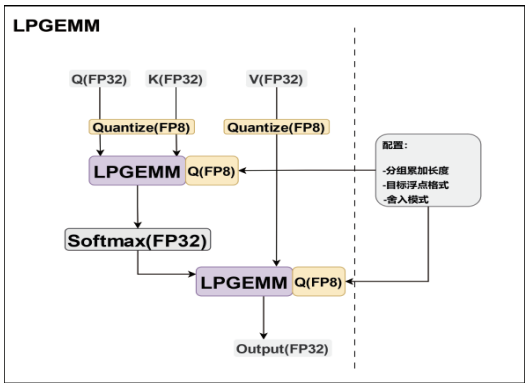


图 3 LPGEMM 的核心流程

LPGEMM 的核心流程如下。

（1）读取用户的配置，用户的配置示例如表 3 所示。

表 3 LPGEMM 配置入参模板

	参数
1	groupedCumulativeLength = 16
2	targetFloatType = (1,5,2)
3	roundingType = "nearest"
4	quantLayer = [conv1,conv2,FC1,FC2]

（2）将 GEMM 的输入矩阵表示为用户指定的低精度浮点数。

（3）依照配置参数设置 GEMM 的分组累加长度，通过单精度浮点或半精度浮点（取决于低精度表示后数据精度是否低于半精度，如果低于半精度则用半精度浮点存储）计算来模拟低精度 GEMM。在 GEMM 累加器中设计一个低精度量化模块来去除每次累加时额外的精度。该模块会检查累加前后的操作数并量化。对于 LPGEMM 的首要目标是模拟的准确性。

另外，本文十分注重模拟计算的速度，更快的模拟速度使大规模的真实模型模拟成为可能，由于 LPGEMM 的细粒度模拟需要比以往低精度模拟框架更多的性能开销，为了减少这些开销带来的影响，本节介绍了在缩小模拟计算和实际设备计算的性能差距做的许多工作，并实验得到了 LPGEMM 在不同规模矩阵上的计算时间。

4.1 LPGEMM 功能

数值低精度格式化：LPGEMM 可以模拟多种低精度浮点格式，表 1 展示了部分支持格式，具体来说，LPGEMM 可以模拟任何使用比单精度位数更少的浮点格式（即那些少于 8 位的指数和少于 23 位的尾数的浮点格式）。其中 BF16 和 FP16 格式被直接实现而非模拟。LPGEMM 在累加器中增加了量化模块，该量化模块会在累加器每次乘法或加法操作时截断操作数至目标精度。需要强调的是，LPGEMM 只在 GEMM 及可被映射为 GEMM 的算子中进行浮点模拟，而不是关注所有神经网络算子。

LPGEMM 的核心流程如图 3 所示，以 FP8 精度的自注意力算子为例展示 LPGEMM 的核心流程，其中配置由用户指定，Q(FP8) 表示 LPGEMM 的累加器量化模块。

可配置分组累加长度：分组乘积累加可以减少低精度计算中的“淹没”现象，但不同的模型的最佳配置可能对应不同的分组累加长度，基于这种情况，LPGEMM 实现了 GEMM 的分组累加长度可配置，用户可以指定分组累加的长度来探索 GEMM 的最佳配置。

分层量化与数据统计：神经网络中的不同层量化敏感度不同 [11]，对应最低所需精度和累加长度不同，可能需要指定不同的精度和累加长度。LPGEMM 支持用户分层指定模型各层的 GEMM 算子的分组累加长度和精度格式。另外为了更好地支持用户探索不同层的量化配置带来的计算影响，针对每层的量化配置，LPGEMM 会自动记录模型不同层的统计数据，

这些统计数据类型包括淹没数、数据分布、方差、均值和等数据。

PyTorch 集成：LPGEMM 提供了一个方便的接口来独立处理不同的数值格式，允许搜索不同精度和累加长度的更有效组合。具体来说，设计了一个自动注入量化模块，用户只需使用少量的代码对 GEMM 的参数进行配置，而不需要单独的低精度模型定义就可进行 GEMM 模拟。

4.2 LPGEMM 性能

想要在 GEMM 的模拟计算中获得接近真实设备性能是很困难的，这种困难主要来源于两方面。

（1）用户一般希望模拟当前硬件中不存在的数值格式，因为对于当前硬件中已支持的数值格式，用户可以直接使用无需模拟，这需要额外的量化开销。如 QPyTorch，TensorQuant 等低精度模拟框架为了减小这部分开销选择在神经网络算子前后进行量化，称为外部量化。外部量化省下了大量量化开销，虽然保证了量化速度，但损失了准确性。以 GEMM 为例，外部量化无法模拟 GEMM 内部乘加运算的量化误差，内部使用高精度累加器累加，导致结果过于理想。

LPGEMM 的目标之一是模拟的准确性，采用内部量化，但会极大增加量化开销。

（2）为了保证模拟 GEMM 计算行为的可配置性，不同的配置所带来的空间局部性有较大差异，其并行性也有非常大的差异，不同配置构成的搜索空间巨大，针对每一种配置进行特定的并行优化并不现实。

为了缩小模拟计算和实际设备计算的性能差距，对于（1）中提到的问题，使用 CUDA 语言重写了 GEMM 算子，并重新设计了累加器量化模块，使量化直接在寄存器内部完成，从而加速这部分量化带来的计算开销。对于（2）所提到的问题，LPGEMM 基于特定分组数 GEMM 方法优化并推广到其他分组数，LPGEMM 首先针对分组为 1（不分组），8 和 16 的 GEMM 进行了并行性能优化，基于这三种分组 GEMM 进而实现分组可配置的 GEMM，实现了可配置性与性能的平衡。

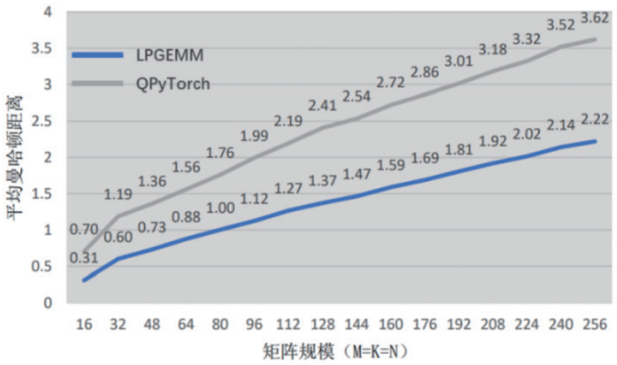
5 实验

5.1 实验平台

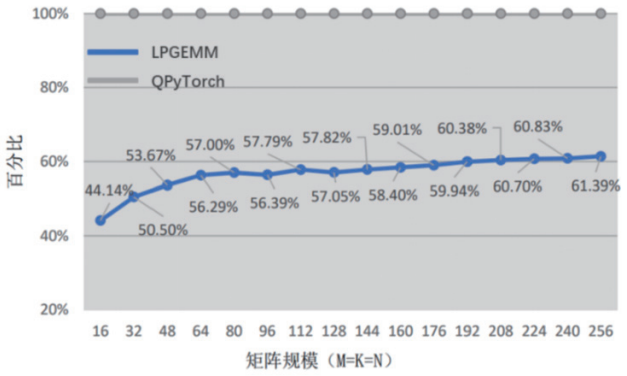
本实验完成了 LPGEMM 框架在不同规模矩阵下的误差与性能评估。运行 LPGEMM 的测试平台是一台配备 CentOS 操作系统的服务器，GPU 为 NVIDIA A100。为了与真实的低精度硬件进行对比，使用 FPGA 实现了纯 FP8 精度矩阵乘法加速器（加速器的设计不在本文的讨论范围之内）。

5.2 模拟误差对比

对于低精度 GEMM 的模拟来说，矩阵的平均曼哈顿距离是十分重要的指标，它表示模拟精度计算与真实精度计算的平均误差。为了与真实低精度计算对比，额外实现了基于 FPGA 的 FP8(1,5,2) 矩阵乘，实验结果如图 4 所示。



（a） 矩阵的平均曼哈顿距离



（b）归一化后的平均曼哈顿距离

图 4 LPGEMM 和 QPyTorch 在不同矩阵规模上的误差比较

实验选取的比较对象是 QPyTorch——最近社区流行的低精度量化框架，首先初始化了这些不同规模的矩阵，然后使用 LPGEMM 与 QPyTorch 量化矩阵至目标精度进行计算，最后用 Mat_l 和 Mat_q 分别代表 LPGEMM 与 QPyTorch 的结果矩阵，用 Mat_{FP8} 代表 FPGA 的 FP8 矩阵乘法，则 LPGEMM 相比真实 FP8 计算结果的平均曼哈顿距离为 $\overline{D}_{(Mat_l,Mat_{FP8})}$，QPyTorch 为 $\overline{D}_{(Mat_q,Mat_{FP8})}$。

表 4 为 LPGEMM 和 PyTorch-GEMM 在不同矩阵规模上计算时间的比较，矩阵使用 PyTorch 的“randn”函数生成，没有使用特殊参数，单位为毫秒（ms），第二列为 LPGEMM 启用低精度模拟的计算时间，第三列为 LPGEMM 直接计算所用时间，第四列表示 PyTorch-GEMM 的计算时间。

表 4 LPGEMM 和 PyTorch-GEMM 在不同矩阵规模上计算时间的比较

矩阵规模（$M=K=N$）	LPGEMM with simulation	LPGEMM without simulation	PyTorch GEMM
1024	0.55	0.14	0.12
2048	2.2	0.58	0.49
3072	4.97	1.24	1.01
4096	17.65	2.62	2.02
5120	21.9	4.54	3.83
6144	47.78	8.33	6.21

表 4（续）

矩阵规模（M=K=N）	LPGEMM with simulation	LPGEMM without simulation	PyTorch GEMM
7168	58.57	12.5	9.7
8192	82.21	20	13.96
9216	110.44	27.25	19.71
10 240	153.12	36.15	27.23

图 4（a）展示了平均曼哈顿距离随矩阵规模的变化，图 4（b）以 QPyTorch 的计算结果作为归一化基准，显示 LPGEMM 在 FP8(1,5,2) 精度模拟上相比 QPyTorch 最多减少了约 56% 的平均误差，平均减少 43.1%。

5.3 LPGEMM 性能评估

LPGEMM 的主要目的是提供评估低精度格式与分组累加对计算结果影响的深度学习框架，经过的调查，并未发现有相同类型的工作，在不同规模矩阵上对比了 LPGEMM 和 PyTorch 的 GEMM 计算时间差距，如表 4 所示。相较于 PyTorch GEMM，LPGEMM 在累加器中增设了逐操作量化模块，并且实现了通用的分组累加，在分组为 16，量化精度为 FP32（不量化）时，LPGEMM 平均性能为 PyTorch 的 0.79×，当分组为 16，量化精度为 BF16 时，LPGEMM 平均性能为 PyTorch GEMM 的0.17×。

6 结论与展望

本文中提出了一个 GEMM 低精度计算模拟框架 LPGEMM。LPGEMM 针对 GEMM 的低精度计算研究，促进对各种浮点数字格式、分组选择的研究。LPGEMM 通过重写 GEMM 内核，提供了准确且细粒度的 GEMM 模拟。

在 FP8 的模拟实验中进行了 LPGEMM 的误差评估，验证了 LPGEMM 相比 QPyTorch 最多 56%，平均能获得 43% 左右的平均误差减少。在保证准确性的同时，在不同规模的矩阵进行了性能测试，LPGEMM 最高性能为 PyTorch 的 0.79×。未来将进一步发展 LPGEMM，优化模拟的性能并进一步提升 GEMM 模拟的准确性。

参考文献：

[1] JIA Y . Learning semantic image representations at a large scale[EB/OL].(2014-07-05)[2023-09-26].https://www.semanticscholar.org/paper/Learning-Semantic-Image-Representations-at-a-Large-Jia/13a5b779b46b507f59df866e369ac4e78388240a.

[2] SAKR C,WANG N,CHEN C Y,et al. Accumulation bit-width scaling for ultra-lowprecision training of deep networks[EB/OL].(2014-07-05)[2023-09-26].2019. https://arxiv.org/abs/1901.06588.

[3] FOUSSE L , HANROT G , LEFEVRE V ,et al.MPFR: A multiple-precision bi- nary floating-point library with correct rounding[J].Information tech- nology & internet business report for the 21st century, 2007, 33(2):510- 515.

[4] FASI M, MIKAITIS M. CPFloat: A C library for simulating low-precision arithmetic[J]. ACM transactions on mathematical software,2020,18(2):181-183.

[5] HIGHAM N J , PRANESH S .Simulating low precision floating-point arithmetic[J].SIAM journal on scientific computing, 2019,41(5):20-25.

[6] OSORIO J, ARMEJACH A, PETIT E, et al. FASE: A fast, accurate and seam- less emulator for custom numerical formats[C]//2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Piscataway:IEEE, 2022: 144-146.

[7] ZHANG T, LIN Z, YANG G, et al. Qpytorch: A low-precision arithmetic simulation framework[C]//2019 Fifth Workshop on Energy Efficient Ma-chine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS). Piscataway:IEEE, 2019: 10-13.

[8] LOROCH D M, PFREUNDT F J, WEHN N, et al. Tensorquant: A sim- ulation toolbox for deep neural network quantization[EB/OL].(2017-10-13)[2023-09-27].https://arxiv.org/abs/1710.05758.

[9] ROBERTAZZI T G , SCHWARTZ S C .Best ordering for floating-point addition[J].ACM Transactions on Mathematical Software (TOMS), 1988,14(1):101-110.

[10] CASTALDO A M , WHALEY R C , CHRONOPOULOS A T .Reducing float- ing point error in dot product using the superblock family of al- gorithms[J].Siam journal on scientific computing, 2015, 31(2):1156- 1174.

[11] CAI Y, YAO Z, DONG Z, et al. Zeroq: a novel zero shot quantization framework[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2020:13166-13175.

[12] CARL C.Mixed Precision Training of Deep Neural Networks[EB/OL].(2019- 03-15)[2023-07-06]. https://www.nvidia.com/en-us/about-nvidia/cookie-policy/.

[13] MICHAEL C.Automatic Mixed Precision in Py-Torch[EB/OL].(2019-03-15)[2023-07-07]. https://pytorch.org/tutorials/recipes/recipes/amp_recipe.html.

[14] NICHOLAS H.The accuracy of floating point summation[J]. SIAM journal on scientific computing, 1993,7(1):783-799.

【作者简介】

黄浩岚（1997—），男，湖南郴州人，硕士，研究方向：深度学习加速器。

（收稿日期：2023-12-01）