arXiv:2304.10618v1 [cs.AR] 20 Apr 2023

ULEEN：一种面向超低能耗边缘神经网络的新型架构

Zachary Susskind,Aman Arora德克萨斯大学奥斯汀分校Igor DantasDosSantosMiranda巴伊亚州雷孔卡沃联邦大学Alan Tendler LeibelBacellar,Luis Armando Quintanilla Villon, Rafael Fontella Katopodis里约热内卢联邦大学Leandro Santiago de Ara´ujo弗卢米嫩塞联邦大学DiegoLeonel CadetteDutra,PriscilaMachado Vieira Lima里约热内卢联邦大学FelipeMaiaGalv˜ao Franc¸a电信研究所˜，波尔图 里约热内卢联邦大学Mauricio Breternitz Jr.里斯本大学学院Lizy K.John德克萨斯大学奥斯汀分校

摘要——**在低功耗、实时边缘设备上部署人工智能模型，需要使用能耗、延迟和面积均为首要考量的加速器。在这一领域，已有多种方法可用于实现深度神经网络（DNN），包括剪枝、量化、压缩和二值神经网络（BNN）；然而，随着“极端边缘”场景的兴起，市场对效率更高的模型提出了新的需求。为满足超低能耗设备的限制，我们提出 ULEEN：一种基于无权重神经网络的模型架构。**

无权重神经网络（WNNs）是一类通过查表而非算术运算进行计算的神经网络模型。由于消除了能耗密集型算术运算，WNNs 从理论上非常适合边缘推理；然而，长期以来，它们一直面临准确率低、内存占用过高的问题。ULEEN 融合了算法改进，并采用受二值神经网络启发的新颖训练策略，在提升准确率和缩小模型大小方面取得了显著进展。

我们将 ULEEN 推理加速器的现场可编程门阵列（FPGA）和专用集成电路（ASIC）实现方案，与针对边缘设备优化的深度神经网络（DNN）和二值神经网络（BNN）设备进行比较。在 Xilinx Zynq Z-7045 现场可编程门阵列上，我们实现了基于 MNIST 数据集的分类任务，推理速度达到每秒 1430 万次（每焦耳 1300 万次推理），延迟为 0.21 μ 秒，准确率为 96.2%；相比之下，Xilinx FINN 的推理速度为每秒 1230 万次（每焦耳 169 万次推理），延迟为 0.31μ 秒，准确率为 95.83%。在 45nm 专用集成电路中，我们实现了每焦耳 510 万次推理、每秒 3850 万次推理的性能，准确率达到 98.46%；而量化后的 Bit Fusion 模型每焦耳可执行 9230 次推理，每秒可执行 19100 次推理，准确率为 99.35%。在不断探索更高能效边缘设备的过程中，ULEEN 表明，权重神经网络（WNNs）值得受到关注。

I. 引言

过去十年间，深度神经网络（DNN）推动了图像分类、自然语言处理（NLP）和医学诊断等领域的革命性进步，在某些情况下甚至达到了超越人类的准确率 [1]。这一进步得益于我们能够训练的模型在规模和复杂性上的指数级增长。与此同时，另一个方向的发展也在兴起：将人工智能模型部署在越来越小的设备上。剪枝、压缩、低精度量化 [2], 和 Once-for-All [3]等技术可以大幅降低大型模型的内存需求和计算需求，使其能够应用于低功耗、资源受限的边缘设备。然而，在计算直接邻近物理传感器进行的“极端边缘”场景中， [4], 能量预算极其有限，因此效率至关重要。这类设备通常部署在难以抵达的环境中，可能需要依靠小型电池运行数年甚至数十年，也可能通过直接从其测量的现象中采集能量来维持运行 [5]。

二值神经网络（BNNs） [6], [7], [8], [9], [10]将量化推向极致，把网络权重和激活值压缩为单比特值。通过用 XNOR 运算取代能耗密集型乘法运算，它们的能耗效率比 DNN 高出数个数量级。然而，与 DNN 一样，BNN 仍须让激活值经过多层计算。这对于实时边缘应用而言，仍可能构成显著的关键路径。

无权重神经网络（权重神经网络）是一类独特的

一种主要通过查找表而非算术或逻辑函数执行计算的神经模型。单个无权重神经元（也称为 RAM 节点）将二进制输入拼接起来，形成其查找表中的一个地址，并产生二进制响应。权重神经网络（WNNs）的灵感来源于生物神经元的树突结构，这些树突结构能够执行高度非线性的解码处理 [11]。与传统神经元或二进制神经元不同，RAM 节点能够学习非线性输入函数。

权重神经网络（WNNs）的概念并非新事物——其最早的实现可追溯至20世纪50年代 [12]。其简单的结构非常适合早期的超大规模集成电路（VLSI）技术，并在20世纪70至80年代取得了一定的影响力。然而，到20世纪90年代，这些早期的权重神经网络在准确率和内存效率方面均被深度神经网络（DNNs）超越。因此，尽管权重神经网络的简洁性和非线性特征使其非常适合部署在边缘设备上，但仍需要对算法和架构进行改进，才能使其性能达到或超越经过优化的深度神经网络和二值神经网络（BNN）模型。

本文展示了提高权重神经网络（WNNs）准确率并降低其硬件需求的技术。我们提出了一种称为 ULEEN（超低能耗边缘神经网络）的无权重神经网络架构，适用于能耗约束极为严格的推理应用。ULEEN 融合了一系列此前未应用于权重神经网络的技术，包括高效集成、剪枝，以及一种采用直通估计器 [13]的基于梯度的多轮次训练算法。我们还在以往工作的基础上进行了改进，引入了采用硬件友好型哈希的计数布隆过滤器、一种新颖的非线性温度计编码，以及基于阈值的一次性训练方法 [14], [15]。我们给出了该架构的推理加速器的现场可编程门阵列（FPGA）和专用集成电路（ASIC）实现，并将其与当前最先进的高效深度神经网络（DNNs）和二值神经网络（BNNs）推理平台进行了比较。

本文的具体贡献如下：1）提出 ULEEN，这是一种无权重神经网络架构，融合了先前权重神经网络的原理、受二值神经网络启发的训练方法，以及进一步的算法改进。通过这些改进，我们将已报告的最佳权重神经网络在 MNIST 数据集上的准确率从 91.5% 提升至 98.46%。2）提出一种适用于 ULEEN 的高能效推理加速器架构，可在现场可编程门阵列上实现，也可制成专用集成电路。3）将 ULEEN 加速器与专用集成电路和现场可编程门阵列上的最先进量化优化深度神经网络及二值神经网络进行比较。与 FINN [8]（现场可编程门阵列）相比，在准确率相同或更高的情况下，我们实现了 1.4–2.6 倍的延迟改善、1.2–2.6 倍的吞吐量提升，以及 6.8–8.5 倍的能耗改善。与 Bit Fusion [16]（专用集成电路）相比，在准确率降低 ¡1% 的情况下，我们实现了 2014–19549 倍的吞吐量提升和 479–663 倍的能耗改善。4）提供一套工具链，可采用单次或多次训练规则生成 ULEEN 模型；另有一套工具链，可利用我们的加速器架构从训练好的 ULEEN 模型生成 RTL。工具链见：为遵守双盲评审要求，*URL* 已省略。

本文其余部分安排如下：第 II 节将补充介绍权重神经网络的背景，以及一种特定的权重神经网络 WiSARD；我们以它为基础模型进行改进。第III 节将详细介绍 ULEEN 架构。第 IV 节将讨论软件和硬件实现细节。第 V 节将把我们的加速器架构与先前的深度神经网络和二值神经网络加速器进行比较，并将我们的模型架构与先前的高内存效率权重神经网络进行比较，同时提供更多结果以探究模型的性能。第 VI 节将进一步介绍该领域的相关先前工作。最后，第VII 节将讨论未来工作并作总结。

II. 背景

无权重神经网络（权重神经网络） 是使用查找表执行计算的神经模型。基本计算单元——随机存取存储器节点——是一个 n输入、 2^n输出的查找表，其中存储着经过学习的 1位条目。该表内容的每一种排列都代表一个唯一的布尔函数，这意味着单个随机存取存储器节点有 2^{2^n} 种可能的函数，其中许多是非线性的。相比之下，二值神经网络中的单个 XNOR 与popcount 神经元只能学习其输入的$n*2^n$线性函数中的一种。

这种表达能力的缺点在于，RAM 节点的大小会随着其输入数量的增加而呈指数级增长，很快就会变得难以处理。因此，WNN 中的层通常由许多 RAM 节点组成，每个节点仅对该层输入的一个子集敏感。

训练权重神经网络需要学习其组成随机存取存储器节点中的布尔函数。针对这一目标，已有多种方法得到探索，包括监督式 [17] 和无监督式 [18]技术。通常，权重神经网络采用监督式单次训练方法进行训练。所有随机存取存储器节点初始时均填充为零。编码后的输入依次输入网络。当某个随机存取存储器节点接收到一个训练输入时，它会将其存储器中对应的位置设为 1。需要注意的是，再次向节点输入相同的模式不会产生任何进一步影响；因此，进行多个训练周期并无优势。借助单次训练技术，权重神经网络的训练速度可比深度神经网络（DNN）以及其他广为人知的计算智能模型（如支持向量机（SVM） [19]）快多达四个数量级。

由于权重神经网络只能学习训练过程中接触到的模式，人们可能会认为它们难以泛化到新数据。然而，尽管某个推理样本可能与任何训练样本都不完全相同，但各个 RAM 节点看到的许多“子模式”也曾出现在训练数据中。因此，只要推理样本与训练样本的差异不太大，网络仍能有效地进行泛化。

我们在此仅介绍了权重神经网络（WNNs）的基础知识；目前已有许多变体得到探索，对于有兴趣深入了解的读者，我们建议参阅 [12]，以获取更为详尽的讨论。不过，我们将详细介绍一种权重神经网络模型架构——WiSARD，因为它是本文所讨论的改进方法的基准模型。

虽然这并非本文的重点，但计算机架构师可能会注意到，权重神经网络（WNNs）与现代微处理器中使用的预测器之间存在一些相似之处。例如，RAM 节点在概念上类似于分支预测器表，而使用拼接输入向量对 RAM 节点进行索引，在概念上类似于在基于表的分支预测器中使用分支历史寄存器。分支预测器中的表更新可以视为一次性训练规则的在线版本。

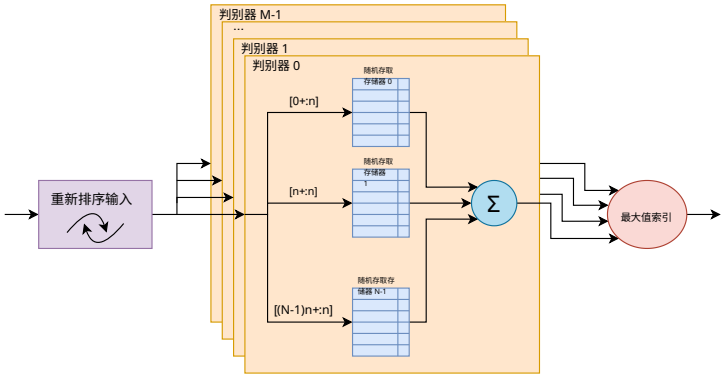


图1. WiSARD权重神经网络模型示意图，其中包含M 个类别、每个随机存取存储器节点有 n 个输入，且每个判别器有N 个随机存取存储器节点。该模型总计有MN 个随机存取存储器节点，以及$MN2^n$ 位状态。

WiSARD： 虽然权重神经网络的基础研究始于20世纪50年代，但首个取得广泛成功的权重神经网络架构是WiSARD（Wilkie、Stonham 和 Aleksander 的识别装置）[20]，，于1981 年推出，并自1984年起投入商业生产。WiSARD旨在用于异常检测等分类任务。图1所示的WiSARD 模型由称为判别器的子模型组成，每个子模型都专门对应一个可能的输出类别。这些判别器又由具有n个输入的随机存取存储器节点构成；对于一个具有I个输入的模型，每个判别器包含$N = I/n$ 个此类节点。网络的输入通过伪随机映射（也称为重排序）分配给随机存取存储器节点；通常，所有判别器共享同一个映射，这意味着不同判别器中索引相同的随机存取存储器节点具有相同的输入。

训练期间，输入只会提供给与正确输出类别对应的判别器。推理期间，输入会先提供给所有判别器。随后，将每个判别器中 RAM 节点的输出相加，得到响应值，并将响应最强的判别器所对应的类别作为网络的预测结果。图 2 展示了一个执行推理的简化 WiSARD 模型。在此示例中，由于输入图像包含数字“1”，判别器 1 的响应最强。

如果推理过程中遇到的输入与训练过程中遇到的输入完全相同，则相应判别器的所有 RAM 节点都将输出1，从而产生可能的最大响应。另一方面，如果某个模式相似但并不

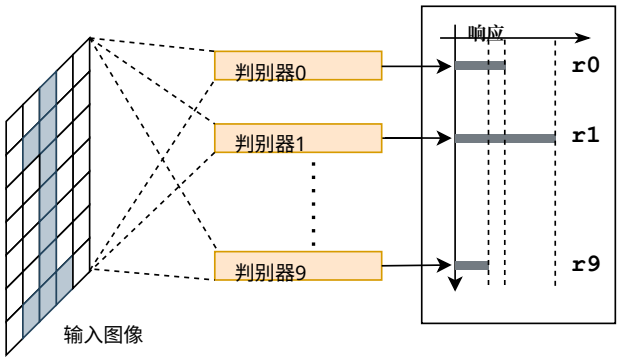


图 2. 用于识别数字的 WiSARD权重神经网络模型。在此示例中，输入图像包含“1”，对应的判别器产生了最强的响应。

与训练模式完全相同，那么部分RAM 节点可能会产生0，但许多节点仍会输出 1。只要正确判别器的响应仍强于任何其他判别器的响应，该网络就会输出正确的预测结果。实际上，与更简单的权重神经网络模型相比，WiSARD 具有强得多的泛化能力。

WiSARD 的性能与n的选择直接相关。较小的n 值会使模型只能学习简单模式，这可能提高其泛化能力，但也可能妨碍其有效学习。较大的n值会产生更专门化的行为，但也可能导致对训练数据过拟合 [20]。近期研究结果从形式上证明，WiSARD 的 VC 维数[1] 非常大 [22]，，这表明它在理论上具有很强的模式学习能力。

III. 提出的设计：ULEEN

ULEEN 模型在基准 WiSARD 模型的基础上进行了多项改进：(i) 计数型/连续布隆过滤器，(ii) 非线性温度计编码，(iii) 集成，以及 (iv) 剪枝。此外，ULEEN 模型可以通过两种方式进行训练：(i) 对传统单次训练技术进行改进，或 (ii) 一种新颖的多次训练算法，该算法受二值神经网络启发，以反向传播和直通估计器为基础。

A. 模型概述

图3 展示了 ULEEN 模型的整体架构，包括采用多轮次技术进行训练。输入会传入由多个较小模型组成的集成。我们发现，小模型集成的准确率优于单体大型模型，因此将这一设计融入 ULEEN。图中“向量化加法”模块负责聚合各个模型的结果。在多轮次训练期间，我们

[1]Vapnik–Chervonenkis（VC）维数用于衡量二分类算法 [21] 能够编码的一组函数所表示的知识的复杂度。虽然通常通过统计方法进行近似，但对于某些学习方法（包括 WiSARD），可以确定其精确的 VC 维数。

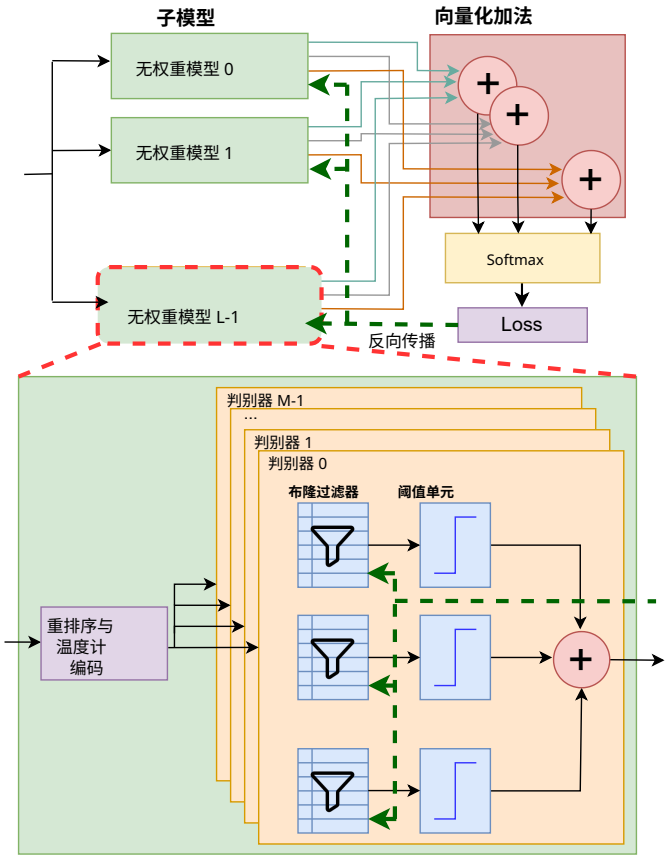


图 3. ULEEN 模型由多个子模型集成构成，每个子模型都是一个 WNN。每个子模型由判别器组成。训练期间，判别器使用连续布隆过滤器，以便通过基于梯度的方式更新权重。此图展示了多轮训练过程，该过程采用基于直通估计器的反向传播。

对聚合后的输出进行 softmax 运算，并计算交叉熵损失。

多位温度计编码用于表示输入，以获得更高的粒度。RAM 节点通过基于哈希的布隆过滤器实现压缩。在多轮训练期间，布隆过滤器内部保存浮点值，并使用单位阶跃函数将其二值化。

ULEEN 的多次训练技术是基于直通估计器对反向传播进行修改后得到的版本 [13]。反向传播期间的梯度流动如图中的绿色虚线箭头所示；阈值单元被视为恒等函数并予以忽略。剪枝是一种训练后技术，它会移除对整体准确率贡献最小的 RAM 节点，因此未在图中显示3。

1) 计数型/连续型布隆过滤器：影响 WiSARD 的一个主要挑战是，随着每个随机存取存储器节点的输入数量增加，模型大小会呈指数级增长。然而，在实际应用中，这些大型随机存取存储器节点学习的是仅包含少量最小项的布尔函数，这意味着其内容具有稀疏性。在 Bloom WiSARD [23], 中，研究证明，可以使用布隆过滤器等基于哈希的数据结构来减少其大小。

RAM 节点的数据占用量。布隆过滤器由一个查找表和 k 个相互独立的哈希函数组成，仅当与某个输入对应的所有 k 个哈希位置均为 1 时，才输出 1。由于不进行冲突检查，布隆过滤器可能产生假阳性。然而，研究发现，这对准确率的影响极小。

为了支持更复杂的训练方法，我们使用布隆过滤器的两种变体：计数型和连续型布隆过滤器。训练完成后，这些过滤器的内容会被二值化，并替换为传统布隆过滤器，以用于推理。图3 展示了在判别器中使用增强型布隆过滤器替代传统随机存取存储器节点的过程。

使用传统布隆过滤器训练 WiSARD 模型时，随机存取存储器条目会在训练过程中首次遇到某个模式时被设置，之后不会再更新。这种方法的问题在于，它将罕见模式与常见模式视为同等重要。衰减（Bleaching） [15] 是一种技术，用于在推理过程中忽略训练期间出现次数少于某个阈值 b 的模式。通常，这通过使用计数器表来实现。在 ULEEN 中，采用单次训练策略时，我们使用计数布隆过滤器替代布隆过滤器。计数布隆过滤器使用多位计数器代替单比特条目，从而能够在同一个模型中实现衰减和哈希。训练期间，每当遇到一个模式时，其对应计数器值中的最小值就会递增（如果存在多个相同的最小值，则同时递增这些计数器）。图 4 展示了计数布隆过滤器在推理过程中的工作方式。过滤器的输入经过哈希处理，这些哈希值用于访问相应的计数器值。对于输入 x_1, , 对应的计数器值为 $\{3, 4, 2\}$；对于x_2, , 对应的计数器值为 $\{3, 4, 3\}$。随后，找出访问到的最小计数器值（x_1 对应的值为 2；x_2对应的值为 3），并将其与阈值b 进行比较，以确定过滤器的响应。在此示例中，阈值 $b = 3$，因此 x_1 的输出为 0，而 x_2 的输出为 1。过滤器的响应变为“可能至少出现过 b 次”和“确定未出现过 b 次”。

使用多次投射策略进行训练时，我们进一步将布隆过滤器条目替换为浮点值。这些连续 布隆过滤器在最小条目大于或等于 0 时输出 1。使用连续布隆过滤器使我们能够采用基于梯度的技术进行训练。

由于我们的布隆过滤器所使用的哈希函数无需具备密码学安全性，且输入长度固定，因此我们可以使用简单的**无算术运算**哈希函数。具体来说，我们从 H3 系列中随机选取函数 [24]，。H3 系列是一组由简单逻辑运算构成的哈希函数，它们之间的差异仅在于随机参数的选择。Bloom WiSARD 使用基于 MurmurHash [25]算法的双重哈希技术，生成相互独立的哈希函数。尽管 MurmurHash 比密码哈希函数简单，但它旨在处理可变长度的输入，并且需要大量算术运算。

2) 高斯非线性温度计编码：权重神经网络（WNNs）的输入传统上采用 1 位值表示：如果某个输入大于训练数据中的均值，则表示为 1；否则表示为 0。

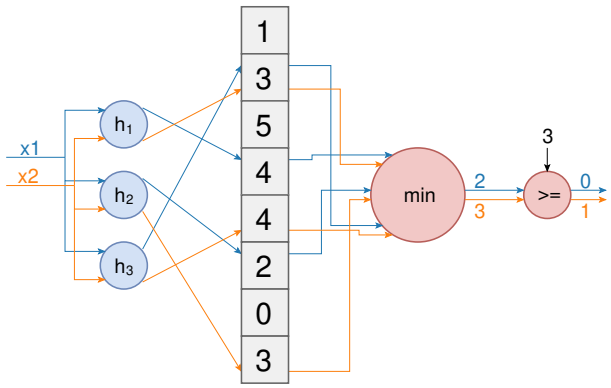


图 4. 使用三个哈希函数和阈值$b = 3$的计数布隆过滤器示例。

然而，使用单个位所能提供的粒度非常有限。整数编码并不是理想的选择，因为不同的位携带的信息量不同，这意味着在用于构成地址时，最低有效位本质上只是噪声。以往研究中，多位编码首选温度计编码 [26]。

温度计编码是一种一元编码方式：将一个值与一系列递增的阈值进行比较，结果的第i位表示该值与第i个阈值的比较结果。如图5 所示，输入值的变化类似于模拟温度计中的水银：随着值增大并越过阈值，各位从最低有效位到最高有效位依次置位。

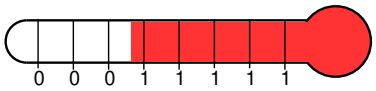


图 5. 在温度计编码中，位的置 1 方式如同水银升过温度计刻度：随着编码值增大，各位从最低有效位到最高有效位依次置为 1。

大多数采用温度计编码的先前研究都在阈值之间使用等间隔。该方法的缺点是，可能需要用大量位来编码离群值。在 ULEEN 中，我们则假设每个输入都服从正态分布，并根据训练数据计算其均值和标准差。对于t位编码，我们使用阈值将高斯分布划分为$t + 1$ 个概率相等的区域。这种高斯编码提高了取值范围中心附近值的分辨率。 [27] 也曾在有限范围内探索过高斯温度计编码；我们则更广泛地使用这种编码，并证明即使底层数据并非真正服从高斯分布，它依然有用。

3）高效集成：如图6所示，集成将多个弱分类器组合成一个强分类器。集成已在机器学习领域得到广泛研究，也是贝叶斯平均、提升法和装袋法等机器学习技术背后的核心思想 [28]。

设计集成模型时需要考虑的一个问题是，如何将多个子模型的预测合并为单一输出。如图 3 所示，ULEEN 采用

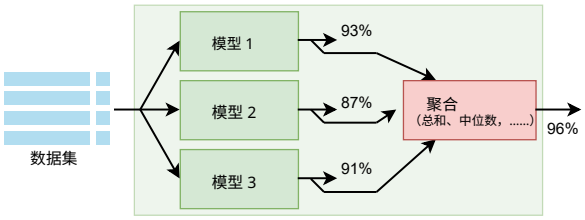


图6. 集成模型的简化示意图。一个集成结合多个弱分类器，构建一个更强大的单一分类器。

通过对所有子模型中每个判别器的响应值求和，得出子模型的结果。换句话说，一个包含L 个子模型和M 个类别的模型将生成一个$L \times M$ 响应矩阵R，以及一个包含M个元素的顶层响应向量$\vec{r} \triangleq R\,\vec{1}$。

人们可能会合理地认为，使用集成会增加模型总大小，因为 RAM 节点的数量要多得多。但在实践中，我们发现情况未必如此——请记住，集成的关键优势在于能够组合弱分类器。这意味着，集成中的各个子模型可以比独立模型小得多，而不会损害集成准确率。

在由每个滤波器输入数量不同的子模型组成的集成中，输入较多的滤波器也可以有选择地“忽略”大多数输入均为无关项的简单模式，而依靠输入较少的滤波器来捕获这些模式。这有助于减少布隆过滤器所需的表项数量。

由于推理所需的哈希运算量会随子模型数量线性增加，我们避免使用包含大量子模型的集成。我们也不使用一次性训练规则训练集成，因为当较小的布隆过滤器包含大量输入时，该规则会导致准确率较低。

研究了 WiSARD 模型的集成 [29]，但未采用反向传播或多轮训练，结果模型规模庞大，准确率提升却微乎其微。

4) 剪枝： 剪枝从模型中移除参数和连接，以减小其规模和复杂度。优化后的深度神经网络 [3], [30], [31], [32], [33]已应用剪枝技术并取得了出色的效果，但类似概念尚未用于以往的权重神经网络。

训练后，计算模型中每个 RAM 节点的输出与正确模型输出之间的相关性。移除每个判别器中相关性最低的固定比例 RAM 节点。由于这种剪枝会降低判别器可能达到的最大响应，因此接下来我们学习一组加到判别器输出上的整数偏置。最后，使用多次训练过程对剩余的 RAM 节点进行微调。

在我们的实验中，我们发现通常可以剪去约 30% 的 RAM 节点，相应地减小模型大小，同时对准确率的影响极小。尽管这无法达到 DNN 通常可实现的剪枝程度，但仍能显著降低 ULEEN 的内存需求。在集成中使用时，偏置可以

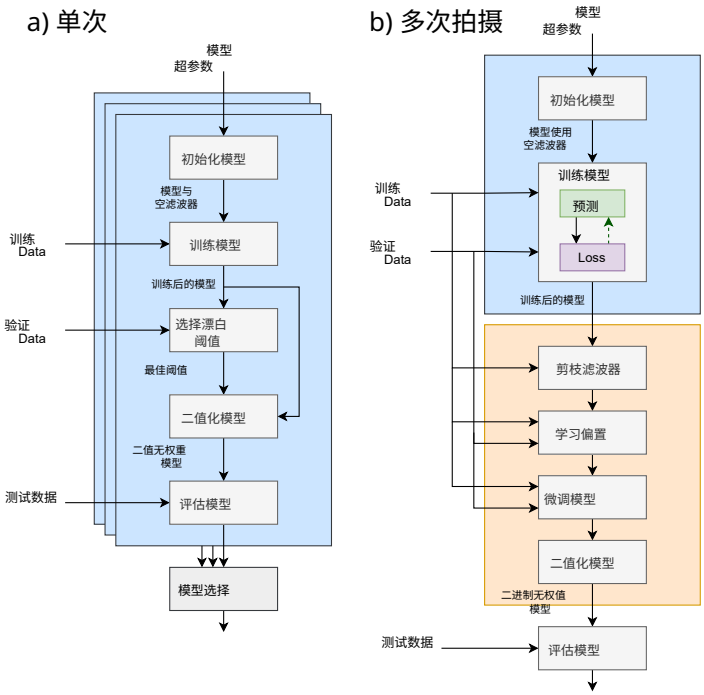


图 7. ULEEN 支持单次训练和多次训练两种技术。(a) 在单次训练中，将编码后的训练样本依次输入模型，并用于更新计数器值。随后，通过优化漂白阈值来消除罕见模式。(b) 在多次训练中，采用基于梯度的更新规则，在多个训练周期内学习计数器值。接下来，剪除相关性较弱的滤波器，并将其替换为恒定偏置，然后对剩余滤波器进行微调。

可以跨子模型求和，这意味着剪枝的唯一开销是一次加法。

B. 训练 *ULEEN*

我们对 ULEEN 采用单次训练技术和多次训练技术。单次训练技术与传统的 WiSARD 模型训练方式类似。多次训练技术能够提供更高的准确率，尤其是在数据集较大时；使用集成模型和剪枝也必须采用这种技术，但与单次训练方法相比，它所需的训练时间要长得多。

1) 单次训练：单次训练技术是一种计算上简单的方法，与以往训练权重神经网络的方式类似。该过程总结于图 7a。超参数包括温度计编码以及布隆过滤器的配置信息（输入数量、表大小和哈希函数数量）。训练过程中，编码后的输入样本会依次传入与正确输出类别对应的判别器。计数布隆过滤器按照先前讨论的方法（第III-A1节）递增计数器，以更新其内容。

训练后，使用漂白 [15]丢弃训练期间很少出现的模式。漂白会找出一个值b，使训练期间出现次数少于b 次的所有模式都被丢弃。如果不使用漂白，在训练数据集较大时，几乎所有随机存取存储器节点条目都可能变为 1，从而损害泛化能力。这种现象称为饱和。不使用漂白的权重神经网络需要仔细地手动提取特征并选择数据，以避免饱和。

我们采用二分搜索策略，寻找能使验证集上的模型准确率最大化的b 值。随后，将所有小于b 的过滤器值替换为 0，其余过滤器值替换为 1。这样，推理时便可用传统二进制布隆过滤器替代计数布隆过滤器，从而简化硬件。

2) 多次训练：虽然单次训练技术效率很高，但由于无法在训练过程中考虑反馈，其效果最终受到限制。然而，传统的基于梯度的技术不适用于权重神经网络，因为布隆过滤器的输出是二进制的，而非连续值。二值神经网络也面临类似问题，因此人们开发了适用于二进制权重的基于梯度的训练算法。我们的多次训练技术基于 [7] 中描述的二值神经网络训练算法。

ULEEN 采用连续型布隆过滤器，其条目为介于 -1.0 和 1.0 之间的浮点数，以实现多次采样技术训练。在前向训练过程中，使用单位阶跃函数将条目二值化：

$$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

反向传播不适用于单位阶跃函数，因为除了 $x = 0$ 之外（此处导数为无穷大），其导数处处为 0。然而，在计算梯度时，我们将单位阶跃函数视为恒等函数 $f(x) = x$，即 $f'(x) = 1$。这种称为直通估计器的技术已用于训练低精度和二值网络 [13], ，但我们是首个将其用于权重神经网络的研究。由于我们的模型只有单层，因此无需通过索引和哈希操作传播梯度。

图 7b 总结了我们的多轮训练过程。训练完成后，根据第 III-A4 节讨论的方法对模型进行剪枝和微调。剪枝后，通过应用单位阶跃函数将连续布隆过滤器二值化。

模型使用 Adam 优化器 [34] 进行训练，基础学习率为 10^{-3}。权重采用均匀分布 $\mathcal{U}(-1,1)$进行初始化。为防止过拟合，我们在滤波器的输出端添加了 Dropout 正则化 [35] ($p = 0.5$)。对于 MNIST [36] 数据集，我们还尝试了一种简单的数据增强方法：将训练集中的每张图像复制 9 份，并分别在水平方向和垂直方向上移动 -1 到 1 个像素。

C. 推理加速器架构

图 8和 9展示了我们流水线式 ULEEN 推理加速器的框图。为简化控制逻辑，芯片上的各单元尽可能以锁步方式运行。这意味着必须先读入完整的输入样本，才能开始计算。此反序列化过程由总线接口执行（图中未显示）。输入数据还可以选择通过将一元温度计编码值替换为表示置位位数的二进制值进行压缩。这样可减少数据传输量

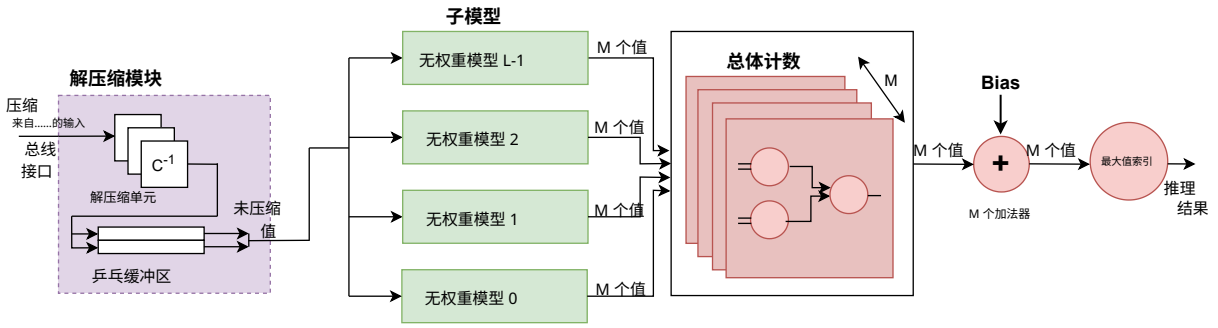


图 8. ULEEN 推理的示意图 子模型。子模型的输出 推理加速器架构。输入会先进行反序列化，并在需要时解压缩，然后传递给集成 o f 会被求和并加上偏置，得到每个类别的响应，并将响应最强者的索引作为预测结果 .

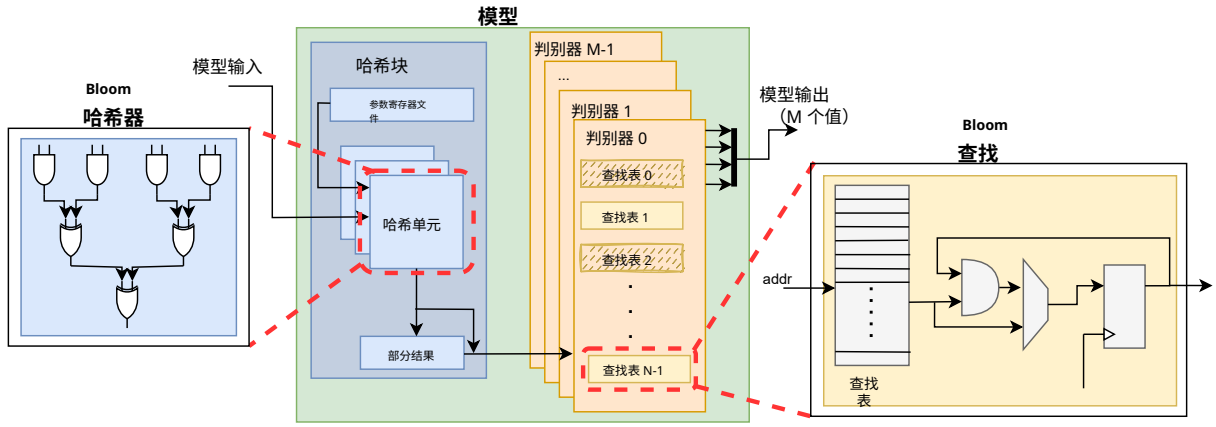


图 9. ULEEN 推理加速器中各子模型的细节。每个子模型包含一个用于计算哈希函数的硬件模块，以及若干用于执行查找的硬件单元。这些模块共同构成布隆过滤器，并被划分为独立单元以消除冗余计算。

片外，但需要一个解压缩单元来还原温度计编码，如图左侧所示 8。如果输入数据未压缩，则可以从设计中移除此单元。

图中的判别器9 展现了多项优化。尽管布隆过滤器中的每个 H3 哈希函数都需要不同的随机参数，但在子模型的所有布隆过滤器之间共享这些参数并无弊端。因此，我们使用一个中央寄存器文件（图中标为“Param RF”）来存储哈希参数。由于子模型中的判别器共享输入顺序，在哈希参数也共享的情况下，所有判别器都会接收到相同的哈希值。因此，无需为每个判别器单独计算哈希；我们改为使用单个中央哈希块。哈希块本身由多个流水线化的哈希单元组成，以每周期 1 次哈希的吞吐量处理输入序列。如图9左侧所示，这些哈希单元仅执行 与 和 或 运算，不涉及算术运算。如果哈希过程完全并行化，设计性能将严重受限于片外带宽。因此，我们将哈希单元的数量减少到足以达到最大吞吐量的最低数量，并在缓冲区中累积部分哈希结果。一旦缓冲区已满且最后一个部分哈希结果可用，所有布隆过滤器会同步执行查找。

由于哈希运算被移至中央模块，判别器本身仅包含查找单元。如图 9 右侧所示，查找单元由一个查找表和用于执行 与 归约的硬件组成。每个查找单元中的 1 位累加器可以将查找表的输出，或该输出与其当前内容的 与 结果作为输入。完成所有哈希查找后，查找单元的输出将被标记为有效。

集成中的每个子模型都必须计算自己的哈希值，因为输入顺序以及哈希输入和输出的位宽各不相同。由于不同子模型的表内容、大小和剪枝情况各异，因此它们也各自拥有独立的过滤器单元组。

总体计数和子模型响应累加通过一组加法树完成，如图 8右侧所示。偏置值被加到这些结果上（第 III-A4节），并计算响应最强项的索引，以生成最终的推理结果。

IV. 评估方法

我们将 ULEEN 与最先进的二值神经网络、量化深度神经网络和权重神经网络进行比较，采用基于 FPGA、专用集成电路和

软件模型的设计。本节介绍各种比较所采用的方法。

FINN [8], [37] 是 Xilinx 研究实验室开发的、用于 FPGA 上 BNN 推理的框架，我们将其作为 FPGA 实现的对比基准。我们采用了 [8]中描述的 SFC-max、MFC-max 和 LFC-max 网络。这些是用于 MNIST 数据集分类的三层全连接网络拓扑 [36], ，具有不同数量的神经元（SFC 的每层包含 256 个神经元，MFC 的每层包含 512 个神经元，LFC 的每层包含 1024 个神经元）。后缀“max”表示这些设计经过性能优化，旨在实现较高的峰值吞吐量。据我们所知，这些是目前 FPGA 上用于 MNIST 的效率最高的 BNN 实现。

我们将 ASIC 实现与 Bit Fu- sion [16], [38], 进行比较。Bit Fusion 是一种支持动态精度的量化 DNN 硬件架构。Bit Fusion 使用高度量化的 LeNet-5 架构（2 位；三值）对 MNIST 进行分类。该 NN 架构在 [39] 中提出，并且据称其准确率接近原始 LeNet5。我们采用了针对该模型优化精度的 Bit Fusion 设计；也就是说，最小和最大精度均为 2 位。我们针对该工作负载确定了三种高效的加速器变体，其配置参数如下：

{S=8x8，WBUF=32KB，ABUF=16KB，OBUF=8KB}{S=16 x16，WBUF=64KB，ABUF=32KB OBUF=16KB}{S=32 x32，WBUF=64KB，ABUF=32KB，OBUF=16KB}

此处，“S”表示 Bit Fusion 核心中脉动阵列的大小，WBUF、ABUF、OBUF 分别表示权重、激活值和输出缓冲区的大小。在第V节中，我们分别将这些变体称为 BF8、BF16 和 BF32。为实现 Bit Fusion 加速器，我们从作者处获取了 RTL [16] ，并使用 Cadence RTL Compiler 2017 和 FreePDK45 [40] 库对其进行了综合。我们使用 Bit Fusion 模拟器 [38], ，该模拟器将 Cadence 报告的核心能耗与 Cacti 报告的随机存取存储器/缓冲区能耗相结合 [41], ，用于评估推理的总能耗。

为了评估与以往权重神经网络相比，我们的改进所带来的影响，我们在最初用于评估该模型的相同九个多类别分类数据集上，将 ULEEN 模型的大小和准确率与 Bloom WiSARD [23] 进行了比较。对于没有明确划分训练集和测试集的数据集，我们采用了 2:1 的训练集/测试集划分比例。

A. 软件实现

我们使用 Python 实现了一次性训练（one-shot training），并采用 Numba JIT 编译器优化对性能要求较高的代码部分。该代码仅支持 CPU，且采用单线程执行，但我们使用了一台配备 64 个核心（2× Intel E5-2698 v3）的机器并行训练多个模型。我们采用半自动化的参数扫描方法来确定最优的模型超参数。具体而言，我们对每个布隆过滤器的输入数量、条目数量和哈希函数数量，以及每个输入对应的温度计编码位数，在不同的值范围内进行了扫描。随后，我们利用此次扫描的结果来进行……

以识别在准确率与模型大小之间实现良好平衡的模型。

多轮训练流程的计算开销显著更大，因为除了需要进行多个训练周期外，还需要计算梯度。然而，该方法能够提高准确率，并且是使用集成和剪枝的必要条件。我们使用 PyTorch 机器学习框架实现了多轮训练，并在 NVIDIA Tesla M40 GPU 上运行训练任务。布隆过滤器的前向传播和反向传播被实现为单个多维 gather/scatter 操作，从而实现了高效的内存并行计算。尽管进行了这些优化，其训练时间仍远长于单轮算法。我们认为，该算法能够从新一代 GPU 所提供的更高内存带宽和更大缓存中显著获益。

B. 硬件实现

ULEEN 加速器源代码使用 Mako 模板化 SystemVerilog 编写。Mako [42] 是一个模板库，可将 Python 解释器作为预处理步骤，用于处理任意语言编写的代码。我们使用 Mako 自动从训练好的模型中提取参数、构建状态机，并确定要实例化的功能单元数量，从而在达到吞吐量目标的同时尽量减小面积。借助这种模板机制，只需更改命令行参数，我们就能生成新的加速器。我们使用 Synopsys VCS 2018 对设计进行仿真，以确保功能正确，并评估加速器的延迟和吞吐量。

在 FPGA 实现与对比中，我们使用 Zynq Z7045 SoC 平台，该平台也曾用于 FINN。我们的设计具有与 FINN 相同的 I/O 接口位宽（112 位）。我们使用 Xilinx Vivado 2019.2 对设计进行综合与实现。我们采用了与 FINN 相同的工作频率（200 MHz），但由于 FPGA 布线拥塞，在某些情况下未能达到这一频率。资源使用情况和功耗数据均通过 Vivado 报告获取。

对于专用集成电路的实现和比较，为了确保公平比较，我们采用与 Bit Fusion 相同的工具和库（FreePDK45 [40] 和 Cadence RTL Compiler 2017）。我们的设计也采用相同的 192 位接口宽度。工作频率为 500 MHz，与 Bit Fusion 设计相同。我们使用 Cadence RTL Compiler 报告的面积和功耗数据（我们的设计中没有随机存取存储器模块，因此不需要使用 Cacti，这一点不同于 Bit Fusion）。

V. 结果

A. 选定的 *ULEEN* 模型

我们确定了三种 ULEEN 模型（ULN-S、ULN-M、ULN- L），如表 I所示，它们针对尺寸/准确率曲线上的不同设计点。所有子模型的每个滤波器都使用两个哈希函数——我们发现，使用一个哈希函数并非总能避免过多的冲突，而使用两个以上的哈希函数会增加硬件成本，却不能明显提高准确率。同一模型中的所有子模型均采用

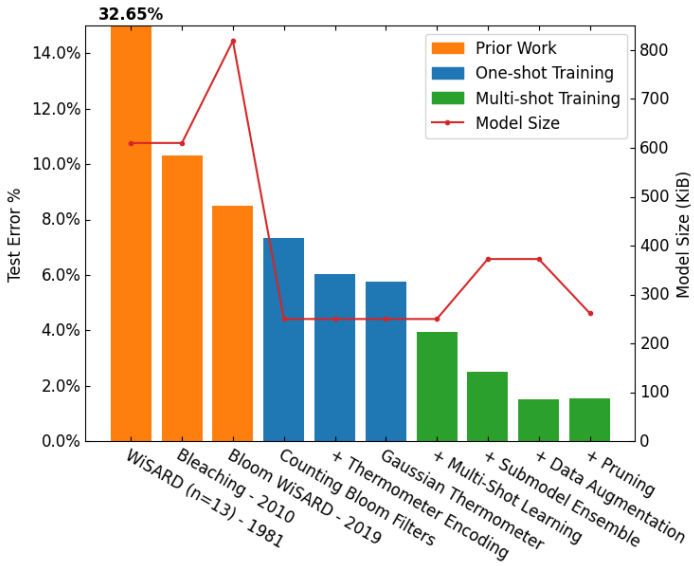


图 10. 我们对权重神经网络（WNNs）所做改进的迭代影响。我们展示了三个先前的结果，包括此前表现最佳的模型（Bloom WiSARD），以及随着我们逐步引入 ULEEN 的优化措施，误差和模型大小方面的改进。结果基于 MNIST 数据集。

相同的温度计编码，且所有模型均按 30% 的比例进行剪枝。如表所示，集成由 3 到 6 个子模型组成，集成准确率介于 96.20% 和 98.46% 之间。部分子模型的准确率低至 80%；然而，正如第 III-A3节所述，集成能够组合弱分类器，从而得出准确的预测。

表 I 所选 ULEEN 模型的详细信息。SMX 指构成集成模型的各个子模型。

模型	Sub-模型	比特/Inp	/筛选	/Filter	Size	Test 准确率（%）
ULN-S	集成	2	{}	{}	16.9	96.20
	SM0	2	12	64	7.19	92.91
	SM1	2	16	64	5.39	90.25
	SM2	2	20	64	4.38	86.16
ULN-M	集成	3	{}	{}	101	97.79
	SM0	3	12	64	10.9	83.54
	SM1	3	16	128	16.0	90.93
	SM2	3	20	256	26.0	92.92
	SM3	3	28	256	18.44	87.05
	SM4	3	36	512	29.38	80.93
ULN-L	集成	7	{}	{}	262	98.46
	SM0	7	12	64	25.0	88.78
	SM1	7	16	128	37.7	93.24
	SM2	7	20	128	30.2	92.44
	SM3	7	24	256	50.3	93.92
	SM4	7	28	256	43.1	90.47
	SM5	7	32	512	75.6	90.44

B. ULEEN 改进方案的性能

图 10 展示了我们在 ULEEN 中采用的优化所带来的影响。前三种模型是先前的研究成果，包括最初的 1981 年 WiSARD 架构 [20]以及最先进的 2019 年 Bloom WiSARD 架构 [23]。后续模型逐步纳入了更多改进。我们的最终模型 ULN-L 与 BloomWiSARD 相比，误差降低了 82%，模型大小缩减了 68%。采用多次学习和集成模型显著提升了准确率，而剪枝则大幅缩减了模型大小。

C. 现场可编程门阵列对比（与 *FINN* 对比）

表 II 和图 11 展示了 ULEEN 与 FINN 的比较。ULN-S、ULN-M 和 ULN-L 分别与 FINN 的 SFC、MFC 和 LFC 模型进行比较。我们报告加速器延迟（单位为 μ秒）以及峰值吞吐量（每秒数千次推理，kIPS）。我们还给出了单个样本独立处理时的能耗（batch=1）以及稳态下每次推理的能耗（batch=∞）。请注意，FINN 未提供 MFC 模型的部分结果。

表 II ULEEN 与 FINN 模型 SFC、MFC、LFC（现场可编程门阵列）的比较。FINN 对应的行以阴影标示。N/A 表示无可用数据。

模型	延迟 (μs)	吞吐量（千条指令每		μJ/次推理 b=1	μJ/次推理 b=∞	LUT	BRAM	Test 准确率（%）
ULN-S	0.21	14,286	1.1	0.234	0.077	17,319	0	96.20
SFC	0.31	12,361	7.3	2.263			4.5	95.83
ULN-M	0.29	14,286	3.1	0.887	0.214	49,445	0	97.79
MFC	N/A	6,238	11.3	N/A		N/A	N/A	97.69
ULN-L	0.94	4070	3.4	3.148	0.826	123 ,117	0	98.46
LFC	2.44	1561	8.8	21.5		,988	396	98.40

总体而言，与 FINN 相比，我们的延迟降低了 1.4–2.6 倍，吞吐量提高了 1.2–2.6 倍，且在所有三种情况下准确率都更高。能耗方面的改善幅度更大。单次推理的能耗降低了 6.8–9.6 倍，稳态推理的能耗降低了 6.8–8.5 倍。这些改进清楚地表明，权重神经网络甚至能够胜过经过优化的二值神经网络实现。

所有三种 FINN 模型以及 ULN-S 和 ULN-M 均以 200 MHz 实现。由于现场可编程门阵列布线拥塞以及由此产生的长连线，我们只能将大型设计实现为 85 MHz。

D. 专用集成电路对比（与 *Bit Fusion* 相比）

表III 和图12 展示了 Bit Fusion 与 ULEEN 的对比。Bit Fusion 的设计批次大小为 16；为确保公平比较，我们对 ULEEN 也采用相同的批次大小。ULEEN 的单次推理（批次=1）延迟介于 0.032 μ秒（ULN-S）和 0.056 μ秒（ULN-L）之间。速度最快的 Bit Fusion 模型（BF32）需要 838μ秒，但由于该模型同时以 16 个批次执行推理，因此难以直接与单次推理延迟进行比较。

Bit Fusion 实现了一个激进量化的 LeNet-5 模型，准确率比 ULN-L 高出 0.89%。

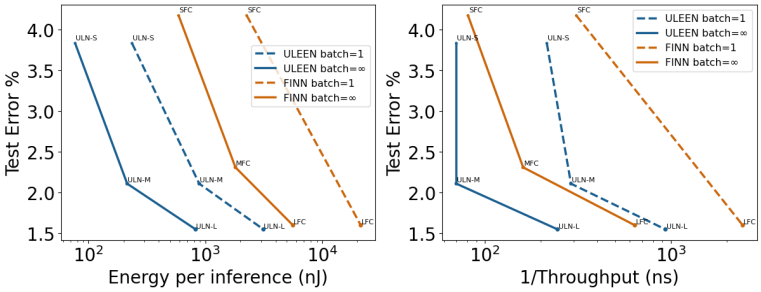


图 11. ULEEN 和 FINN 的能耗与吞吐率倒数相对于误差的帕累托前沿。图中展示了单次推理（batch=1）和稳态（batch=∞）下的值。请注意，对于 batch=1，吞吐率倒数等于延迟。

表 III ULEEN 与 Bit Fusion/LENET-5（专用集成电路）的比较。Bit Fusion 对应的行以阴影标示。

模型	Xput (千条指令每秒)	功耗 (W)	纳焦/推理 b=16	Area (mm^2)	Test 准确率（%）
ULN-S	55,556	0.84	17.5	0.61	96.20
ULN-M	55,556	2.58	57.1	2.09	97.79
ULN-L	38,462	6.23	195.5	5.22	98.46
BF8	2.0	0.26	129,731	0.60	99.35
BF16	7.1	0.81	114,914	1.59	99.35
BF32	19.1	1.79	93,589	1.65	99.35

然而，ULN-L 的能耗降低了**479-663x** ，吞吐量则提高了**2014-19549x**。Bit Fusion 等模型适用于手机等设备，这些设备通常可以接受数百 μJ 的能量预算。然而，对于极端边缘设备，或者在需要极高吞吐量的大型设备上，ULEEN 都是更优的解决方案。

请注意，Bit Fusion 加速器是一种旨在运行任意量化卷积神经网络的硬件架构。这里展示的是在 Bit Fusion 上运行一个量化（三值）LeNet-5 模型。改变 Bit Fusion 加速器的规模（例如脉动阵列的维度）会影响加速器的性能，但不会影响模型的准确率。另一方面，在 ULEEN 中，加速器规模与模型大小相互关联。ULEEN 建立了准确率、效率和面积之间的权衡关系，可根据具体应用进行探索。

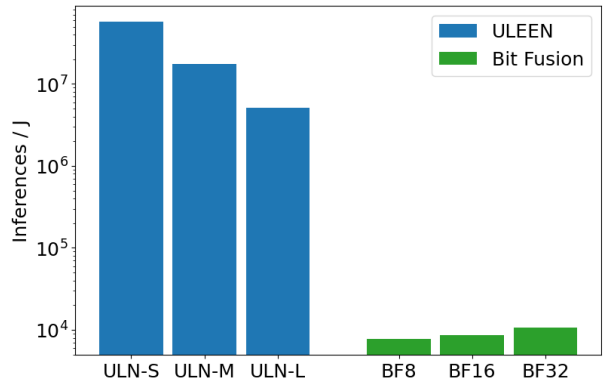


图 12. ULEEN 和 BitFusion 的功耗效率（每焦耳推理次数）

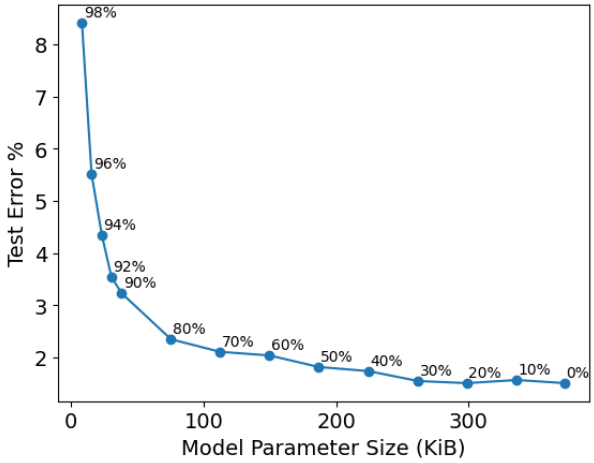


图13. ULN-L 中剪枝后大小与误差的关系。每个数据点旁的 % 值表示所进行的剪枝量。

E. 模型比较（与 *Bloom WiSARD* 对比）

Bloom WiSARD 通过使用（二进制）布隆过滤器进行压缩，以极小的模型大小实现了与大型传统权重神经网络相当的准确率 [23]，确立了高效权重神经网络的最新水平。我们在与 Bloom WiSARD 评估中相同的九个多类别分类数据集上评估 ULEEN，数据集见表 IV。在 MNIST 上，我们使用 ULN-L 将测试准确率从 91.5%（此前小于 1 MB 的权重神经网络模型中的最佳结果）提升至 98.46%。在所有情况下，ULEEN 都能生成准确率更高、规模更小的模型，模型大小平均减少 46.1%，测试误差降低 49.8%。

ULEEN 在 Shuttle 数据集上实现了惊人的 ~99% 误差降低。Shuttle 是一个异常检测数据集，其中 80% 的训练数据属于“正常”类别 [43]。我们推测，由于 Bloom WiSARD 未采用漂白机制，与该类别对应的判别器在训练过程中变得饱和。

表 IV ULEEN（ULN-L）与此前最佳 WNN 工作（Bloom WiSARD）的准确率和模型大小 [23]。

模型名称	测试 准确率（%） (Bloom WiSARD)	(ULEEN)	大小（KiB） (Bloom WiSARD)	大小（KiB） (ULEEN)
MNIST [36]	91.5	98.5	819	262
大肠杆菌 [44]	79.9	87.5	3.28	0.875
鸢尾花 [45]	97.6	98.0	0.703	0.281
字母 [46]	84.8	90.0	91.3	78.0
卫星图像 [47]	85.1	88.0	12.7	9.00
航天飞机 [43]	86.8	99.9	3.69	2.63
车辆 [48]	66.2	76.2	4.22	2.25
元音 [49]	87.6	90.0	6.44	3.44
葡萄酒 [50]	92.6	98.3	2.28	0.422

F. 敏感性分析

1) 剪枝：图 13 展示了在不同目标比例下对 ULN-L 进行剪枝所实现的模型大小和准确率。我们以 10% 为步长，将剪枝比例从 0% 扫描至 90%，并以 2% 为步长，从 90% 扫描至 98%。我们观察到，当剪枝比例不超过 30% 时，准确率几乎没有损失，而此后准确率仅呈现逐渐

退化幅度最高可达 80%，此时模型大小为 75.31 KiB，准确率达到 97.65%。超过这一临界点后，准确率会迅速下降，但即使剪枝率达到 98%，我们的测试误差仍低于 10%。

虽然模型大小会随着剪枝比例的增加而成比例减小，但所需的哈希次数并不会相应减少，因为不同判别器中的过滤器可能会被剪枝。因此，实际上，采用较高的剪枝比例可能不如直接训练一个更小的模型高效。

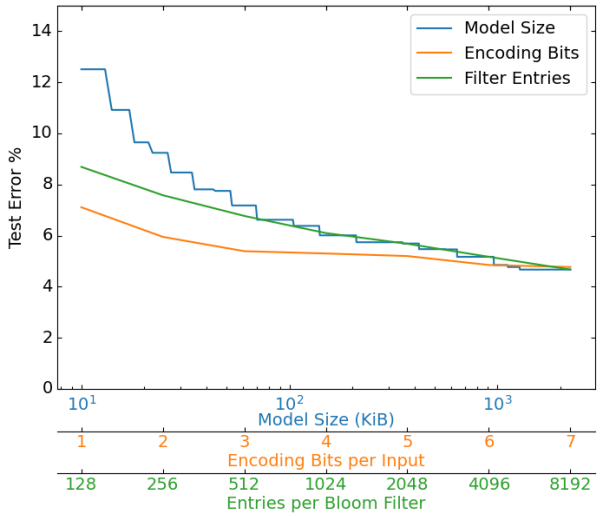


图 14. 单次训练扫描实验的结果，展示了在给定最大模型大小、给定编码位数以及给定每个布隆过滤器条目数的情况下所能达到的最高准确率。

2) 单次模型扫描：图14总结了使用一次性训练规则对各种模型配置进行扫描的结果。我们观察到，增加每个布隆过滤器的温度计编码位数和条目数带来的收益逐渐递减（请注意，图中每个过滤器的条目数采用对数刻度）。

使用单次训练技术训练的模型，其准确率大致随模型大小呈对数增长。模型大小小于 2 MB 时，我们无法达到 96% 的准确率。这表明了单次训练算法的局限性，也凸显了引入反馈的重要性；使用多次学习规则，我们能够以小得多的模型达到更高的准确率。

VI. 相关工作

由于 ULEEN 具有较高的能效，我们关于权重神经网络（WNNs）的工作适用于功耗预算极低的边缘推理应用。在本节中，我们将讨论其他针对类似应用场景的研究工作。

量化的DNN： 量化涉及将较高精度的浮点数转换为较低精度的浮点数、定点数或整数 [51] 值。量化网络可降低内存访问成本并提高计算效率。DNN 加速器通常为较低精度提供硬件支持，例如 Google TPU中的8 位整数 [52] 或 Microsoft 中的 12 位浮点数。Brainwave [53]。Bit Fusion [16]，提出了一种加速器架构，可动态组合低位宽计算单元，以匹配 DNN 的精度要求；我们将其作为比较对象。

三值权重网络（TWNs） [39], [54], [55] 已被提出，其中权重被限制为 +1、0和-1。研究表明，TWNs 的准确率仅略低于全精度网络，但显著高于二值神经网络。我们将 ULN-L 模型的准确率和效率与运行三值化LENET-5 的 Bit Fusion加速器进行了比较。 [56], 还介绍了更多用于 MNIST 的三值神经网络专用集成电路和现场可编程门阵列实现，但与我们的 ULEEN 模型相比，这些实现的准确率、吞吐量和能耗效率更低。

二值神经网络： 许多将DNN中的权重或激活函数二值化的方法已被提出 [6], [7], [9], [10], [57]。大多数现代二值神经网络方法首先对输入与学习得到的权重向量执行XNOR运算，然后进行popcount操作。已有多项研究探索了用于二值神经网络的专用集成电路和现场可编程门阵列加速器（例如，FINN [8], FP-BNN [58], 和Yodann [59]）。如与FINN的比较所示，ULEEN权重神经网络能够实现比二值神经网络更低的延迟和能耗。ReBNet[60] 是另一种基于现场可编程门阵列、用于MNIST的二值神经网络加速器。我们的吞吐量和准确率均高于ReBNet，但论文中未提供能耗数据。

压缩 DNN： 为了在硬件资源有限的设备上高效运行 DNN，采用模型压缩 [31], [61], [62], [63]。剪枝 [30], [33], [64]可减少参数总数，但可能需要重新训练以避免精度下降。CNN 中的权重稀疏性已被用于降低计算复杂度 [65], [66], [67], [68],，并在 [69], [70]中利用了比特稀疏性。我们在ULEEN 中采用两种压缩形式：使用布隆过滤器减小 RAM 节点的大小，并通过剪枝减少 RAM 节点的数量。

先前的权重神经网络： 先前的工作 [23], [71] 表明，将 WiSARD 中的 RAM 节点替换为布隆过滤器可提高内存效率和模型鲁棒性，不过我们是首个使用计数布隆过滤器（这有助于实现漂白）的研究。我们也是首个在硬件中实现基于布隆过滤器的权重神经网络的研究，因为先前的工作使用了一种不切实际的哈希算法。在 [72], 中，研究人员使用三值系统构建了一个基于 WiSARD 的石头剪刀布游戏程序，其中每个输入数字的取值都与石头、布或剪刀 状态唯一对应。先前的内存高效权重神经网络 [14], [23] 报告称，其在 MNIST 上的准确率最高可达 91.5%，而我们的改进将准确率提升至 98.46%。

符号主义方法： 符号主义范式（如决策树和贝叶斯推理）通常适用于小规模问题。然而，这些模型通常需要复杂且繁重、依赖领域的预处理，或者需要大规模（例如，半百万个查找表 [73]）或高度定制的电路实现（例如，无时钟概率 p 位 [74]）。

其他边缘 DNN 加速方法： 基于微控制器的边缘推理方法（如 TinyML）近期备受关注，因为

它们能够使用价格低廉的现成硬件 [75]。然而，这些机器学习方法的速度比专用加速器慢数千倍。使用缩小至 8x8 的图像，在 Arduino Nano 上实现的 TinyML MNIST 方案，其吞吐量比我们在 28x28 图像上对 ULN-M 的现场可编程门阵列实现低约 130,000 倍，延迟则高出 32,000 多倍，而且准确率也更低 [75]。

VII. 结论

极端边缘设备上的推理需要机器学习的新方法。现有技术使用量化 DNN 或二值神经网络，而我们提出了 ULEEN，一种基于无权重神经网络的方法。ULEEN 引入并融合了计数布隆过滤器、高斯分布温度计编码、加性集成、剪枝以及一种新颖的多次训练算法，相较于量化 DNN 和二值神经网络，实现了更优的能耗和延迟。我们展示了 ULEEN 推理加速器的现场可编程门阵列和专用集成电路实现。与基于现场可编程门阵列的二值神经网络平台 FINN 相比，我们将延迟提升了 1.4-2.6 倍，将吞吐量提升了 1.2-2.6 倍，并将能耗降低了 6.8-8.5 倍。与 Bit Fusion 低精度 DNN 专用集成电路相比，我们将能耗降低了 479-663 倍，将吞吐量提升了 2014-19549 倍，但准确率略低。准确率最高的 FINN 模型每次推理的能耗为 5.637 μJ，而对于所展示的最大模型，ULEEN 每次推理仅消耗 0.826μJ。对于 Bit Fusion 模型，每次推理的能耗范围为 93.5-129μJ，而 ULEEN 模型仅消耗 0.01-0.2 μJ（即提升超过 400 倍），展示了 ULEEN 在极端边缘设备上的潜力。

我们认为，未来工作中最直接的方向是开发卷积权重神经网络（WNNs）。卷积对于在 ImageNet 等大型图像数据集上取得良好的准确率至关重要。尽管支持卷积会增加模型和硬件的复杂性，但我们相信，权重神经网络（WNNs）有望在这一领域表现出色，因为它们能够学习非线性滤波器，而传统卷积神经网络（CNNs）无法实现这一点。

我们的结果表明，权重神经网络在边缘推理方面潜力巨大。权重神经网络效率高，但历来在准确率方面落后于二值神经网络和量化深度神经网络；ULEEN 的改进缩小或弥合了这一差距，证明了权重神经网络适用于超低能耗、高吞吐量推理。

致谢： 本研究部分得到了半导体研究公司（*SRC*）项目 *3015.001/3016.001* 以及美国国家科学基金会（*NSF*）资助项目（编号：*1763848*）的支持。本文中的任何观点、研究发现、结论或建议均代表作者个人立场，与资助机构无关。

参考文献

[1] O. Russakovsky、J. Deng、H. Su、J. Krause、S. Satheesh、S. Ma、Z. Huang、A. Karpathy、A. Khosla、M. Bernstein、A. C. Berg 和 L. Fei-Fei，“Imagenet 大规模视觉识别挑战”，国际计算机视觉杂志，第 115 卷，第 211–252 页，2015 年 4 月。[2] A. Gholami、S. Kim、Z. Dong、Z. Yao、M. W. Mahoney 和 K. Keutzer，“高效神经网络推理的量化方法综述”，2021 年。

[3] H. Cai、C. Gan、T. Wang、Z. Zhang 和 S. Han，“Once-for-All：训练一个网络并针对高效部署进行专门化”，2019 年。 [在线]。网址：https://arxiv.org/abs/ 1908 09791 。E. Covi、

[4] E. Donati、X. Liang、D. Kappel、H. Heidari、M. Payvand 和 W. Wang，“面向可穿戴设备的自适应极端边缘计算”，《神经科学前沿》，第 15 卷，2021 年。 [在线]。网址：https://www.frontiersin.org/article/ 10 3389 2021 611300 。/fnins. . J.

[5] Portilla、G. Mujica、J.-S. Lee 和 T. Riesgo，“物联网底层的极端边缘：综述”，《IEEE 传感器期刊》，第 19 卷，第 9 期，第 3179–3190 页，2019 年。[6] M. Courbariaux、Y. Bengio 和 J.-P. David，“BinaryConnect：在传播过程中使用二值权重训练深度神经网络”，载于《第 28 届神经信息处理系统国际会议论文集》第 2 卷，NIPS’15 系列。美国马萨诸塞州剑桥：MIT 出版社，2015 年，第 3123–3131 页。[7] “二值化神经网络”，CoRR，第 abs/1602.02505 卷，2016 年，已撤稿。 [ ] 1602 02505 在线。网址：

[8] http://arxiv.org/abs/。Y. Umuroglu、N. J. Fraser、G. Gambardella、M. Blott、P. Leong、M. Jahre 和 K. Vissers，“FINN：用于快速、可扩展二值化神经网络推理的框架”，载于《2017 年 ACM/SIGDA 现场可编程门阵列国际研讨会论文集》，FPGA ’17 系列。美国纽约州纽约：计算机协会，2017 年，第 65 74 [ ] 10 1145 3020078 3021744 – 页。在

[9] 线。网址：https://doi.org/。/。M. Rastegari、V. Ordonez、J. Redmon 和 A. Farhadi，“XNOR-Net：使用二值卷积神经网络进行 ImageNet 分类”，CoRR，第 abs/1603.05279 卷，2016 年。 [在线]。网址：http://arxiv.org/abs/ 1603 05279 。M. Courbariaux、I. Hubara、D. Soudry、

[10] R. El-Yaniv 和 Y. Bengio，“二值化神经网络：将深度神经网络的权重和激活值限制为 +1 或 -1 进行训练”，2016 年。[11] I. Aleksander、M. De Gregorio、F. Franc.a、P. Lima 和 H. Morton，“无权重神经系统简介”，载于第 17 届欧洲人工神经网络研讨会（ESANN），2009 年 4 月，第 299–305 页。[12] T. Ludermir、A. de Carvalho、A. Braga 和 M. Souto，“无权重神经模型：当前与既往研究综述”，《神经计算调查》，第 2 卷，第 41–61 页，1999 年 1 月。[13] P. Yin、J. Lyu、S. Zhang、S. J. Osher、Y. Qi 和 J. Xin，“理解训练激活量化神经网络中的直通估计器”，CoRR，第 abs/1903.05662 卷，2019 年。 [在线]。网址：http://arxiv.org/abs/ 1903 05662 。B. P. A.

[14] Grieco、P. M. V. Lima、M. De Gregorio 和 F. M. G. Franc.a，“从‘心理’图像生成模式示例”，《神经计算》，第 73 卷，第 7–9 期，第 1057–1064 页，2010 年 3 月。 [在线]。网址：http://dx.doi.org/ 10 1016 2009 11 015 。/j.neucom. . . D. Carvalho、

[15] H. Carneiro、F. Franc.a 和 P. Lima，“B-bleaching：WISARD 无权重神经分类器中的敏捷过度训练规避方法”，载于 ESANN，2013 年 4 月。[16] H. Sharma、J. Park、N. Suda、L. Lai、B. Chau、V. Chandra 和 H. Esmaeilzadeh，“Bit Fusion：用于加速深度神经网络的位级动态可组合架构”，载于第 45 届 ACM/IEEE 年度国际计算机体系结构研讨会，ISCA 2018，美国加利福尼亚州洛杉矶，2018 年 6 月 1–6 日，2018 年。[17] R. Rohwer 和 M. Morciniec，“n元组分类器的理论与实验现状”，《神经网络》，第 11 卷，第 1 期，第 1–14 页，1998 年 1 月。[18] I. Wickert 和 F. Franc.a，“验证无监督无权重感知器”，载于《第 9 届神经信息处理国际会议论文集》，2002 年。ICONIP ’02，2002 年 12 月，第 537–541 页，第 2 卷。[19] D. O. Cardoso、D. Carvalho、D. S. F. Alves、D. F. P. de Souza、H. C. C. Carneiro、C. E. Pedreira、P. M. V. Lima 和 F. M. G. Franc.a，“通过聚类无权重神经分类器进行金融信用分析”，《神经计算》，第 183 卷，第 70–78 页，2016 年。[20] I. Aleksander、W. Thomas 和 P. Bowden，“WISARD：图像识别领域的重大突破”，《传感器评论》，第 4 卷，第 3 期，第 120–124 页，1984年。 [在线]。网址：https://www.emerald.com/insight/ content/doi/10.1108/eb007637/full/html[21] V. Vapnik 和 A. Chervonenkis，《事件相对频率一致收敛至其概率的研究》。Springer，2015 年 1 月，第 11–30 页。[22] H. Carneiro、C. Pedreira、F. Franc.a 和 P. Lima，“WISARD n元组分类器的精确 VC 维数”，《神经计算》，第 1–32 页，2018 年 11 月。[23] L. S. de Ara´ujo、L. D. Verona、F. M. Rangel、F. F. de Faria、D. S. Menasch´e、W. Caarls、M. Breternitz、S. Kundu、P. M. V. Lima 和

F. M. G. Franca，《使用布隆过滤器的高效内存无权神经网络》，发表于ESANN，2019 年。[24] J. Carter 和 M. N. Wegman，《哈希函数的通用类》，《计算机与系统科学杂志》，第 18，2，143 154，卷，第 页，–。1979 [ ]
在线资源。可访问：https://www.sciencedirect.com/science/article/ pii/0022000079900448[25] A. Appleby，《Murmurhash3》，https://github.com/aappleby/smhasher，2016 年。[26] A. Kappaun、K. Camargo、F. Rangel、F. Firmino、P. M. V. Lima 和 J. Oliveira，《评估 WiSARD 的二进制编码技术》，发表于 2016 年第 5 届巴西智能系统会议（BRACIS），2016 年，第 103–108 页。[27] P. Xavier、M. D. Gregorio、F. M. G. Franca 和 P. M. V. Lima，《使用 WiSARD n元组分类器检测基本粒子》，发表于 ESANN，2020 年。[28] T. G. Dietterich，《机器学习中的集成方法》，载于《多分类器系统》。柏林、海德堡：Springer Berlin Heidelberg，2000 年，第 1–15 页。[29] L. A. Lusquino Filho、L. F. Oliveira、A. L. Filho、G. P. Guarisa、L. M. Felix、P. M. Lima 和 F. M. Franca，《扩展无权 WiSARD 分类器以实现回归》，《神经计算》，第 416 卷，第 280–291 页，2020 年。[在线资源]。可访问：https://www.sciencedirect.com/science/article/pii/S092523122030504X[30] S. Han、H. Mao 和 W. Dally，《深度压缩：通过剪枝、训练量化和霍夫曼编码压缩深度神经网络》，发表于国际学习表征会议（ICLR），2016 年 10 月。[31] J. Yu、A. Lukefahr、D. Palframan、G. Dasika、R. Das 和 S. Mahlke，《 Scalpel：根据底层硬件并行性定制 DNN 剪枝》，发表于 2017 年第 44 届 ACM/IEEE 国际计算机体系结构研讨会（ISCA），2017 年，第 548–560 页。[32] T. Zhang、S. Ye、K. Zhang、J. Tang、W. Wen、M. Fardad 和 Y. Wang，《一种采用交替方向乘子法的系统化 DNN 权重剪枝框架》，载于《计算机视觉——ECCV 2018》，V. Ferrari、M. Hebert、C. Sminchisescu 和 Y. Weiss 编。Cham：Springer International Publishing，2018 年，第 191–207 页。[33] W. Niu、X. Ma、S. Lin、S. Wang、X. Qian、X. Lin、Y. Wang 和 B. Ren，《 PatDNN：通过基于模式的权重剪枝在移动设备上实现实时 DNN 执行》。美国纽约州纽约市：计算机协会，2020 年，第 907–922 页。 [在线资源]。可访问：https://doi.org/10 1145 3373376 3378534 。/。[34] D. Kingma 和 J. Ba，《Adam：一种随机优化方法》，国际学习表征会议，2014 年 12 月。[35] N. Srivastava、G. Hinton、A. Krizhevsky、I. Sutskever 和 R. Salakhutdi- nov，《Dropout：一种防止神经网络过拟合的简单方法》，《机器学习研究杂志》，第 15 卷，第 1 期，第 1929–1958 页，2014 年 1 月。[36] Y. LeCun 和 C. Cortes，《MNIST手写数字数据库》，http://yann.lecun.com/exdb/mnist/，2010 年。 [在线资源]。可访问：http: // yann.lecun.com/exdb/mnist/ [37] Xilinx。（2021 年）FINN：用于在 FPGA 上进行 QNN 推理的数据流编译器。[在线资源]。可访问：https://xilinx.github.io/finn/ [38] G. I. o. T. ACT实验室。（2018 年）Bit Fusion：用于加速 DNN 的位级动态可组合架构。 [在线资源]。可访问：http://www.act-lab.org/artifacts/bitfusion/ [39] F. Li 和 B. Liu，《三值权重网络》，CoRR，第 abs/1605 04711，卷。在线资源。可访问：http://arxiv.org/abs/。2016 [ ] 1605 04711
[40] 2018 45 [ ] NCSU。（）FreePDK。在线资源。可访问：https://www.eda.ncsu.edu/wiki/FreePDK45:Contents[41]H. Packard。（2017 年）Cacti。 [在线资源]。可访问：https://github.com/ HewlettPackard/cacti/[42] M. Bayer，《 Python 的 Mako模板》，2021 年。 [在线资源]。可访问：https://www.makotemplates.org/ [43] J. Catlett，《Statlog（航天飞机）数据集》。 [在线资源]。可访问：https: //archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle) [44] K. Nakai，《Ecoli数据集》。 [在线资源]。可访问：https://archive.ics.uci. edu/ml/datasets/ecoli[45] R. Fisher，《Iris数据集》。 [在线资源]。可访问：https://archive.ics.uci.edu/ ml/datasets/iris[46] D. J. Slate，《字母识别数据集》。 [在线资源]。可访问：https://archive.ics.uci.edu/ml/datasets/letter+recognition [47] A. Sri nivasan，《Statlog（Landsat卫星）数据集》。 [在线资源]。可访问：https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)
[48] P. Mowforth 和 B. Shepherd，“Statlog（车辆轮廓）数据集。” [在线]。获取地址：https://archive.ics.uci.edu/ml/datasets/Statlog+（车辆+轮廓）[49] D. Deterding、M. Niranjan 和 T. Robinson，“连接主义基准（元音识别— Deterding 数据）数据集。” [在线]。获取地址：https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+ （元音+识别+-+Deterding+数据）[50] S. Aeberhard，“葡萄酒数据集。” [在线]。获取地址：https://archive.ics. uci.edu/ml/datasets/wine[51] B. Jacob、S. Kligys、B. Chen、M. Zhu、M. Tang、A. Howard、H. Adam 和 D. Kalenichenko，“用于高效仅整数算术推理的神经网络量化与训练”，2017 年。 [在线]。获取地址：https://arxiv.org/abs/ 1712 05877 . N. P. Jouppi、C. Young、N. Patil、D. Patterson、G. Agrawal、R. Bajwa、S. Bates、S. Bhatia、N. Boden、A. Borchers、R. Boyle、P. luc Cantin、C. Chao、C. Clark、J. Coriell、M. Daley、M. Dau、J. Dean、B. Gelb、T. V. Ghaemmaghami、R. Gottipati、W. Gulland、R. Hagmann、C. R. Ho、D. Hogberg、J. Hu、R. Hundt、D. Hurt、J. Ibarz、A. Jaffey、A. Jaworski、A. Kaplan、H. Khaitan、A. Koch、N. Kumar、S. Lacy、J. Laudon、J. Law、D. Le、C. Leary、Z. Liu、K. Lucke、A. Lundin、G. MacKean、A. Maggiore、M. Mahony、K. Miller、R. Nagarajan、R. Narayanaswami、R. Ni、K. Nix、T. Norrie、M. Omernick、N. Penukonda、A. Phelps 和 J. Ross，“张量处理单元在数据中心中的性能分析”，2017 年。 [在线]。获取地址：https://arxiv.org/pdf/1704.04760.pdf [53] J. Fowers、K. Ovtcharov、M. Papamichael、T. Massengill、M. Liu、D. Lo、S. Alkalay、M. Haselman、L. Adams、M. Ghandi、S. Heil、P. Patel、A. Sapek、G. Weisz、L. Woods、S. Lanka、S. K. Reinhardt、A. M. Caulfield、E. S. Chung 和 D. Burger，“面向实时人工智能的可配置云规模深度神经网络处理器”，载于 2018 年 ACM/IEEE 第 45 届年度国际计算机体系结构研讨会（ISCA），2018 年，第 1–14 页。[54] C. Zhu、S. Han、H. Mao 和 W. J. Dally，“训练式三值量化”，载于第五届国际学习表征会议（ICLR 2017），法国土伦，2017 年 4 月 24–26 日，会议论文集。OpenReview.net，2017 年。 [在线]。获取地址：https: // openreview.net/forum?id=S1 pAu9xl [55] Z. Lin、M. Courbariaux、R. Memisevic 和 Y. Bengio，“少乘法神经网络”，CoRR，第 abs/1510.03009 卷，2016 年。[56] H. Alemdar、V. Leroy、A. Prost-Boucle 和 F. P´etrot，“面向资源高效型人工智能应用的三值神经网络”，载于 2017 年国际神经网络联合会议（IJCNN），2017 年，第 2547–2554 页。[57] J. Fromm、M. Cowan、M. Philipose、L. Ceze 和 S. Patel，“Riptide：快速端到端二值化神经网络”，载于《机器学习与系统论文集》，I. Dhillon、D. Papailiopoulos 和 V. Sze 编，第 2 卷，2020 年，第 379–389 页。 [在线]。获取地址：https://proceedings.mlsys.org/paper/2020/file/2a79ea27c279e471f4d180b08d62b00a-Paper.pdf [58] S. Liang、S. Yin、L. Liu、W. Luk 和 S. Wei，“FP-BNN：基于现场可编程门阵列的二值化神经网络”，《神经计算》，第 275，1072 1086，卷，第 – 页。2018 [ ] 在线。获取地址：https://www.sciencedirect.com/science/article/ pii/S0925231217315035[59] R. Andri、L. Cavigelli、D. Rossi 和 L. Benini，“Yodann：一种用于超低功耗二值权重卷积神经网络加速的架构”，《IEEE 集成电路与系统计算机辅助设计汇刊》，第 37 卷，第 1 期，第 48–60 页，2018 年。[60] M. Ghasemzadeh、M. Samragh 和 F. Koushanfar，“ReBNet：残差二值化神经网络”，载于 2018 年 IEEE 第 26 届年度现场可编程定制计算机国际研讨会（FCCM），2018 年，第 57–64 页。[61] J. H. Ko、D. Kim、T. Na、J. Kung 和 S. Mukhopadhyay，“面向内存高效神经网络的自适应权重压缩”，载于《欧洲设计、自动化与测试会议》（DATE），2017 年，第 199–204 页。[62] C. Ding、S. Liao、Y. Wang、Z. Li、N. Liu、Y. Zhuo、C. Wang、X. Qian、Y. Bai、G. Yuan、X. Ma、Y. Zhang、J. Tang、Q. Qiu、X. Lin 和 B. Yuan，“CirCNN：利用分块循环权重矩阵加速和压缩深度神经网络”，载于 2017 年 IEEE/ACM 第 50 届年度国际微体系结构研讨会（MICRO），2017 年，第 395–408 页。[63] S. Liu、Y. Lin、Z. Zhou、K. Nan、H. Liu 和 J. Du，“面向移动设备的按需深度模型压缩：一种使用驱动的模型选择框架”，载于第 16 届年度移动系统、应用与服务国际会议论文集，MobiSys ’18 系列。美国纽约州纽约：计算机协会，2018 年，第 389–400 页。 [在线]。获取地址：https://doi.org/10.1145/3210240.3210337

[64] F. N. Iandola、M. W. Moskewicz、K. Ashraf、S. Han、W. J. Dally 和 K. Keutzer，“Squeezenet：以少 50 倍的参数实现与 Alexnet 同等的准确率，且模型大小小于 1mb”，*ArXiv*，第 abs/1602.07360 卷，2016 年。[65] J. Albericio、P. Judd、T. Hetherington、T. Aamodt、N. E. Jerger 和 A. Moshovos，“Cnvlutin：无无效神经元的深度神经网络计算”，载于 *2016* 年 *ACM/IEEE* 第 *43* 届年度国际计算机体系结构研讨会 *(ISCA)*，2016 年，第 1–13 页。[66] A. Parashar、M. Rhu、A. Mukkara、A. Puglielli、R. Venkatesan、B. Khailany、J. Emer、S. W. Keckler 和 W. J. Dally，“Scnn：一种用于压缩稀疏卷积神经网络的加速器”，载于*2017* 年 *ACM/IEEE* 第 *44* 届年度国际计算机体系结构研讨会 *(ISCA)*，2017 年，第 27–40 页。[67] S. Zhang、Z. Du、L. Zhang、H. Lan、S. Liu、L. Li、Q. Guo、T. Chen 和 Y. Chen，“Cambricon-x：一种用于稀疏神经网络的加速器”，载于 *2016* 年第 *49* 届 *IEEE/ACM* 微体系结构年度国际研讨会 *(MICRO)*，2016 年，第 1–12 页。[68] X. Zhou、Z. Du、Q. Guo、S. Liu、C. Liu、C. Wang、X. Zhou、L. Li、T. Chen 和 Y. Chen，“Cambricon-s：通过软硬件协同方法解决稀疏神经网络中的不规则性”，载于*2018* 年第 *51* 届 *IEEE/ACM* 微体系结构年度国际研讨会 *(MICRO)*，2018 年，第 15–28 页。[69] J. Albericio、A. Delm´as、P. Judd、S. Sharify、G. O’Leary、R. Genov 和 A. Moshovos，“Bit-pragmatic 深度神经网络计算”，载于 *2017* 年第 *50* 届 *IEEE/ACM* 微体系结构年度国际研讨会 *(MICRO)*，2017 年，第 382–394 页。

[70] M. Mahmoud、K. Siu 和 A. Moshovos，《Diffy：一种用于无 D´ej`a 的差分深度神经网络加速器》，载于 2018 年第 51 届 IEEE/ACM 微体系结构国际研讨会（MICRO），2018 年，第 134–147 页。[71] L. Santiago、L. Verona、F. Rangel、F. Firmino、D. S. Menasch´e、W. Caarls、M. Breternitz Jr、S. Kundu、P. M. Lima 和 F. M. Franc¸a，《作为内存分段布隆过滤器的无权重神经网络》，《神经计算》，第 416 卷，第 292–304 页，2020 年。[72] D. F. d. Souza、H. C. Carneiro、F. M. Franc¸a 和 P. M. Lima，《石头剪刀布 WiSARD》，载于 2013 年 BRICS 计算智能大会暨第 11 届巴西计算智能大会，2013 年，第 178–182 页。[73]R. Choudhury、S. R. Ahamed 和 P. Guha，《决策树训练加速器的高效硬件实现》，载于 https://ieeexplore.ieee.org/document/942607345。[74] R. Faria、J. Kaiser、K. Y. Camsari 和 S. Datta，《自主贝叶斯网络的硬件设计》，《计算神经科学前沿》，第 15 卷，2021 年。 [在线]。链接：https://www.frontiers.in.org/article/10.3389/fncom.2021.584797[75] Simone，《TinyML 或 Arduino 与 STM32：卷积神经网络（CNN）示例，访问日期：2021 年 11 月 22 日》，2020 年。 [在线]。链接：https://eloquentarduino.github.io/2020/11/tinyml-on-arduino-and-stm32-cnn-convolutional-neural-network-example/