MAx-DNN: 多级算术近似用于节能的 DNN 硬件加速器

Vasileios Leon*, Georgios Makris*, Sotirios Xydis†, Kiamal Pekmestzi*, Dimitrios Soudris*

*National Technical University of Athens, School of Electrical and Computer Engineering, 15780 Athens, Greece

†Harokopio University of Athens, Department of Informatics and Telematics, 17778 Athens, Greece

摘要—如今，深度神经网络（DNN）架构的快速增长使其成为提供高级机器学习任务并具有出色准确性的实际方法。针对低功耗 DNN 计算，本文探讨了细粒度错误容限的 DNN 工作负载与硬件近似技术之间的相互作用，以实现更高水平的能量效率。利用最先进的 ROUP 近似乘法器，我们系统地探索了它们在网络中的细粒度分布，根据我们的层、滤波器和内核级别方法，并考察其对准确性和能量的影响。我们使用 CIFAR-10 数据集上的 ResNet-8 模型来评估我们的近似值。提出的解决方案与基线量化模型相比，在最多 4%的精度损失下可获得高达 54%的能量增益，同时在与最先进的 DNN 近似方法相比时提供了更好的精度和 2× 的能量增益。

***Index Terms*—近似计算，不精确乘法器，专用集成电路，深度神经网络，残差网络，CIFAR-10，TensorFlow。**

I. 介绍

数字信号处理（DSP）和人工智能（AI）领域中复杂工作负载的激增标志着集成电路和嵌入式系统设计的新时代。为了在有限的功率范围内实现高性能计算，研究界正在探索替代的设计策略。在此方向上，近似计算是一种新兴的设计范式 [1]，它利用容错性来以牺牲精度换取功耗、能源、面积和/或延迟方面的收益。在不精确硬件领域中，近似技术被应用于不同层次，即从电路 [2]–[5] 到加速器 [6]–[10]。

深度神经网络（DNN）的内在错误鲁棒性和增加的计算需求，作为最先进的 AI 方法来解决计算机视觉任务（例如图像分类、目标检测、姿态估计），使其成为近似计算的有前景候选者。在 DNN 上的典型近似技术包括不精确算术运算符 [7]、低精度数值格式 [11]、权重量化 [12] 以及神经元连接剪枝 [13]。用于 DNN 推理的近似硬件加速器 [7]–[9] 因其吸引人的性能功耗比而逐渐受到重视。此外，ASIC/FPGA 技术通过允许低级优化和自定义数据路径位宽，促进了近似计算的发展，这与通用 GPU/CPU 解决方案相反。

近似 DNN 加速器的设计带来了若干挑战。正如在 [7] 中讨论的，研究目标之一是避免重新训练以减少准确性损失。原因有两点：(i) 专有的数据集/模型可能不可用，(ii) 由于硬件近似的仿真以及使用自定义近似算术运算符而导致的训练时间增加。在近似 DNN 中的另一个重要挑战是近似定位，即在哪里插入近似值，以最大限度地提高能效同时保持可接受水平的准确性。

在本文中，我们介绍了 MAx-DNN 框架来解决上述设计挑战，通过检查 DNN 架构不同层级中的近似处理，即在网络层、层的滤波器或滤波器的核心内。考虑到 DNN 中的大多数计算都是乘法-累加 [9] 运算，我们专注于优化这些乘法运算。MAx-DNN 采用了 ALWANN [7] 来生成不需要重新训练的近似 DNN 硬件加速器，并通过我们的近似定位技术进行了扩展，该技术基于 ROUP 乘法库 [2]。与原始的 ALWANN 框架相比，我们的设计空间更大，在每个卷积层中引入了相同的不精确乘法器（不同层间的乘法器可能不同）。有趣的是，所提出的近似方法为每个卷积层、滤波器或核心分配不同的近似 ROUP 乘法器。为了评估我们的近似处理，我们使用在 CIFAR-10 数据集上训练的量化 ResNet-8 网络，并采用标准单元 ASIC 技术以及 TSMC 45 纳米库。首先，我们在独立和组合方式下检查 DNN 层对近似的敏感性，然后进行广泛的设计空间探索及精度-能耗帕累托分析

以提取我们方法中最突出的配置。结果表明，与量化模型相比，我们的设计在牺牲 4%准确度的情况下可获得高达 54%的能量增益，并且相较于默认的 ALWANN 近似处理提供了 2× 倍的能量效率。本文的贡献在于提出了一种改进的细粒度方法来近似 DNN 中的乘法运算，以及量化了来自各种近似方法和配置的结果。

本文的其余部分组织如下。第二节介绍了 ALWANN 框架 [7]。第三节定义了我们的逼近空间。第四节报告了实验评估。最后，第五节得出了结论。

II. ALWANN 近似框架

ALWANN 框架 [7] 接受以下输入：(i) 以 protobuf 格式训练（冻结）的网络模型，(ii) 近似乘法器库，以及 (iii) 硬件加速器的架构约束（例如，流水线或电源门控模式，近似单元的数量）。它假设加法准确且乘法近似，以及每层卷积只有一个近似类型。此外，为了在不重新训练的情况下提高准确性，根据乘法器的属性调整/更新网络权重。这些近似的网络，标记为 AxNNs，是初始模型的修改版本，并满足用户对加速器架构的约束。

为了启用 ALWANN，扩展 TensorFlow 以支持近似量化层，通过创建一个新的操作符来实现，该操作符将常规的量化卷积 2D 层替换为卷积 2D 层。此操作符允许通过 *AxMult(字符串)* 参数指定要使用的近似乘法器（需要乘法器的 C/C++模型），并且可选地使用权重调整功能，通过 *AxTune(布尔值)* 实现。冻结的模型由 TensorFlow 的变换图形工具处理，该工具插入了 `AxConv2D` 层，然后，通过 NSGA-II 算法提取帕累托最优 AxNNs。

在本文中，我们为 ALWANN 配备了新的近似方法，涉及网络中近似乘法的分布。扩展框架版本的工具流程如图 1所示。与 ALWANN 相比，我们将轴卷积 2DTensorFlow 操作符修改为支持我们在三个不同层次（层、滤波器、内核）上的近似方法，并且还采用了最先进的 ROUP 库中的近似乘法器 [2]。

III. 定义近似空间

本节中，我们讨论了所提出的近似空间，解释了如何在 DNN 的每一层、滤波器或内核中分配近似的乘法器。对于我们的分析，假设典型的 DNN 架构：每个卷积层包括 N 个滤波器，每个滤波器由 M 个卷积核组成以处理 M 个输入通道并输出一个特征图（总共生成 N 个输出特征图）。近似并不是在所有被检查的级别上均匀应用，也就是说，每一层/滤波器/内核都被分配了一个不同的 ROUP 乘法器。最后，我们介绍了 ROUP 乘法器族 [2] 和用于估计能量的模型。

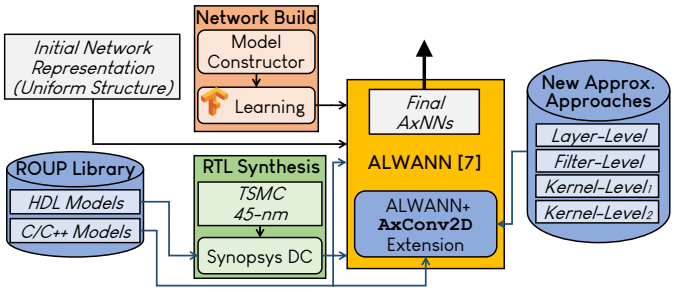


图 1. MAx-DNN 工具流程与架构，ALWANN 的扩展 [7]。

A. *LLAM: 层级近似乘法*

第一种方法，如图 2a所示，旨在在网络中插入不同强度的近似值。具体来说，我们对每一卷积层的所有乘法操作使用 ROUP 乘法器进行处理。这种方法是传统的做法，其中通过为每个卷积层分配一个近似的乘法器，无论是相同的（均匀分布）还是不同的（非均匀分布），来在网络中分散这些近似值。

B. *FLAM: 滤波级近似乘法*

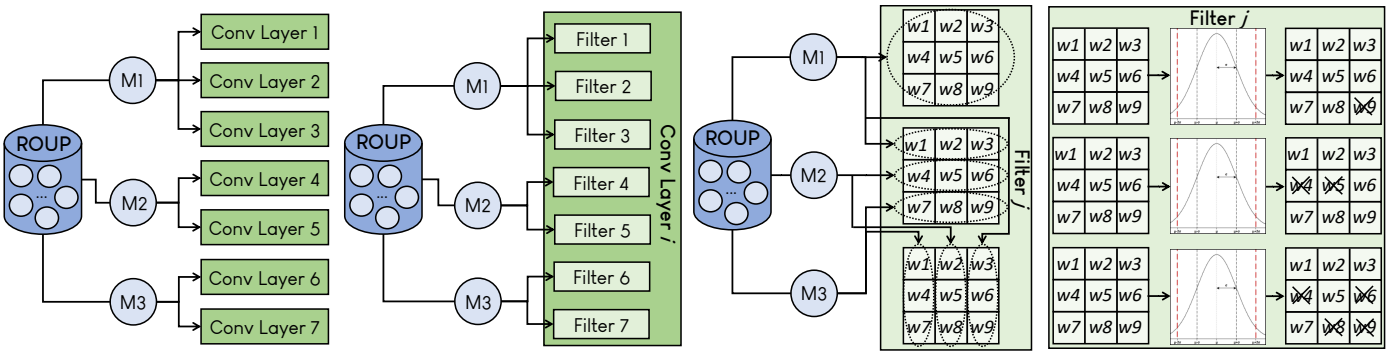
类似 LLAM 的方法，我们的第二种方法如图 2b所示，在 DNN 的每个卷积层中为滤波器创建不同近似值。具体来说，我们在每一层的各个滤波器中使用不同的 ROUP 乘数因子，这与第一种方法相反——在第一种方法中，同一层的所有滤波器均采用相同的 ROUP 乘数因子。该方法通过创建滤波器组并将它们分配对应的 ROUP 乘数因子来实现。

C. *KLAM: 内核级近似乘法*

在第三次近似方法中，我们在 DNN 架构中进一步深入，并将乘法分别对应每个卷积核进行近似。结果是，每个滤波器的卷积操作使用不同的 ROUP 乘数执行。图 2c描述了该方法提出的三种变体：通道变体，其中与内核对应的全部乘法操作都使用相同的乘数进行，以及行/列变体，其中对于内核的每一行/列使用不同的乘数。

D. *KLMS: 内核级乘法跳过*

我们的最后一种方法，如图 2d所示，基于权重值跳过每个卷积的一些乘法。更具体地说，我们计算每层中所有核权重的平均值，并仅执行与接近该平均值的权



(a) LLAM (b) 火焰 (c) 卡 lam (d) KLMS

图 2. 提出的非均匀逼近方法在不同层次上的应用：(a) 层级，(b) 滤波器层级& (c) 内核层级近似乘法，以及 (d) 内核层级乘法跳过。

重对应的乘法。假设均值为 μ，标准差为 σ，我们选择仅执行属于区间 $[\mu-\sigma,\mu+\sigma]$ 或 $[\mu-2\sigma,\mu+2\sigma]$ 内的权重的乘法。

E. ROUP 近似乘法器

ROUP 多乘器近似系列 [2] 基于 4 进制编码，并应用了两种正交近似技术，即，非对称舍入和穿孔。在此混合近似中，穿孔省略了最不重要部分积的生成，而不对称舍入将每个剩余的部分积四舍五入到不同的位宽，这取决于它在部分积矩阵中的重要性。

两个 N 位的补码数的 ROUP 乘法是通过累积未打孔且已四舍五入的部分积来完成的，即：$\tilde{P}_j$，这些部分积是基于基数-4 编码 [2] 生成的：

$$\mathrm{ROUP}(A,B)=\sum_{j=P}^{N/2-1}\tilde{P}_j 4^j=\sum_{j=P}^{N/2-1}A_j^r\cdot b_j^{\mathrm{R4}^*}\cdot 4^j \qquad (1)$$

参数 P 表示连续被打孔的最低位部分积的数量。对于每个剩余的部分乘积 $\tilde{P}_j$，输入操作数 A 被舍入到其 r 位最小有效位 (r 每种产品不同) 如下：

$$A_j^r=\langle a_{N-1}a_{N-2}\cdots a_r\rangle_{2\text{'s}}+a_{r-1} \qquad (2)$$

比特级操作在 A_j^r 中简化了舍入过程，因为上述加法被吸收到基数 4 校正项的计算中，并且避免了进位传播。最后，$b_j^{\mathrm{R4}^*}$ 是编码信号 B 的常规四进制逻辑函数，带有额外的一个异或门涉及 a_{r-1}，这是由优化以避免加法所强加的。

ROUP 乘法器提供了一个大的设计空间来探索各种误差-能量折中，因为近似值由两个独立的参数调整。在 [2] 的比较评估中，ROUP 系列形成了具有更高分辨率的误差-能量帕累托前沿，超过了几个最先进的乘法器。与用于 ALWANN 的 EvoApprox8b 库中的乘法器 [5] 相比，ROUP 提供了更多的近似配置，特别是在更大的位宽情况下。因此，它的误差缩放更加密集，而在能量方面，有几个 ROUP 配置比 EvoApprox8b 乘法器高出 15%。

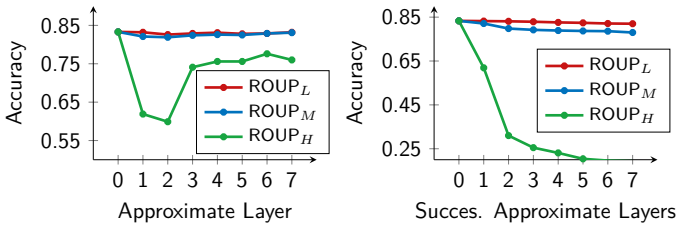
F. 近似 *DNN* 的能量模型

近似 DNN 加速器的能耗是基于近似乘法器的实现进行估算的。针对标准单元 ASIC 技术，这些乘法器使用工业级工具实现了 TSMC 45 纳米库，即，Synopsys Design Compiler 用于综合，Synopsys PrimeTime 用于测量功耗。

与 ALWANN [7] 类似，我们根据每层的乘法次数和每个乘法器的能量来计算每层的能量，即，$\#mult\times avg\_mult\_energy$。为了估计整个 DNN 的能量，我们将各个层的能量累加起来。

IV. 实验评估

为了评估我们的方法，我们采用了 ResNet-8 深度神经网络和基于 TensorFlow 1.14 的开源 ALWANN 框架 [7]。ResNet-8 在 CIFAR-10 数据集上进行量化训练，实现了 83%的分类准确率。



(a) (b)

图 3. ResNet-8 准确率随层的近似缩放：(a) 每次一层（例如，仅第 5 层），(b) 连续几层（例如，第 1 到 5 层）。“Layer=0” 表示基线量化模型。

分层错误恢复能力研究：首先，我们检查卷积层的误差敏感性，以了解哪些层可以进行近似。在这个实验中，我们选择了三个不同强度的 ROUP 乘数，即“低”, “中等”, 和 “高”, 分别标记为 ROUP_L, ROUP_M, 和 ROUP_H。图 3a说明了使用这些乘数时的精度变化仅在第 m 卷积层中使用 (m=1,2, ...7)。无论近似的强度如何，都表明近似第一层之一的结果是显著的精度损失 (m=0 显示基线模型，准确率为 83%)。这一点在 ROUP_H 配置中更为明显，其中插入了显著的计算误差。在这种情况下，当近似于 4-7 层之一时，精度损失减少并稳定在约 8%。图 3b, 描绘了在逼近前 m 层时的精度缩放。如预期那样，随着近似层数量的增加，精度损失也在增大。然而，我们再次注意到最后一层的误差鲁棒性。具体来说，当扩展第 4 层之后的近似时，精度损失减缓。此次探索的另一个重要结果是，无论逼近哪一层或多大数量的层，ROUP_L 和 ROUP_M 配置的精度损失可以忽略不计，因为这些乘法器与准确设计相比提供了显著的能量增益，即分别大约为 10%和 20%。

近似空间的探索：随后，我们在所提出的方案上进行了广泛的架构空间探索，涉及各种 ROUP 乘法器，以提取它们在准确性和能耗方面最为突出的配置。对于每个方案，检查了几种乘法替换和组合。在图 4中，我们展示了所有提供至少 75%分类准确率的配置，即与量化模型相比最多损失 8%的准确度。如图表左上角所示，多个配置相较于量化模型带来了可以忽略不计的准确度损失，范围从 0.02%到 1%之间，同时由于使用了 ROUP 乘法器而提高了能效。因此，我们的方案可以在接近零准确度损失的情况下实现更高效的计算。关于每个方案的效率，帕累托前沿几乎完全由内核和滤波级别配置构成。对于相同的准确度损失，这些配置提供了比传统的层级别方法更好的能耗性能，后者为了达到这种能效不得不牺牲大量的准确度，即超过 40%。

与 ALWANN 框架的比较：表 I比较了使用 EvoApprox8b 乘法器的 ALWANN 网络的 Pareto 前沿配置 [5]。所提出的的设计提供了更高的精度，因为 EvoApprox8b 配置的平均损失为~23%, 而在功耗方面，它们提供了 2× 的增益。这一比较突显了研究在更低级别的 DNN（例如，滤波器或核）中进行近似的优点。即，使用不同强度的乘法器的细粒度方法优于传统的所有层乘法操作的近似。

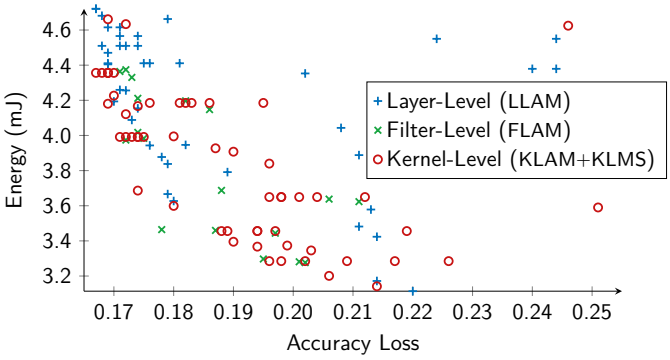


图 4. 提出的实现方法的近似 ResNet-8 硬件加速器的能量消耗和精度损失。

经验教训：根据我们的多级设计空间探索和评估，我们实验证明了：

(i) 第一层卷积层在近似上更为敏感，即不如最后的层那样能够容忍误差。

(ii) 基于非均匀细粒度滤波器/核级别的逼近方法的乘数近似比粗粒度层级逼近提供更高的精度。

V. 结论与未来工作

在本工作中，我们提出了 MAx-DNN 框架，以探索多级算术近似在 DNN 硬件加速器中的效率。MAx-DNN 扩展了先前的设计方法，考虑了近似乘法器的异构性，不仅针对每个 DNN 层，还针对每个滤波器和内核。结果显示我们的最突出配置实现了高达 54%的能量减少和 4%的精度损失，同时相较于使用 EvoApprox8b 乘法器的直接近似提供了 2× 增益。我们未来的工作包括对不同类型的 DNN 进行评估，以及研究 DNN 节点之间的误差传播。

参考文献

[1] S. Mittal, “A Survey of Techniques for Approximate Computing,” *ACM Computing Surveys*, vol. 48, no. 4, Mar. 2016.

[2] V. Leon *et al.*, “Cooperative Arithmetic-Aware Approximation Techniques for Energy-Efficient Multipliers,” in *Design Automation Conference (DAC)*, 2019, pp. 1–6.

[3] D. Hernandez-Araya *et al.*, “AUGER: A Tool for Generating Approximate Arithmetic Circuits,” in *IEEE Latin American Symposium on Circuits Systems (LASCAS)*, 2020, pp. 1–4.

[4] V. Leon *et al.*, “Approximate Hybrid High Radix Encoding for Energy-Efficient Inexact Multipliers,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 421–430, Mar. 2018.

[5] V. Mrazek *et al.*, “EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods,” in *Design, Automation and Test in Europe Conference (DATE)*, 2017, pp. 258–261.

表 I
近似 ResNet-8 硬件加速器的比较

近似	方法/配置	能量增益	准确率损失 [1]
Proposed – ROUP [2]	FLAM-3clas._2_1_1	49%	18%
	FLAM-3clas._2_2_1	52%	20%
	KLAM-chan._1_0_1	46%	17%
	KLAM-chan._2_0_2	53%	21%
	KLAM-chan._1_1_2	50%	19%
	KLAM-chan._2_1_2	54%	21%
	KLAM-row_2_1_1	50%	19%
	KLAM-row_2_1_2	52%	20%
Uniform – Evo [5]	Evo_mul8_2AC	23%	20%
	Evo_mul8u_2HH	23%	23%
	Evo_mul8u_NGR	32%	23%
	Evo_mul8u_ZFB	39%	23%
	Evo_mul8u_7C1	20%	24%

[1] 报告称与全精度模型相比。基准量化模型，我们在其中应用了我们的近似方法，已经损失了 17%的准确性。

[6] V. Leon *et al.*, "Exploiting the Potential of Approximate Arithmetic in DSP & AI Hardware Accelerators," in *Int' l. Conference on Field Programmable Logic and Applications (FPL)*, 2021, pp. 1–2.

[7] V. Mrazek *et al.*, "ALWANN: Automatic Layer-Wise Approximation of Deep Neural Network Accelerators without Retraining," in *Int' l. Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.

[8] G. Lentaris *et al.*, "Combining Arithmetic Approximation Techniques for Improved CNN Circuit Design," in *Int' l. Conference on Electronics, Circuits and Systems (ICECS)*, 2020, pp. 1–4.

[9] S. Venkataramani *et al.*, "Efficient AI System Design With Cross-Layer Approximate Computing," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2232–2250, Dec. 2020.

[10] V. Leon *et al.*, "ApproxQAM: High-Order QAM Demodulation Circuits with Approximate Arithmetic," in *Int' l. Conference on Modern Circuits and Systems Technologies (MOCAST)*, 2021, pp. 1–5.

[11] U. Köster *et al.*, "Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks," in *Int' l. Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 1740–1750.

[12] J. Wu *et al.*, "Quantized Convolutional Neural Networks for Mobile Devices," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4820–4828.

[13] S. Anwar *et al.*, "Structured Pruning of Deep Convolutional Neural Networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, Feb. 2017.