arXiv:2301.07484v1 [cs.AR] 12 Jan 2023

揭示永久性故障下近似深度神经网络的可靠性退化与缓解方法

Ayesha Siddique 和 Khaza Anuarul Hoque*IEEE* 高级会员

摘要——**近似计算通过引入可容忍的准确率损失来提升深度神经网络加速器的能效。然而，细微的准确率变化可能会提高这些加速器对不期望的细微扰动（如永久性故障）的敏感性。文献已对精确深度神经网络加速器（AccDNN）中的永久性故障影响进行了全面研究。相较之下，近似深度神经网络（AxDNN）加速器中的永久性故障影响及其缓解方法尚未得到充分探索。为此，我们首先使用 GPU 和 TPU 加速器中的先进 Evoapprox8b 乘法器，对近似多层感知机（MLP）和卷积神经网络（CNN）进行全面的故障韧性分析。随后，我们提出一种新颖的故障缓解方法，即故障感知权重重新调优（ Fal-reTune）。Fal-reTune 在存在故障的情况下，利用权重映射函数重新调优权重，以提高分类准确率。为了评估故障韧性以及所提出缓解方法的有效性，我们采用了应用最广泛的 MNIST、Fashion-MNIST 和 CIFAR10 数据集。结果表明，与 AccDNN 加速器相比，永久性故障会加剧 AxDNN 中的准确率损失。例如，AxDNN 中的永久性故障可导致 56% 的准确率损失，而同一故障位在 AccDNN 加速器中仅会导致 4% 的准确率损失。我们通过实验表明，即使故障率高达 50%，所提出的 Fal-reTune 缓解方法仍可使 AxDNN 的性能提升至多 98%。此外，我们观察到，AxDNN 中的故障韧性与其能效相互独立。**

索引词—**深度神经网络、近似计算、永久性故障、故障容错、故障缓解。**

I. 引言

深度学习算法能够处理并分类不同应用中的大数据，例如智慧医疗、汽车、智能工厂等，因此有望为工业 4.0系统赋予高度自主性。这些算法主要基于深度神经网络，而深度神经网络近来一直在满足不同的性能要求。例如，针对多个硬件平台上的计算和内存挑战，研究人员开发了多种卷积神经网络模型，如 AlexNet [1], VGG [2],等 [3]。最广泛使用的硬件平台包括图形处理器（GPU）和张量处理器（TPU） [4]，它们都具备大规模数据并行能力。尽管这些硬件平台对现代深度神经网络的鲁棒性极具吸引力，但这些硬件平台庞大的计算能力是以高能耗为代价的，而冷却负荷又进一步加剧了这一问题 [5], [6]。因此，将它们用作智能纳米尺度应用中的深度神经网络加速器变得极具挑战性 [7] 。例如，在资源受限的边缘设备上运行复杂的深度神经网络分析时，安全性和能量是关键考量；意外的能源中断可能危及人类生命。利用近似计算可以解决这一问题 [8], [9] ——这是一种非精确计算方法，通过利用机载应用固有的容错能力，提高深度神经网络加速器的能效。然而，近似硬件加速本质上被认为可靠性较低[10]。

基于近似计算的深度神经网络加速器（AxDNN）通过集成非精确算术单元、 [11] [12],计算跳过[13]，内存跳过 [14]，等方式设计而成。其在纳米尺度下的制造遵循复杂的制造工艺，而工艺中的不完善之处可能导致制造缺陷，例如工艺偏差和永久性故障（固定型故障） [9]。正如本文所述，永久性故障会在每个执行周期影响 AxDNN 加速器的计算单元，而这些故障若未被屏蔽，就会导致整个系统严重失效。事实上，与精确设计方案，即精确深度神经网络加速器（AccDNN） [15]相比，由于自诱错近似计算，永久性故障对 AxDNN 加速器的影响更为严重。然而，AxDNN 的故障韧性与 AccDNN 的故障韧性之比，取决于数据精度、永久性故障的位置和类型、加速器规模、近似程度、激活函数、神经网络拓扑结构及加速器等因素。需要考虑的变量如此之多，因此分析 AxDNN 应对永久性故障的可靠性极具挑战性。

永久性故障通常通过制造后测试来检测，以剔除有故障的制造芯片。然而，如果制造出的芯片中有大量存在故障，剔除它们会大幅降低良率。一种可行的解决方案是采用冗余执行（重新执行）来确保输出正确 [16],，但重新训练近似深度神经网络会增加总体运行时间 [17]。例如，训练一个包含 50 层、具有任意近似组件的 AxDNN 需要 89 天 [18]。这是因为近似算术组件的仿真通常难以良好扩展，因而会使训练速度在中央处理器上降低一个数量级，在图形处理器上降低三个数量级 [19]。因此，在当前资源受限的

Ayesha Siddique 和 Khaza Anuraul Hoque 任职于美国密苏里州哥伦比亚市密苏里大学电气工程与计算机科学系。电子邮件： {ayesha.siddique@mail.missouri.edu, hoquek@missouri.edu}稿件收到日期：xxxx，xx

纳米级硬件范式中，近似算术单元因其能效而受到广泛探索，因此不仅必须全面分析 AxDNN 的可靠性，还必须采用高效且容错的策略，最大限度地提高其良率。

研究并缓解 AxDNNs 中的可靠性问题是一项有趣的研究课题。然而，近期工作的重点主要局限于 AccDNN。最近，Kundu 等人阐明了小规模、类似 Google TPU 的脉动阵列式深度神经网络加速器中永久性故障的屏蔽与非屏蔽影响，并给出了形式化保证 [20]。在另一项工作中，Santos 等人向图形处理器的寄存器文件中注入永久性故障，以展示这些故障对低精度 AccDNN 的影响 [21]。Guerrero-Balaguera 等人分析了运行 AccDNN 的图形处理器中，寄存器文件和功能单元受到故障影响的情况 [22]。Condia 等人研究了在图形处理器上运行 AccDNN 的卷积计算中，关键模块和用户不可见模块（如 Warp 调度器和流水线寄存器）发生故障所产生的影响 [23]。最近，我们在 [15] 中的先前工作探讨了 AxDNN 的故障韧性及其能耗权衡，研究对象是在一个 256x256 的近似脉动阵列式深度神经网络加速器上运行的 AxDNN，该加速器类似于 Google TPU。然而， [15]中的分析存在若干局限：（i）分析仅限于简单的近似多层感知机；（ii）仅分析了基于 TPU 的 AxDNN 中逐层永久性故障的影响，缺少非逐层分析；（iii）分析仅针对基于脉动阵列的架构，忽略了永久性故障对基于 GPU 的加速器的影响；（iv）分析未对永久性故障的缓解提供任何见解。值得注意的是，重新训练 AxDNN 是一个复杂的问题；因此，当前最先进的、针对 AccDNN 的基于重新训练的故障缓解方法 [24]–[26] 无法直接应用于 AxDNN。总之，尽管已有多项工作分析并缓解了 AccDNN 在基于 TPU 和 GPU 的加速器上的可靠性问题，但对于映射到基于 TPU 和 GPU 的加速器上的 AxDNN，永久性故障影响的分析与缓解仍存在显著的研究空白。

受此启发，本文在 [15] 中大幅扩展了我们先前的工作，并针对 AxDNN 提出了以下三项创新贡献：(i) 我们使用多种近似卷积神经网络（CNN）和多层感知机（MLP），对张量处理器和 GPU 加速器进行了全面的故障韧性分析。具体而言，我们分析了固定型故障对近似 LeNet-5、VGG-11、AlexNet 和两个多层感知机的影响，所用数据集包括 MNIST、Fashion-MNIST 和 CIFAR-10。我们的故障韧性分析（逐层及非逐层）涵盖了不同的故障类型、故障位位置、故障层、激活函数、层宽度以及近似误差程度；(ii) 我们探讨了这些近似网络中故障韧性与能耗之间的权衡；(iii) 我们还提出了一种新颖的故障缓解方法，即对权重进行**故**障**感**知**重调**整（Fal-reTune）。Fal-reTune 首先将 AxDNN 故障感知映射到底层硬件上，然后重新调整使用近似算术组件的权重映射函数对权重进行调整。我们的结果表明，故障会加剧近似计算带来的准确率损失；AxDNN 的故障韧性会根据激活函数类型和近似误差大小，从输出层到输入层呈现变化。近似误差越大，故障破坏输出质量的倾向就越强。我们通过实验表明，近似多层感知机中的永久故障可导致高达 56% 的准确率损失。相比之下，同一故障发生在精确对应模型的相同位置时，只会导致 4% 的准确率损失。有趣的是，即使故障率高达 50%，我们提出的 Fal-reTune 缓解方法仍能有效提升近似 MLP 和卷积神经网络的分类准确率，最高可达 98%。此外，我们观察到，近似网络中的故障韧性与能耗彼此正交。

本文其余部分安排如下：第二节讨论分析准确深度神经网络和 AxDNN 可靠性的最新研究。第三节介绍深度神经网络、深度神经网络硬件加速器和近似乘法器的预备知识。第四节和第五节分别介绍我们的评估方法以及提出的 Fal-reTune 缓解方法。第六节介绍我们针对 AxDNN 开展的广泛故障韧性分析以及 Fal-reTune 故障缓解所获得的结果。最后，第七节总结全文。

II. 相关工作

安全关键应用对高可靠性神经网络日益增长的需求，促使研究人员深入分析故障对深度神经网络的影响。故障可能是永久性的，也可能是瞬态的 [27]。永久性故障会影响深度神经网络加速器的每个执行周期，因此比偶发的瞬态故障更显著地降低其性能 [20]。为此，当前研究提出了不同的分析方法和框架，以全面分析永久性故障对准确深度神经网络的影响 [28]。此前，Reagen 等人提出了一种轻量级框架，用于实证分析因工艺偏差或闪存寿命磨损问题而产生的永久性故障对准确深度神经网络的故障韧性 [29]。可用的故障位置仅限于存储器域，包括权重、活动值和隐藏状态。在另一项研究中，Rusopo 等人基于 darknet 开源框架，提出了一个支持不同精度的永久故障注入环境 [30]。研究采用了多种位宽方案和数据类型（即浮点和定点），对精确 CNN 模型进行了全面的故障韧性分析 [30]。Hong 等人通过在晶体管级注入永久性故障，评估了精确前馈神经网络的缺陷容忍能力 [31]。

现有最先进的研究也探讨了永久性故障对AccDNN硬件加速的影响。Kundu 和 Hanif 等人阐明了永久性故障对运行于 Google TPU [24] [26] 上的精确深度神经网络的影响。Zhang 等人通过对精确前馈神经网络中的固定型故障进行形式化建模，研究了这一影响

[20]。其他研究则关注运行于 GPU 加速器上的准确深度神经网络中的故障。Guerrero-Balaguera 等人提出了一个框架，借助二进制插桩工具，在精确 CNN 模型中开展故障注入活动，针对 GPU 内部的不同组件，例如寄存器文件和功能单元 [22]。Tsai 等人介绍了一款名为 NVBitFI 的工具，它无需了解 AccDNN 所用 GPU 加速器的信息，即可便捷地在 AccDNN 中开展永久性和瞬态故障注入活动 [32]。Sio 等人分析了混合平台可编程硬件上精确 CNN 模型的可靠性[33]。为缓解永久性故障对硬件加速器的影响，Zhang 等人建议针对底层硬件（如 TPU）采用故障感知映射，并对 AccDNN 参数进行剪枝，随后重新训练模型，以提高分类准确率 [24]。Hanif 等人提出，在软件层的重新训练阶段，根据硬件中的故障位置重新排列 AccDNN 参数，以改进这一缓解方法 [26]。然而，使用近似乘法器的近似硬件本身就难以重新训练 [34]。此外，另一项研究利用权重范围绕过 AccDNN 中的故障乘法器和 MAC 单元 [25]；不过，这种故障缓解方法是否适用于 AxDNN，也取决于近似训练方法。总之，现有最先进的研究大多聚焦于分析和缓解 AccDNN 中的可靠性威胁。然而，在脉动阵列和基于 GPU 的硬件加速器上，针对近似 CNN 模型的故障韧性开展研究并加以缓解，仍存在显著的研究空白。受此启发，我们深入研究并缓解了 AxDNN 中固定为 0 和 1 的故障所带来的影响。

III. 背景

本节简要介绍深度神经网络、其硬件加速器（如图形处理器和张量处理器），以及近似乘法器，以便更好地理解本文。

A. 深度神经网络

考虑一个简单的全连接神经网络，该网络由如图 1 所示的 L 个全连接层堆叠而成。每个层 $l \in [1, L]$ 包含 N 个神经元，其输出称为激活值 a。层 l 将

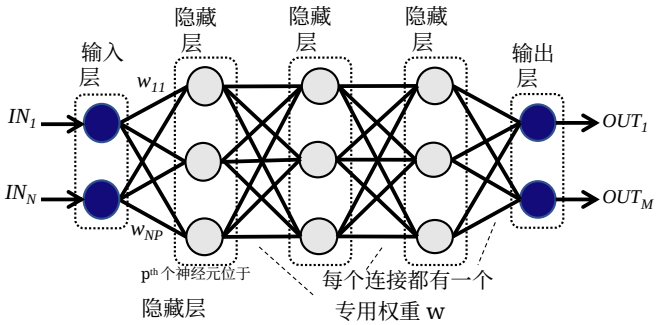


图 1：全连接深度神经网络

来自前一层的激活值 $l-1$ 与权重矩阵 w相乘，该矩阵的维度为 NxN，并加上维度为 $Nx1$的常数偏置b，然后经过激活函数ϕ。数学上可表示为：$y_i^l = \sum$，其中ϕ 可以是 ReLU、双曲正切、Sigmoid函数等。全连接层最常见的特殊情况是卷积神经网络模型中的卷积层。这些层将激活值处理为三维张量，即高度 × 、宽度 × 和通道。权重则处理为四维向量，即高度 x 宽度 x 通道 x 数量，其中数量指定应用于同一输入的滤波器数量。卷积层的输出具有与输入激活值相同的布局；但是，高度和宽度根据卷积核的形状确定。在图形处理器实现中，卷积层和全连接层中的矩阵乘法都采用典型的分块通用矩阵乘法（GEMM）进行处理，同时利用线程并行。这最终加快了仿真速度。

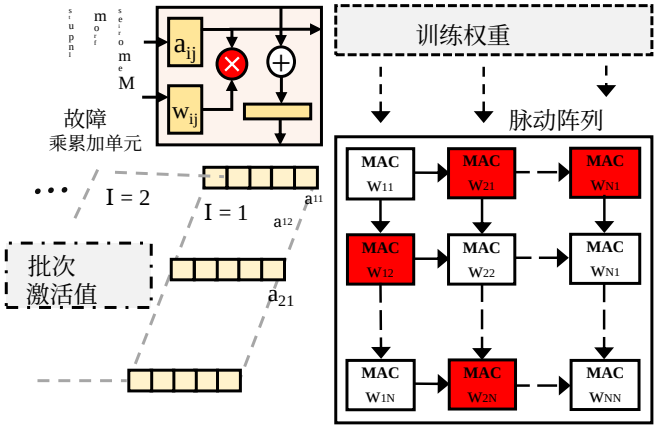


图 2：由 $N \times N$ 个乘加（MAC）单元组成的脉动阵列网格。脉动阵列中的红色表示存在乘法故障的故障 MAC 单元

B. 深度神经网络硬件加速器

用于深度神经网络硬件加速的最常见硬件加速器包括图形处理器和张量处理器 [4]。如图 2 所示，张量处理器采用二维脉动阵列，由一个$N \times N$ 乘加（MAC）单元网格组成。它在全连接层和卷积层中执行并行矩阵乘法。为了理解张量处理器的工作原理，我们以一个具有N 个输入神经元和 N 个输出神经元的全连接层为例，因此，需将一个 $N \times N$权重矩阵加载到脉动阵列中。权重矩阵中的$w_{i,j}$ 首先被加载到位于第 i^{th} 行、第j^{th} 列的 $MAC_{i,j}$ 中。随后，激活值被加载到脉动阵列中，并周期性地从左向右传递。在第一个时钟周期，脉动阵列第一列中的 $MAC^{1,1}$ 计算 $w_{1,1}a_1$，并将结果向下传递给 $MAC^{1,2}$。在下一个时钟周期，$MAC^{1,2}$ 将乘积 $w_{1,2}a_2$ 加到其输入上，并继续向下传递结果。经过 N 个时钟周期后，$MAC_{1,N}$ 最终输出$w_{1,i}a_i$，它是权重矩阵的第一个元素。请注意，第二列也会接收并处理与第一列相同的输入流

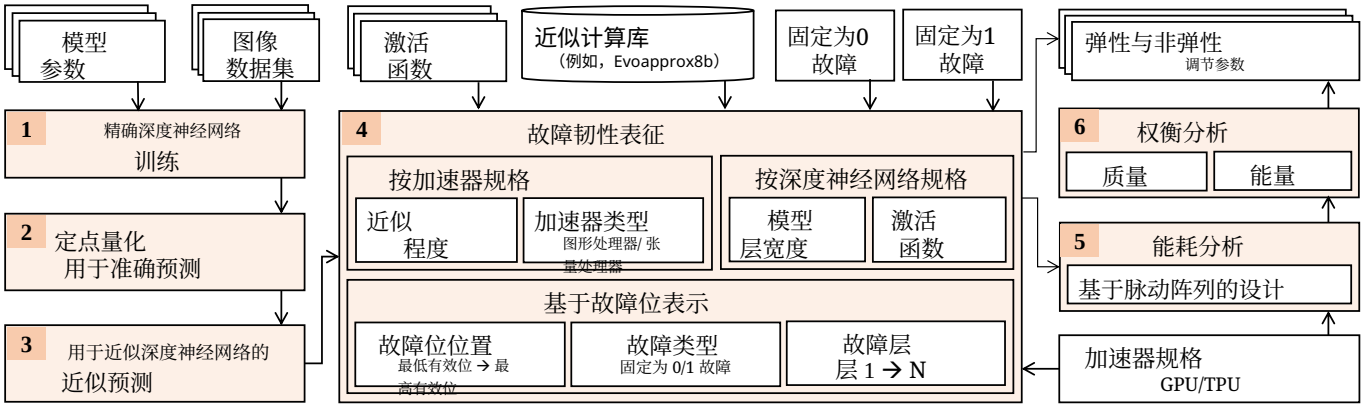


图 3：评估方　　法：分析基于脉动阵列的深度神经网络加速器中的故障与能耗权衡　　rs

，但延迟一个时钟周期。该过程沿脉动阵列的宽度重复进行。请注意，在张量处理器上，大于脉动阵列尺寸的输入矩阵会分多个批次通过脉动阵列进行处理。另一方面，图形处理器通过使用多个线程，简化了硬件中大规模矩阵乘法的实现，从而实现稳健的深度神经网络实现。它们采用典型的分块通用矩阵乘法（GEMM），线程从每个输入矩阵中加载一个二维瓦片（由滤波器和通道块组成）到共享内存中。每个线程计算一个输出值 [34]。共享内存中的瓦片会进行量化，以避免可能发生的共享内存访问冲突。近期最先进的研究着重于在张量处理器的脉动阵列中引入分块机制 [35], [36]。本文使用的张量处理器配备了一个 256 x 256 的脉动阵列，与 Google TPU 类似。

C. 近似乘数

近似乘法器常被推荐用于容错应用中的能效计算[37]。它们采用近似计算技术，例如近似计数器或压缩器 [11]，部分积的约简 [38]，以及进位传播链的截断 [39] 等，以降低延迟和功耗或能耗。近年来，已有许多开源近似乘法器库问世，例如 SMApproxLib [39]， DeMAS[40]，Evoapprox8b[41],lpAClib [42]，leAp [43]，SIMDive [44]，等。其中，Evoapprox8b 是最全面的库，包含 28 种精确乘法器的不同有符号和无符号近似版本。这些乘法器面向 ASIC，并采用多目标笛卡尔遗传编程构建 [39]。AxDNN 通过采用兼容 TensorFlow 的近似二维卷积，将此类近似乘法器作为查找表纳入模型中 [34]。本文分别使用 Evoapprox8b [39]库中的有符号精确乘法器和近似乘法器，实现 AccDNN 和 AxDNN。

IV. 可靠性评估方法

我们的可靠性评估方法发挥三项作用：首先，为硬件加速器实现 AxDNN；其次，利用故障位表示以及模型和加速器的规格来表征 AxDNN 的故障韧性；第三，探索 AxDNN 的故障-能耗权衡。图3概述了我们的评估方法。

我们评估方法的首要步骤是训练达到基准准确率的准确深度神经网络。为此，我们针对不同数据集，通过反复试验确定设计参数，例如隐藏层数量、神经元数量、学习率、动量等。随后，我们对 AxDNN 中能耗最高的全连接层和卷积层的输入进行量化，即分别对近似多层感知机和卷积神经网络模型进行量化。然而，如 TensorFlow 中所示，量化之后还需进行反量化。具体而言，我们采用支持 8 位有符号乘法的 8 位有符号整数定点量化。接下来，我们将准确深度神经网络中的精确乘法器替换为 Evoapprox8b [45]库中的不同近似乘法器，以设计多个 AxDNN。通常，8 位乘法器的查找表在图形处理器 [34] 上仅占用 128 KB。因此，我们利用查找表，无需针对每个输入计算近似乘法器的输出，从而以极低的内存开销加速图形处理器上的 AxDNN仿真。我们将各层分解为一组瓦片，以便利用 GPU加速器实现 AxDNN。有趣的是，这些瓦片可以映射到脉动阵列，即张量处理器的核心。因此，本文将脉动阵列实现为 AxDNN 的硬件加速器。接下来，我们将如下表征 AxDNN 的故障韧性。

(a) 加速器规格：我们在故障MAC单元数量固定的情况下，改变脉动阵列中的近似程度；另一种情况下，我们改变 TPU 脉动阵列中的故障MAC单元数量，并将相应的准确率损失与图形处理器瓦片内部发生故障时的准确率损失进行比较。故障仅注入一个瓦片中的一个执行周期。此分析有助于我们了解加速器的不同组件如何影响 AxDNN 的故障韧性。

（b）故障位表示：有关故障韧性的详细说明

分析中，我们研究故障对近似乘法器输出位的影响，以及这如何导致显著的准确率损失。具体而言，我们通过分析分类准确率，研究了 AxDNN 中所有近似乘法器在不同输出位位置（从最低有效位到最高有效位）发生固定为 0 和固定为 1 故障时的影响。

(c) 模型规格：我们进一步将分析扩展至 AxDNN 的逐层容错能力。由于每个瓦片或层中的故障都可能传播到所有下游层 [23]，因此本次详细分析采用固定型故障这一特殊情形。我们的分析还包括研究不同激活函数和层宽对 AxDNN 容错能力的影响。最后，我们通过分析每个 AxDNN 中近似脉动阵列的能耗和故障韧性，探讨故障能耗权衡。这种权衡分析得出了多个兼顾能效与故障韧性的调节参数，以及不具备故障韧性的调节参数。

评估方法的实用性很大程度上取决于定制 AxDNN 以及在硬件加速器中模拟故障注入是否便捷。本文通过确定 AxDNN 的计算在张量处理器和图形处理器上的映射，使得为故障注入定制 AxDNN 变得简单。请注意，我们的评估方法可用于任何类型的 AxDNN。

V. 用于故障缓解的故障感知权重重新调整方法（FAL-RETUNE）

本节提出一种新颖的 Fal-reTune 缓解方法，以增强 AxDNN 的故障韧性。Fal-reTune 首先对映射到故障 MAC 单元的权重进行剪枝。故障位置通过对 AxDNN 芯片进行制造后测试来确定。此初始步骤类似于在脉动阵列中通过硬件层面的多路复用器旁路准确深度神经网络中的 MAC 单元 [24]。启用旁路路径后，故障 MAC 单元对列和的贡献将被跳过。然而，由于数据处理过程中会复用脉动阵列，旁路单个故障 MAC 单元可能导致多个预训练权重被剪枝。对于准确深度神经网络模型，当前最先进的故障缓解技术通常通过重新训练来解决这一问题[24]–[26] 。然而，这些技术无法直接用于 AxDNN 的故障缓解，因为重新训练 AxDNN 会增加整体训练时间。例如，训练一个包含 50 层且具有任意近似组件的 AxDNN 需要 89 天[18]。这是因为近似算术组件的模拟通常难以扩展，因此训练速度会在中央处理器上降低一个数量级，在图形处理器上降低三个数量级， [19]。

请回顾一下，我们的 AxDNN 是在推理阶段用近似乘数替换精确乘数构建而成的。然而，其训练阶段使用的是精确乘数。这为我们提供了缓解 AxDNN 故障的思路，例如，重新训练 AxDNN 的精确对应模型，然后在近似推理中重新调整未剪枝权重（受 [19] 中权重重新调整方法的启发）。基于

算法1： Fal-reTune缓解算法

输入： (i) 预训练权重：wts；(ii) 训练数据：trData；(iii) 测试数据：tsData；(iv) 故障映射：fmaps；(v) 最大再训练轮数：trEpochs；(vii) 近似乘法器：mx；(viii) 准确率阈值：accThresh；
输出： 准确率：acc;
1: ind =FindPrunedWeightsIndices（特征图，权重）
//从故障映射中查找剪枝权重的索引
2: pWts =SetPrunedWeightsToZero(ind, wts)
//将上述索引处的剪枝权重设为零
3: **对于** epochs = 0 : trEpochs - 1**执行**
4: nWts =更新权重 (pWts, trData)
//通过反向传播更新权重
5: nWts = 将更新后的权重置零(nWts, ind)
//使用步骤 1 中的索引将所有剪枝权重置零
6: **结束循环**
7: nWts' =近似权重映射(nWts, mx)
//使用权重映射方案更新权重
8: acc =检查推理准确率(nWts', tsData)
//使用新权重检查推理准确率
9:**返回** acc;

这一思路，我们提出的 Fal-reTune 缓解方法使用权重映射函数 f_M 重新调整未剪枝权重；该函数针对每个乘数 M 预先离线计算。对于每个权重 w， f_M 确定一个新权重 w'，使近似乘法与精确乘法在所有激活值 $a \subset A$ 上的输出之间的绝对差之和最小（A 指激活矩阵），如下所示：

$$f_M(w) = \arg \max_{w' \in W'} \sum_{a \in A} |M(a, w') - a \cdot w|, \qquad (1)$$

其中， W'表示近似权重矩阵。对于精确乘法器，$w' = w$。算法1概述了所提出的Fal-reTune缓解方法所涉及的步骤。第1至2行对映射到脉动阵列中故障处理单元的AccDNN预训练权重进行剪枝。从第3行开始，对AxDNN的精确对应模型进行再训练。第4行通过再训练中的反向传播更新未剪枝权重。第5行在每个再训练轮次结束时，将映射到故障处理单元的权重设为零。第7行在近似推理中对未剪枝权重进行重新调整。最后，Fal-reTune使用更新后的权重检查近似推理的分类准确率。Fal-reTune算法将再训练准确率返回给用户。

VI. 结果与讨论

本节讨论使用 Evoapprox8b [41]库对 AccDNN 和 AxDNN 进行广泛故障韧性分析与故障缓解的实验设置和结果。从本节开始及后续内容中，精确乘法器记为 M1；近似乘法器则指从 Evoapprox8b 库中获得的 KVB、KX2、L2D、KTY 和 L1G 有符号近似乘法器（记为 M2-M11）。请注意，使用这些乘法器的误差度量——平均绝对误差（MAE）——来量化近似误差。

表 I：基准测试中精确多层感知机（MLP）和卷积神经网络（CNN）的分类准确率

数据集	架构	缩写	Acc.
MNIST	MLP-tanh	MP-tanh	96%
	MLP-softmax	MP-softmax	96%
	CNN-Lenet5	ML-base	97%
	宽版 CNN-Lenet5	ML-wide	97%
Fashion-MNIST	MLP-tanh	FP-tanh	82%
	MLP-Softmax	FP-Softmax	81%
CIFAR10	CNN-Alexnet	CC-Alx	82%
	CNN-VGG-11	CC-VGG	82%

A. 数据集

我们使用三个图像分类数据集，即 MNIST[46]，Fashion-MNIST [47] 和 CIFAR10 [48]。深度神经网络研究社区广泛使用这些数据集来评估深度神经网络 [49]–[51] 在嵌入式平台上的性能。MNIST 是一个用于识别手写数字（从零到九）的灰度图像数据集。另一方面，Fashion-MNIST 包含带有 10 个不同类别标签的灰度服饰图像，例如靴子、衬衫等。这些数据集包含 60,000 张训练图像和 10,000 张验证图像，图像尺寸为 28x28 像素。CIFAR10 是一个用于目标识别的彩色图像数据集。它包含 50,000 张训练图像和 10,000 张验证彩色图像，图像尺寸为 32x32 像素，图像带有不同类别的标签。

B. 模型配置

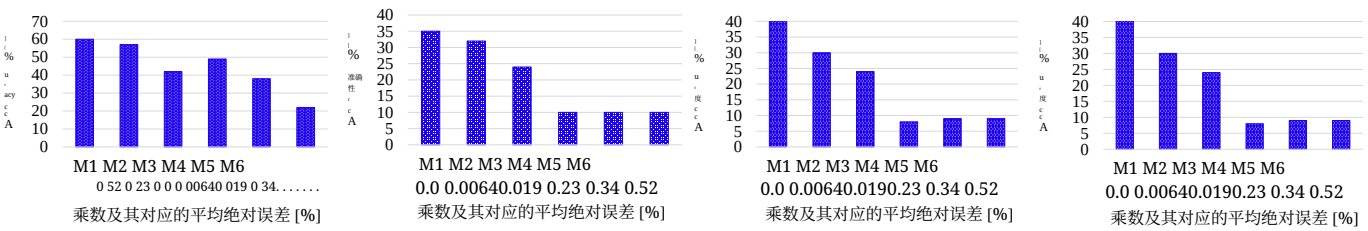
我们针对三种不同的基础深度神经网络模型开展故障韧性分析。对于 MNIST，我们定义了一个基线多层感知机架构，其层配置为 784-256-256-256-10，并生成两个变体，即 MP-tanh 和 MP-softmax。MP-tanh 和 MP-softmax 的区别仅在于所采用的激活函数分别为双曲正切和 Softmax。我们还使用 LeNet-5 作为基础模型 ML-base，并通过将 Conv1、Conv2、FC1 和 FC2 层的层大小分别扩展为 5x5x20、5x5x40、100 和 10，生成其变体 ML-wide。这里，Conv1 和 Conv2 表示 LeNet-5 中前两个卷积层，FC1 和 FC2 表示两个全连接层。同样，对于 Fashion-MNIST，我们生成一个层配置为 784-512-512-512-10 的基线多层感知机架构 base (FP)，并生成两个采用不同激活函数的多层感知机变体：FP-tanh 和 FP-softmax。我们还考察了一个现成的卷积神经网络模型 AlexNet，记为 FC-Alex。最后，对于 CIFAR10，我们分别采用 AlexNet 和 VGG-11，即 CC-Alex 和 CC-VGG。精确 MP-tanh、MP-softmax、FP-tanh、FP-softmax 的分类准确率，

表 I 给出了 CC-Alex、CC-VGG、ML-base 和 ML-wide。其近似版本通过在推理阶段将精确乘法器简单替换为近似乘法器生成。在以下各节中，我们将讨论本文中这些模型的应用。

C. 故障注入与能量估算

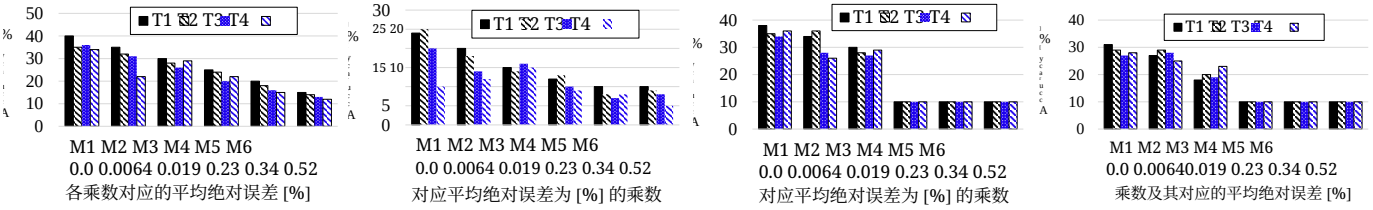
我们的故障韧性分析框架会系统地在 AccDNN 和 AxDNN 模型中注入固定为 0 和固定为 1 的故障，并使用准确率损失指标量化其影响；该指标表示模型注入故障后的分类准确率相较于无故障模型的下降幅度。该框架仅关注数据通路中的固定型故障，而忽略存储器组件中的故障，因为这些故障可通过使用纠错码轻松缓解。此外，该框架还忽略控制逻辑中的故障，因为控制逻辑仅占设计中微不足道的一小部分。在准确深度神经网络和 AxDNN 中，故障可能位于脉动阵列的计算单元内部或外部，即位于乘法器内部或乘法器输出端。然而，乘法器内部的故障可能会被乘法器逻辑的其他部分屏蔽，也可能不会。因此，我们模拟故障未被屏蔽并传播至乘法器输出端的情况。具体而言，我们在乘法器输出端注入故障。请注意，近似乘法器的输出（近似输出）不同于精确乘法器的输出。因此，近似乘法器输出端的固定型故障与精确乘法器输出端的同一故障会产生不同影响。本文使用 Python3.73 和 Tensorflow 2.2 实现故障注入分析；Tensorflow 2.2 支持 CUDA 10.1，可利用图形处理器加速计算。我们的实验在 NVIDIA GeForce RTX 2080 Ti 图形处理器上运行，处理器为 Intel Core i9-10900kF，运行频率为 3.06 GHz，配备 32 GB 随机存取存储器。

为了分析近似乘法器的能耗，我们首先为每个近似乘法器开发行为级设计，并在 Xilinx 中对其进行仿真，以生成值变更转储 (VCD) 文件。我们将该文件和行为级设计作为 Synopsys Design Compiler 的输入，使用 OSU FreePDK 45nm



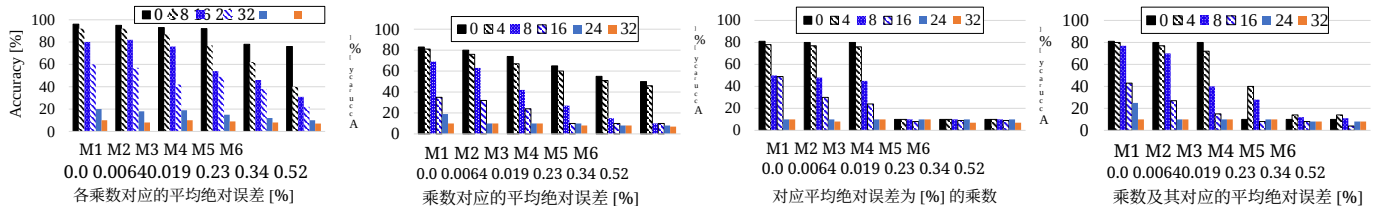
(a) MP-tanh 分类 (b) CC-Alx 分类 (c) FP-tanh 分类 (d) CC-VGG 分类

图 4：在张量处理器的 16% MAC 单元中注入固定型故障时，固定型故障对基于近似乘法器 M*n* 的 AxDNN 的影响。各近似乘法器在 AxDNN 中的平均绝对误差标注在底部。



(a) MP-tanh 分类 (b) CC-Alx 分类 (c) FP-tanh 分类 (d) CC-VGG 分类

图 5：当近似乘法器 M*n* 型深度神经网络中的乘法器发生故障并被注入图形处理器的不同瓦片时，固定型故障对其的影响。每种乘法器在深度神经网络中的平均绝对误差标注在底部。



(a) MP-tanh 分类 (b) CC-Alx 分类 (c) FP-tanh 分类 (d) CC-VGG 分类

图 6：当固定型故障分别注入 TPU 的 0%、4%、8%、16%、24% 和 32% 的 MAC 单元时，对基于近似乘法器 M*n* 的 AxDNN 的影响。AxDNN 中每种乘法器的平均绝对误差标注在底部。

标准单元库进行逻辑综合后，估算延迟和功耗。随后使用延迟和功耗报告来确定设计的能量。

D. 使用加速器规格进行故障分析

本节研究近似方式和底层加速器类型如何影响 AxDNN 的故障韧性。

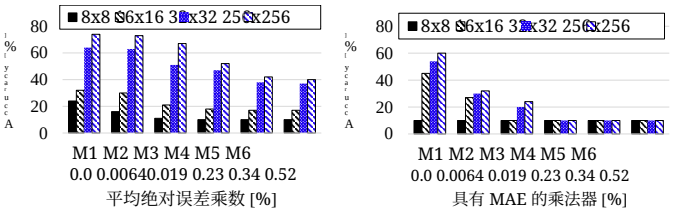
1）近似程度的影响：为了考察不同近似程度对底层硬件故障韧性的影响，我们在一个 256x256 的脉动阵列中，通过扰动近似乘法器的输出最高有效位，在 TPU 的 16% 乘加单元中注入固定型故障。具体而言，我们生成了一张故障图，然后将其映射到运行 AxDNN 的近似 TPU 脉动阵列中的乘加单元。故障图是用于记录脉动阵列中故障乘加单元的矩阵。故障图的大小与脉动阵列相同，其中的元素与脉动阵列的乘法累加器（MAC）单元一一映射。请注意，本实验选择最高有效位进行故障注入，以分析最坏情况，因为最高有效位发生故障会导致显著的准确率损失 [45]。图 4 中的结果表明，AxDNN 中的近似误差越大，故障韧性就会显著下降。例如，在 MP-tanh 中，向 M6乘法器注入故障后，分类准确率仅为 22%；而向 M1乘法器注入相同故障后，分类准确率为 76%（见图 4a）。这是因为 M6乘法器的平均绝对误差较高（即 0.52%），而 M1乘法器的平均绝对误差为零。M1乘法器是精确乘法器。近似误差越大，平均绝对误差也越大。我们还观察到，在 CC-Alex 中，向 M6乘法器注入故障后，分类准确率仅为 10%；与向 M1乘法器注入相同故障后达到的 35% 分类准确率相比，这一准确率也非常低（见图 4b）。此外，M4、M5 和 M6 近似乘法器在存在故障时会遭受显著的准确率损失；使用这些乘法器时，MP-tanh 的准确率损失相对较小。这是因为 MP-tanh 在无故障时的预测准确率相对较高，如表 I 所示。FP-tanh（见图 4c）和 CC-VGG（见图 4d）的分类也呈现出类似趋势。

2）加速器类型的影响：我们还分析了 TPU 和 GPU 核心中的故障如何影响 AxDNN 的故障韧性。为此，我们假设固定型故障位于 GPU 的核心瓦片中，并且每次仅有一个瓦片的 40% 受损。我们选择 40% 这一数值，是因为在实验中，当瓦片受损比例低于 40% 时，我们没有观察到故障造成显著影响。图 5 显示，此类故障注入会显著影响准确深度神经网络和 AxDNN。结果表明，在存在瓦片故障的 GPU 上运行时，MP-tanh（见图 5a）和 CC-Alx（见图 5b）的分类准确率分别处于 40% 至 12% 和 5% 至 25% 的范围内；不过，与平均绝对误差较低的近似乘法器相比，平均绝对误差较高的近似乘法器受故障影响更大。FP-tanh（见图 5c）和 CC-VGG（见图 5d）分类中也观察到类似趋势。这是由于与第 VI-D1 节中讨论的相同原因。

GPU 瓦片通常会映射到脉动阵列，以实现更可靠的并行矩阵乘法。然而，脉动阵列中的故障可能会传播到 AxDNN 的所有层。我们来考察在运行 AxDNN 的近似 TPU 中，向不同比例的 MAC 单元注入故障的情况。图 6 的结果表明，即使只有少量故障 MAC 单元，也会导致 AxDNN 的准确率显著下降。例如，8% 的故障 MAC 单元可使分类

准确率在使用 M4 近似乘法器（平均绝对误差为 0.23%）构建时，MP-tanh 和 CC-Alx 的准确率会分别从 90% 降至 52%、从 63% 降至 25%。然而，使用 M1乘法器构建 MP-tanh 和 CC-Alx 时，这些故障 MAC 单元仅会使准确率分别从 96% 降至 52%（见图 6a）和从 82% 降至 69%（见图 6b）。FP-tanh（见图 6c）和 CC-VGG（见图 6d）的分类也呈现出类似趋势。在 TPU 中，准确率下降尤为严重，因为 TPU 会被反复用于层执行，因此所有层都会受到处理单元中故障注入的影响。此外，故障 MAC 单元的比例越高，准确率下降就越显著；例如，当故障 MAC 单元占 32% 时，准确深度神经网络和 AxDNN 的分类准确率都只有 8%。

3) 加速器尺寸的影响：我们进一步扩展上述分析，研究不同尺寸近似脉动阵列中 AxDNN 的故障韧性。我们改变其中故障处理单元的数量，并运行 MP-tanh 和 CC-Alex。图 7 中的分析表明，脉动阵列尺寸越小，故障韧性越低，反之亦然。例如，在 MP-tanh 和 CC-Alx 分类中，8x8 脉动阵列的分类准确率均低于 256x256 脉动阵列。这是因为，尽管较小和较大的 AxDNN 都会重复使用脉动阵列来处理不同层的数据，但较小的脉动阵列需要更多执行周期（因为一次映射无法将该层完整映射到脉动阵列）。



(a) MP-tanh 分类 (b) CC-Alx 分类

图 7：在不同规模的脉动阵列中，对 16% 的乘加单元注入固定型故障时，固定型故障对 AxDNN 的影响（基于乘数 M1-M6）。每个乘数在 AxDNN 中的平均绝对误差列于底部。

E. 基于故障位表示的故障分析

本节研究近似位表示中的故障如何表征 AxDNN 的故障脆弱性。

1）故障位位置的影响：我们考察导致 AxDNN 准确率显著损失的故障位会使近似值产生多大变化。

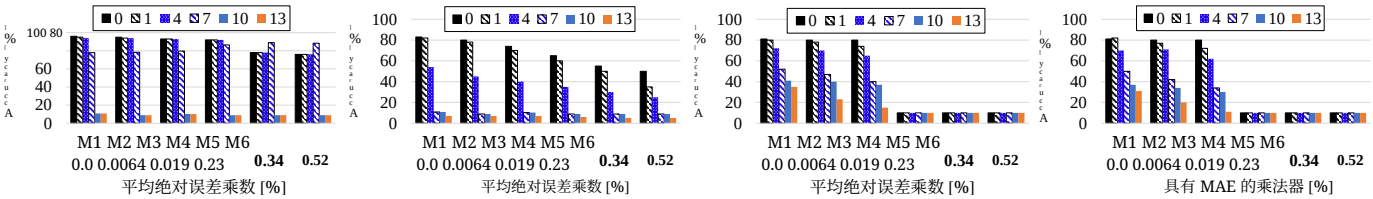
从现有研究可以明显看出，准确深度神经网络的故障韧性从最低有效位到最高有效位逐渐降低。我们研究了 AxDNN 中的类似趋势，以证明与准确深度神经网络相比，它们的故障韧性较低；而且，当我们转向低位（最低有效位）时，这种现象更加明显。如图 8 所示，在精确和近似 MP-tanh 的最高有效位发生固定为 1 故障时，观察到约 98% 至 100% 的准确率损失。另一方面，同样的故障发生在最低有效位时会导致近似 MP-tanh 的准确率损失约为 0% 至 9%，相比之下，精确 MP-tanh 的准确率损失为 1%（见图 8a）。事实上，与准确深度神经网络类似，AxDNN 的故障韧性也会从最低有效位到最高有效位逐渐降低。例如，在精确和近似 CC-Alx 的最高有效位发生固定为 1 故障时，观察到约 95% 至 100% 的准确率损失。另一方面，同样的故障发生在最低有效位时，会导致近似 CC-Alx 的准确率损失约为 0% 至 15%，相比之下，精确 CC-Alx 的准确率损失为 1%（见图 8b）。在 FP-tanh（见图 8c）和 CC-VGG（见图 8d）的分类中也观察到类似趋势。固定为 0 故障也呈现类似趋势（见图 9）。值得注意的是，与准确深度神经网络相比，AxDNN 对故障更敏感。这是因为 AxDNN 会继承非精确计算产生的近似误差，而这些计算中出现故障会加剧准确率损失（见第 VI-D1 节）。此外，最高有效位发生故障会导致乘数的输出值发生剧烈变化，而最低有效位发生故障只会使数值小幅增加或减少，即 [0 1]。这就是最高有效位故障导致的准确率损失显著高于最低有效位故障的原因。

2）故障类型的影响：我们还确定了哪种固定型故障会显著改变 AxDNN 中的近似值。比较图 8 和图 9 可知，固定为 1 故障造成的扰动大于固定为 0 故障。例如，在近似 MP-tanh 中，最高有效位发生固定为 1 故障会导致准确率损失高达 98%–100%；而最高有效位发生固定为 0 故障时，准确率损失略低，最高为 40%（见图 8a）。同样，在近似 CC-Alx 中，最高有效位发生固定为 0 故障会导致略低的准确率损失，最高为 50%（见图 9b）。在 FP-tanh（见图 9c）和 CC-VGG（见图 9d）的分类中也观察到了类似趋势。原因在于，固定为 1 故障会将位从 0 改为 1，从而显著增大乘数的输出值，导致前向传播期间出现极端的神经元激活。另一方面，固定为 0 故障会将位从 1 改为 0，从而减小乘数的输出值。一个简单的例子可以说明这一有趣现象：若乘数的 4 位输出“0001”的第一位发生固定为 1 故障，数值将增加 8（十进制），变为“1001”。然而，若第四位发生固定为 0 故障，数值则会降为零。

F. 基于深度神经网络规格的故障分析

本节中，我们研究 AxDNN 模型的各种属性如何影响其对固定型故障的敏感性。

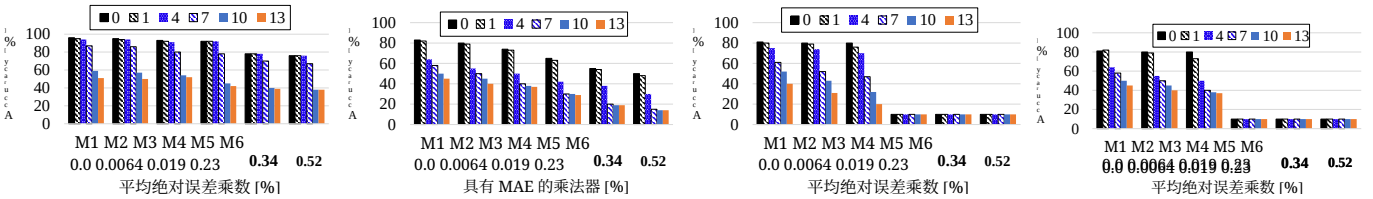
1) 故障层的影响：我们进行了逐层故障注入分析，以确定对 AxDNN 整体性能效率贡献最大的和最小的具有韧性的层。在最高有效位中存在固定型故障时，这种影响更加明显。因此，本文对其进行了分析。图 10—13 中的结果表明，与其他层相比，输入层第1层对故障的韧性相对较低，而输出层的故障韧性则相对较高。这是因为输入层中的故障可能会影响所有 DNN 层的输出，并显著降低故障韧性。例如，



（a）MP-tanh 分类 （b）CC-Alx 分类 （c）FP-tanh 分类 (d) CC-VGG 分类

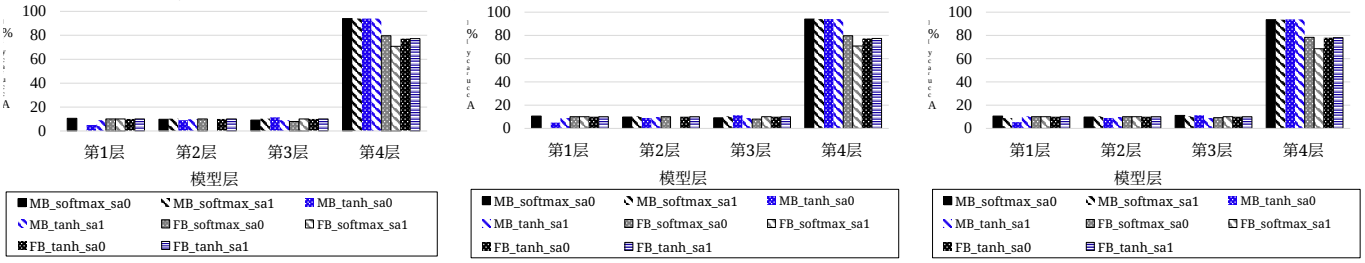
0

图 8：在第 0、1、4、7、10 和 13 位注入固定为 1 故障时，固定为 1 故障对基于近似乘数 M*n* 的 AxDNN 的影响。AxDNN 中各乘数的平均绝对误差标注在底部。



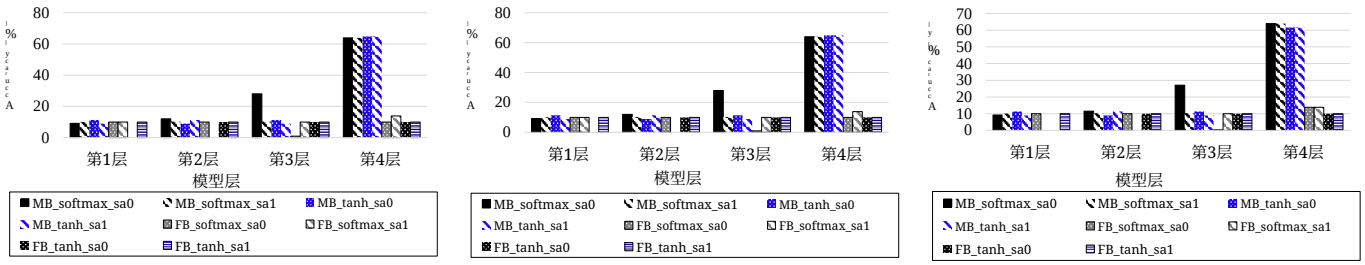
(a) MP-tanh 分类 (b) CC-Alx 分类 (c) FP-tanh 分类 (d) CC-VGG 分类

图9：当固定为 0 故障注入第 0、1、4、7、10 和 13 个位时，其对基于 M*n* 的 AxDNN 中近似乘法器的影响。每个 AxDNN 中乘法器的平均绝对误差标注在底部。



（a）基于M1的近似深度神经网络分类 （b）基于M2的近似深度神经网络分类 （c）基于M3的近似深度神经网络分类

图10：当固定为 1（sa1）和固定为 0（sa0）故障注入不同层时，其对采用近似乘法器 M1（平均绝对误差为 0.0%）、M2（平均绝对误差为 0.0064%）和基于 M3（平均绝对误差为 0.019%）的 MP-tanh 和 FP-tanh 分类的影响。每个 AxDNN 中乘法器的平均绝对误差标注在底部。

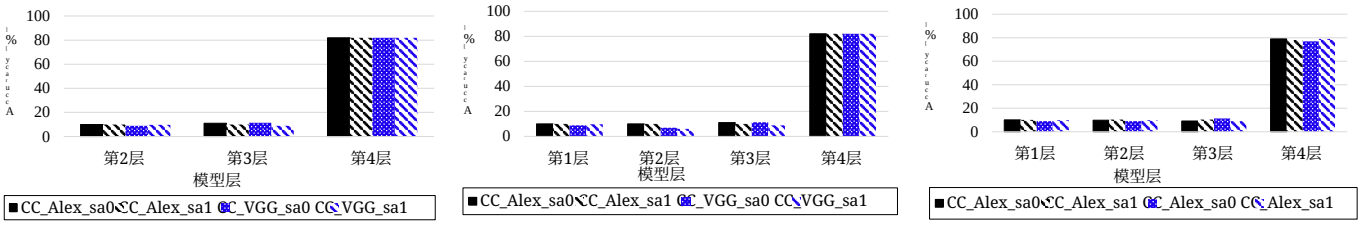


（a）基于M4的近似深度神经网络分类 (b) 基于M5的近似深度神经网络分类 (c) 基于M6的近似深度神经网络分类

图 11：在不同层中注入固定为 1（sa1）和固定为 0（sa0）故障时，对近似乘法器 M4（0.23 % 平均绝对误差）、M5（0.34 % 平均绝对误差）和基于 M6（0.52 % 平均绝对误差）的 MP-tanh 和 FP-tanh 分类的影响。每个乘法器在 AxDNN 中的平均绝对误差标注在底部。

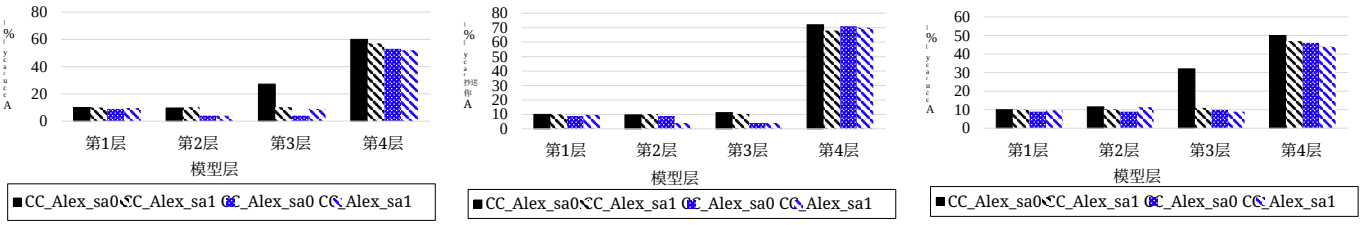
第1层 M6 近似乘法器中的固定为1故障，在精确和近似 MP-tanh 中分别导致约 82% 和 56% 的准确率损失；而在第4层中，仅分别导致约 30% 和 3% 的准确率损失。同样，第1层和第4层 M5 近似乘法器中的固定为1故障，在近似 FP-softmax 中分别导致约 80% 和 66% 的准确率损失。然而，在精确 MP-softmax 中，相同的故障配置仅导致 9% 准确率损失。CC-Alex 和 CC-VGG 分类也呈现出类似趋势。

2）激活函数的影响：图10和图11中 MP-tanh 与 FP-tanh 的对比表明，在 AxDNN 中，故障的影响会随激活函数而异。我们观察到，隐藏层采用双曲正切激活函数的架构似乎比采用 Softmax 激活函数的架构受故障影响更小



(a) 基于 M1 的 AxDNN 分类 (b) 基于M2的近似深度神经网络分类 (c) 基于M3的近似深度神经网络分类

图 12：在不同层中注入故障时，卡在 1（sa1）和卡在 0（sa0）故障对采用近似乘法器 M1（0.0% 平均绝对误差）、M2（0.0064% 平均绝对误差）和 M3（0.019% 平均绝对误差）的 CC-Alex 和 CC-VGG 分类的影响。AxDNN 中各乘法器的平均绝对误差标注在底部。



(a) 基于M4的近似深度神经网络分类 (b) 基于M5的近似深度神经网络分类 (c) 基于M6的近似深度神经网络分类

图 13：当固定为 1（sa1）和固定为 0（sa0）故障注入不同层时，对基于近似乘法器 M4、M5 和 M6 的 CC-Alex 与 CC-VGG 分类的影响。AxDNN 中各乘数的平均绝对误差标注在底部。

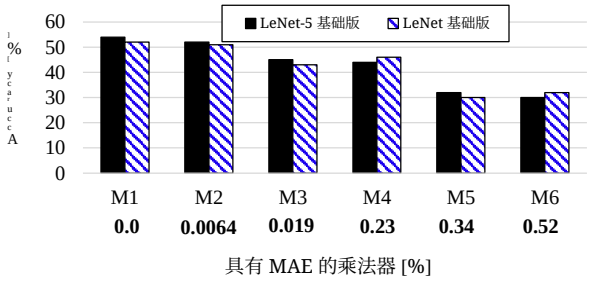


图 14：当固定为 1（sa1）和固定为 0（sa0）故障注入不同层宽的基于近似乘法器 M*n* 的近似深度神经网络时产生的影响。AxDNN 中各乘数的平均绝对误差标注在底部。

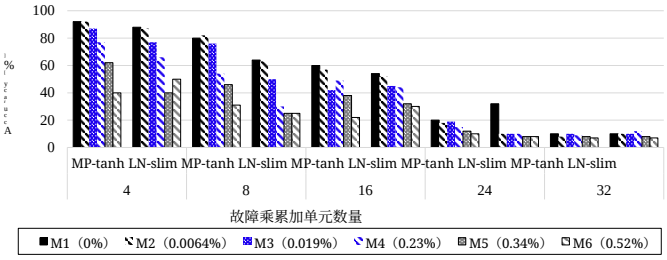


图 15：当固定型故障注入张量处理器中 16% 的乘加单元时，对基于近似乘法器 M*n* 的近似 MLP-双曲正切与近似 ML-base 的影响。各乘数的平均绝对误差标注在其旁边。

。例如，在第1层的 M1乘法器中出现固定为0故障时，MP-tanh 和 MP-softmax 分类的准确率分别为 8.27% 和 15.14%。同样，M3乘法器出现相同故障时，准确率分别为 9.12% 和 16.41%。在 FP-tanh 分类中也观察到了相同趋势。

3) 层宽度的影响：我们通过改变层大小，分析了加宽AxDNN对其故障韧性的影响。宽模型中的所有卷积层和全连接层均为基础模型中对应层的两倍宽。在此分析中，我们比较了近似ML-base模型和ML-wide模型，因为增加层宽度会降低AxDNN仿真的速度。我们在这些模型中不同故障MAC单元的最高有效位注入了固定型故障（占脉动阵列中MAC单元总数的16%）。如图14所示，我们的结果表明，加宽AxDNN不会显著改变其故障韧性，这说明即使存在故障，近似ML-base的表现也与近似ML-wide非常接近。

两个模型的精确版本均以97%的基准准确率进行训练，而在故障注入后，准确率降至52%—54%。有趣的是，即使存在近似误差，两者之间的分类准确率差异仍保持不变。这意味着，窄模型和宽模型具有几乎相同的故障韧性，且这种轻微差异与近似误差无关。这是因为易受影响参数的数量会随着模型宽度成比例增长，因此故障韧性保持不变。

4）模型架构的影响：图15比较了在张量处理器中不同百分比的乘加单元注入固定型故障时，使用近似MP-tanh和ML-base模型进行MNIST分类的结果。我们的结果表明，与近似多层感知机相比，近似卷积神经网络对固定型故障的抵抗能力较弱。例如，近似MP-tanh的分类准确率最高比近似ML-base低26%，尤其是在8%的乘加单元发生故障时。

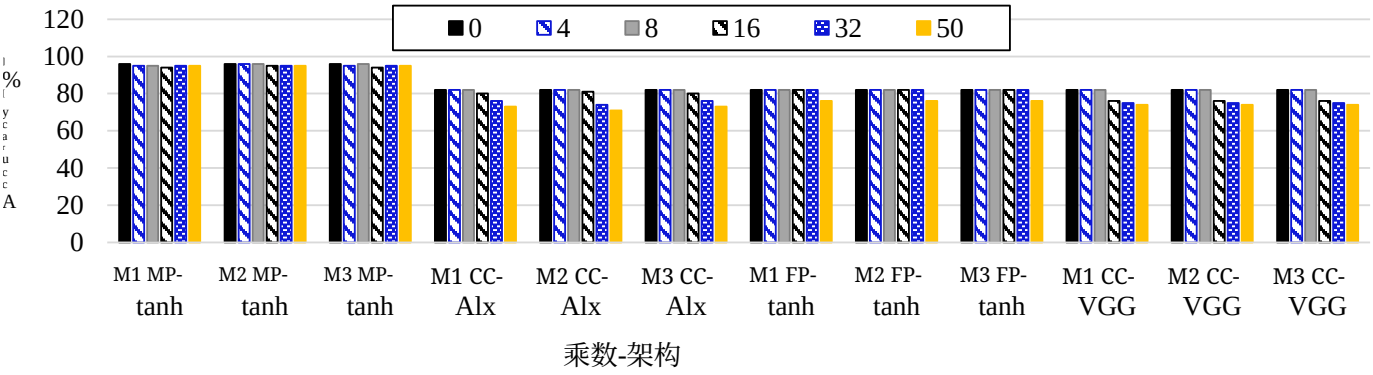


图 16：基于近似 MP-tanh、CC-Alx、FP-tanh 和 CC-VGG 的近似乘法器 M*n* 中固定型故障的 Fal-reTune 缓解方法。故障分别注入脉动阵列中 0%、4%、8%、16% 和 32% 的乘加单元。

G. 故障缓解分析

为分析所提出的Fal-reTune缓解方法的有效性，我们考察了脉动阵列中不同百分比的MAC单元发生固定型故障时，基于M1的AxDNN、基于M2的AxDNN和基于M3的AxDNN的情况。请注意，由于在第VI-D2节的CC-Alx、FP-tanh和CC-VGG分类任务中，M4、M5和M6在故障注入前的准确率过低，因此未纳入相关结果。回顾第VI-D2节的内容，即使32%的MAC单元发生故障，也会导致准确率降低8%。有趣的是，如图16所示，即使高达50%的MAC单元发生故障，Fal-reTune仍能将AxDNN的准确率恢复至接近AccDNN的基准准确率。有趣的是，由于Fal-reTune算法中的近似乘法器权重映射策略消除了近似误差，AxDNN的恢复准确率与Acc DNN的恢复准确率几乎相同。例如，无论使用精确乘法器还是近似乘法器，MP-tanh、CC-Alx、FP-tanh和CC-VGG的准确率分别仍高达98%、82%、81%和82%。我们还观察到，有时即使经过故障缓解，准确率仍会随着故障MAC单元百分比的增加而下降。原因是过多的MAC单元可能会断开，因此重新训练也难以再次实现更高的分类准确率。例如，当脉动阵列中映射的AxDNN有4%的MAC单元发生故障时，进行缓解后相比之下，若有50%的MAC单元发生故障，准确率仍低1%–6%。

H. 故障-能耗权衡探索

由于模拟一个 256x256 的脉动阵列需要大量硬件资源，因此我们使用来自 Evoapprox8b [45] 库的 M1 至 M9 乘数，分析了一个 8x8 脉动阵列的能耗。图 17 中的结果表明，在本文研究的所有 AxDNN 中，基于 M2 的 AxDNN 能效较低，而基于 M4、M5 和 M6 的 AxDNN 能效最高。相反，从第 VI-D 节的讨论可以明显看出，基于 M2 的 AxDNN 具有最强的故障韧性，并且

基于M4、M5和M6的AxDNN是本文研究的所有AxDNN中故障韧性最差的。因此，故障

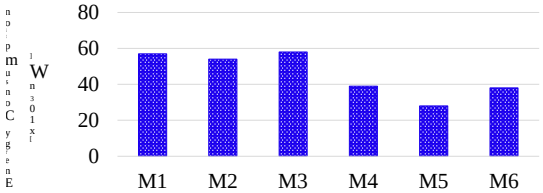


图17：Evoapprox8b [41] 库中M1至M6乘数构成的8×8近似脉动阵列的能耗分析

韧性与能效彼此正交。此外，基于M1和M2的AxDNN能耗略有差异。尽管如此，基于M6的AxDNN故障韧性不如基于M1的AxDNN。研究发现，基于M6的AxDNN能效和故障韧性都相当出色：在输入层最低有效位发生故障时，其在MNIST [52] 和Fashion MNIST [47]分类任务中的准确率分别超过90%和80%（如图10和图11所示）。

VII. 结论

近似计算以近似DNN的分类准确率换取能效，从而提升容错应用的性能。然而，近似DNN中的近似误差所导致的不精确性会降低其故障韧性。本文使用先进的Evoapprox8b [41]库，全面分析并缓解MLP和卷积神经网络等近似前馈神经网络因永久性故障造成的可靠性下降。我们提出一种新颖的故障缓解方法，即故障感知权重重新调优（Fal-reTune）：在存在故障时，利用权重映射函数重新调优权重，以提高分类准确率。为评估故障韧性以及所提缓解方法的有效性，我们采用了应用最广泛的MNIST、Fashion-MNIST和CIFAR10数据集。我们的结果表明，与准确DNN相比，故障对近似DNN分类准确率的影响更大。例如，近似DNN中的永久故障可导致准确率损失高达56%，而同一故障位在准确DNN中仅会导致4%的准确率损失。即使故障率高达50%，我们提出的Fal-reTune缓解方法仍能有效将分类准确率提升至98%。

参考文献

[1] A. Krizhevsky、I. Sutskever 和 G. E. Hinton，“采用深度卷积神经网络进行ImageNet分类”，《神经信息处理系统进展》，第25卷，2012年。[2] K. Simonyan 和 A. Zisserman，“用于大规模图像识别的超深卷积网络”，arXiv预印本 arXiv:1409.1556，2014年。[3] M. Capra、B. Bussolino、A. Marchisio、G. Masera、M. Martina 和 M. Shafique，“加速深度神经网络的硬件和软件优化：当前趋势、挑战及未来展望综述”，《IEEE Access》，第8卷，第225 134–225 180页，2020年。[4] N. P. Jouppi、C. Young、N. Patil、D. Patterson、G. Agrawal、R. Bajwa、S. Bates、S. Bhatia、N. Boden、A. Borchers 等，“张量处理单元在数据中心内的性能分析”，载于《第44届计算机体系结构年度国际研讨会论文集》，2017年，第1–12页。[5] G. Zervakis、I. Anagnostopoulos、S. Alsalamin、O. Spantidi、I. Roman-Ballesteros、J. Henkel 和 H. Amrouch，“近似DNN加速器的热感知设计”，《IEEE计算机汇刊》，2022年。[6] A. Malawade、M. Odema、S. Lajeunesse-DeGroot 和 M. A. Al Faruque，“Sage：一种用于高效端到端自动驾驶车辆控制的分体式架构方法”，《ACM嵌入式计算系统汇刊》(TECS)，第20卷，第5s期，第1–22页，2021年。[7] K.-C. Hsu 和 H.-W. Tseng，“GPTPU：使用边缘张量处理单元加速应用”，arXiv预印本 arXiv:2107.05473，2021年。[8] G. Zervakis、H. Saadat、H. Amrouch、A. Gerstlauer、S. Parameswaran 和 J. Henkel，“机器学习的近似计算：最新进展、挑战与愿景”，载于《2021年第26届亚洲及南太平洋设计自动化会议》(ASP-DAC)。IEEE，2021年，第189–196页。[9] G. Armeniakos、G. Zervakis、D. Soudris 和 J. Henkel，“深度神经网络加速器的硬件近似技术：综述”，《ACM计算调查》(CSUR)，2022年。[10] G. S. Rodrigues，´A. Barros de Oliveira、F. L. Kastensmidt、V. Pouget 和 A. Bosio，“通过故障注入评估连续近似计算算法的可靠性”，《电子测试期刊》，第35卷，第3期，第367–381页，2019年。[11] M. A. Hanif、F. Khalid 和 M. Shafique，“CANN：面向高性能深度神经网络加速器的可修正近似”，载于《2019年第56届ACM/IEEE设计自动化会议》(DAC)。IEEE，2019年，第1–6页。[12] M. Riaz、R. Hafiz、S. A. Khaliq、M. Faisal、H. T. Iqbal、M. Ali 和 M. Shafique，“CAXCNN：探索基于规范有符号数字的近似在硬件友好型卷积神经网络中的应用”，《IEEE Access》，第8卷，第127 014–127 021页，2020年。[13] S. Ganapathy、S. Venkataramani、B. Ravindran 和 A. Raghunathan，“DYVEDeep：动态可变计算量深度神经网络”，arXiv预印本 arXiv:1704.01137，2017年。[14] Z. Song、R. Wang、D. Ru、Z. Peng、H. Huang、H. Zhao、X. Liang 和 L. Jiang，“用于通用图形处理器上DNN训练加速的近似随机丢弃”，载于《2019年欧洲设计、自动化与测试会议暨展览》(DATE)。IEEE，2019年，第108–113页。[15] A. Siddique、K. Basu 和 K. A. Hoque，“探索近似DNN硬件加速器中的故障-能耗权衡”，载于《2021年第22届国际高质量电子设计研讨会》(ISQED)。IEEE，2021年，第343–348页。[16] R. Vidya Wicaksana Putra、M. Abdullah Hanif 和 M. Shafique，“SoftSNN：针对软错误的脉冲神经网络加速器低成本容错方案”，arXiv电子预印本，第arXiv–2203页，2022年。[17] S. Venkataramani、A. Ranjan、K. Roy 和 A. Raghunathan，“AxNN：采用近似计算的高能效神经形态系统”，载于《2014年IEEE/ACM低功耗电子与设计国际研讨会》(ISLPED)。IEEE，2014年，第27–32页。[18] V. Mrazek、S. S. Sarwar、L. Sekanina、Z. Vasicek 和 K. Roy，“面向近似人工神经网络的高能效近似乘法器设计”，载于《2016年IEEE/ACM计算机辅助设计国际会议》(ICCAD)。ACM，2016年，第1–7页。[19] V. Mrazek 等，“ALWANN：无需重新训练的深度神经网络加速器自动逐层近似”，arXiv预印本 arXiv:1907.07229，2019年。[20] S. Kundu、A. Soyyi˘git、K. A. Hoque 和 K. Basu，“深度神经网络加速器中制造故障的高层建模”，载于《2020年IEEE第26届在线测试与稳健系统设计国际研讨会》(IOLTS)。IEEE，2020年，第1–4页。[21] F. F. dos Santos、P. F. Pimenta、C. Lunardi、L. Draghetti、L. Carro、D. Kaeli 和 P. Rech，“分析并提高可靠性卷积神经网络在图形处理器上的应用，”*IEEE* 汇刊《可靠性》，第68卷，第2期，第663–677页，2018年。[22] J.-D. Guerrero-Balaguera、L. Galasso、R. L. Sierra和M. S. Reorda，《GPU上神经网络的可靠性评估：永久故障注入框架》，arXiv预印本 arXiv:2205.12177，2022年。[23] J. E. R. Condia、F. F. dos Santos、M. S. Reorda和P. Rech，《结合体系结构仿真与软件故障注入，快速且准确地评估GPU上卷积神经网络的可靠性》，载于2021年IEEE第39届超大规模集成电路测试研讨会（VTS），IEEE，2021年，第1–7页。[24] J. J. Zhang、T. Gu、K. Basu和S. Garg，《分析并缓解永久性故障对基于脉动阵列的神经网络加速器的影响》，载于2018年IEEE第36届超大规模集成电路测试研讨会（VTS），IEEE，2018年，第1–6页。[25] J. J. Zhang、K. Basu和S. Garg，《用于深度神经网络执行的容错型脉动阵列加速器》，《IEEE设计与测试》，第36卷，第5期，第44–53页，2019年。[26] M. Abdullah Hanif和M. Shafique，《SalvageDNN：通过显著性驱动的故障感知映射，挽救遭遇永久性故障的深度神经网络加速器》，《英国皇家学会哲学汇刊A辑》，第378卷，第2164期，第20190164页，2020年。[27] M. Sabih、F. Hannig和J. Teich，《利用可解释人工智能实现容错低精度深度神经网络》，载于2021年第51届IEEE/IFIP可靠系统与网络研讨会（DSN-W），IEEE，2021年，第166–174页。[28] M. Shafique、M. Naseer、T. Theocharides、C. Kyrkou、O. Mutlu、L. Orosa和J. Choi，《鲁棒机器学习系统：挑战、当前趋势、观点与未来方向》，《IEEE设计与测试》，第37卷，第2期，第30–57页，2020年。[29] B. Reagen、U. Gupta、L. Pentecost、P. Whatmough、S. K. Lee、N. Mulholland、D. Brooks和G.-Y. Wei，《Ares：量化深度神经网络韧性的框架》，载于2018年第55届ACM/ESDA/IEEE设计自动化会议（DAC），IEEE，2018年，第1–6页。[30] A. Ruospo、A. Bosio、A. Ianne和E. Sanchez，《评估卷积神经网络可靠性与其数据表示的关系》，载于2020年第23届Euromicro数字系统设计会议（DSD），IEEE，2020年，第672–679页。[31] O. Temam，《面向新兴高性能应用的缺陷容忍加速器》，载于2012年第39届年度国际计算机体系结构研讨会（ISCA），IEEE，2012年，第356–367页。[32] T. Tsai、S. K. S. Hari、M. Sullivan、O. Villa和S. W. Keckler，《NVBitFI：面向GPU的动态故障注入》，载于2021年第51届IEEE/IFIP可靠系统与网络国际会议（DSN），IEEE，2021年，第284–291页。[33] C. De Sio、S. Azimi和L. L. Sterpone，《FireNN：混合平台上的神经网络可靠性评估》，《IEEE新兴计算主题汇刊》，2022年。[34] F. Vaverka、V. Mrazek、Z. Vasicek和L. Sekanina，《Tfapprox：迈向GPU上深度神经网络近似硬件加速器的快速仿真》，载于2020年欧洲设计、自动化与测试会议暨展览（DATE），IEEE，2020年，第294–297页。[35] H. Kung、B. McDanel和S. Q. Zhang，《自适应分块：将固定尺寸脉动阵列应用于稀疏卷积神经网络》，载于2018年第24届国际模式识别会议（ICPR），IEEE，2018年，第1006–1011页。[36] S. Lym和M. Erez，《FlexSA：用于高效剪枝深度神经网络模型训练的灵活脉动阵列架构》，arXiv预印本 arXiv:2004 13027，2020年。。[37] R. H. Javed、A. Siddique、R. Hafiz、O. Hasan和M. Shafique，《ApproxCT：面向信息物理系统中高能效计算机视觉的近似聚类技术》，载于2018年第12届国际开源系统与技术会议（ICOSST），IEEE，2018年，第64–70页。[38] G. Zervakis、K. Tsoumanis、S. Xydis、D. Soudris和K. Pekmestzi，《通过部分积穿孔实现设计高效的近似乘法电路》，《IEEE 超大规模集成电路系统汇刊》，第24卷，第10期，第3105–3117页，2016年。[39] S. Ullah、S. S. Murthy和A. Kumar，《SMApproxLib：基于FPGA的近似乘法器库》，载于2018年第55届ACM/ESDA/IEEE设计自动化会议（DAC），IEEE，2018年，第1–6页。[40] B. S. Prabakaran、S. Rehman、M. A. Hanif、S. Ullah、G. Mazaheri、A. Kumar和M. Shafique，《DeMAS：一种用于构建基于FPGA系统的近似加法器的高效设计方法》，载于2018年欧洲设计、自动化与测试会议暨展览（DATE），IEEE，2018年，第917–920页。[41] V. Mrazek、R. Hrbacek、Z. Vasicek和L. Sekanina，《EvoApproxSB：用于电路设计及近似方法基准测试的近似加法器和乘法器库》，

近似方法基准测试，载于2017年欧洲设计、自动化与测试会议暨展览（DATE），”2017年。[42] S. Rehman、W. El-Harouni、M. Shafique、A. Kumar、J. Henkel和J. Henkel，《近似乘法器的架构空间探索》，载于*2016*年*IEEE/ACM*计算机辅助设计国际会议（*ICCAD*）。IEEE，2016年，第1–8页。[43] Z. Ebrahimi、S. Ullah和A. Kumar，《LEAP：基于前导1检测、精度可调的软核近似乘法器》，载于*2020*年第*25*届亚洲及南太平洋设计自动化会议（*ASP-DAC*）。IEEE，2020年，第605–610页。[44] ——，《SIMDIVE：面向现场可编程门阵列、精度可调的近似SIMD软乘除器》，载于*2020*年五大湖超大规模集成电路研讨会论文集，2020年，第151–156页。[45] V. Mrazek、R. Hrbacek、Z. Vasicek和L. Sekanina，《EvoApprox8b：用于电路设计及近似方法基准测试的近似加法器和乘法器库》，载于*2017*年欧洲设计、自动化与测试会议暨展览（*DATE*）。IEEE，2017年，第258–261页。[46] G. Cohen、S. Afshar、J. Tapson和A. Van Schaik，《扩展MNIST：扩展MNIST以包含手写字母》，载于*2017*年国际联合神经网络会议（*IJCNN*）。IEEE，2017年，第2921–2926页。[47] H. Xiao、K. Rasul和R. Vollgraf，《Fashion-MNIST：用于机器学习算法基准测试的新型图像数据集》，*arXiv*预印本 *arXiv:1708.07747*，2017年。[48] A. Krizhevsky、G. Hinton 等，《从微小图像中学习多层特征》，2009年。[49] A. Marchisio、V. Mrazek、M. A. Hanif和M. Shafique，《Feeca：面向低延迟、高能效胶囊网络加速器的设计空间探索》，*IEEE*超大规模集成电路系统汇刊，第29卷，第4期，第716–729页，2021年。[50] ——，《DescNet：为胶囊网络硬件开发高效暂存存储器》，*IEEE*集成电路与系统计算机辅助设计汇刊，第40卷，第9期，第1768–1781页，2020年。[51] R. V. W. Putra和M. Shafique，《FSpinn：面向高存储效率与高能效脉冲神经网络的优化框架》，*IEEE*集成电路与系统计算机辅助设计汇刊，第39卷，第11期，第3601–3613页，2020年。[52] L. Deng，《用于机器学习研究的手写数字图像MNIST数据库 [网络精选],》，《信号处理杂志》，第29卷，第6期，第141–142页，2012年。



Ayesha Siddique 于 2013 年获得巴基斯坦拉瓦尔品第 Army Public College of Management and Sciences 电气工程理学学士学位，并于 2018 年获得巴基斯坦伊斯兰堡 National University of Sciences and Technology（NUST）电气工程理学硕士学位。她目前正在攻读电气与计算机工程哲学博士学位，并在密苏里大学哥伦比亚分校可靠信息物理系统（DCPS）实验室担任研究生研究助理。

她获得学士学位后，于2013年至2017年在巴基斯坦伊斯兰堡的工程高级研究中心担任研究助理和设计工程师。她是 DAC 2018 Richard Newton奖学金的获得者。她的研究兴趣包括嵌入式系统的形式化分析与验证、可靠安全的机器学习、高能效系统以及数字图像处理。



Khaza Anuarul Hoque 于2011年和2016年分别获得加拿大蒙特利尔康考迪亚大学电气与计算机工程系的理学硕士和哲学博士学位。他目前是密苏里大学哥伦比亚分校（MU）电气工程与计算机科学系的助理教授，并负责指导可靠信息物理系统（DCPS）实验室。他的研究兴趣包括信息物理系统与嵌入式系统、机器学习、网络安全和形式化方法。

加入密苏里大学之前，他曾在英国牛津大学担任 FRQNT 博士后研究员。Hoque博士曾获得多项奖项和荣誉，包括 FQRNT博士后奖学金奖（2016年）、FQRNT博士研究奖（2012年）以及第八届IEEE国际NEWCAS会议最佳论文奖（2011年）。他是 IEEE 高级会员，也是 ACM、AAAS 和 IEEE信息物理系统技术委员会（CPS）的成员。