通过架构与连接优化增强基于查找表的深度神经网络推理

Binglei Lou、Ruilin Wu、Philip Leong 悉尼大学
电气与计算机工程学院，2006年，澳大利亚 邮箱：{
binglei.lou,ruilin.wu,philip.leong}@sydney.edu.au

arXiv:2601.09773v1 [cs.AR] 14 Jan 2026

摘要—**在资源受限的边缘设备（如现场可编程门阵列）上部署深度神经网络，需要在保持高精度的同时，谨慎平衡延迟、功耗和硬件资源使用量。现有的基于查找表的深度神经网络，例如 LogicNets、PolyLUT 和 NeuraLUT，面临两项关键挑战：查找表大小呈指数级增长，以及随机稀疏连接效率低下。**

本文提出 SparseLUT，这是一种通过两种正交优化来应对上述挑战的综合框架。首先，我们提出一种架构增强方案，通过加法器聚合多个 PolyLUT子神经元，将查找表消耗量显著降低 2.0×–13.9× ，并将推理延迟降低 1.2×–1.6×，同时保持相当的准确率。在此基础上，我们进一步提出一种非贪心训练算法，通过选择性剪枝不太重要的输入，并有策略地重新生长更有效的输入，来优化神经元连接。这种训练优化不会带来额外的面积和延迟开销，并在各项基准测试中持续提升准确率：与现有的查找表深度神经网络方法相比，在 MNIST 上最高提升 2.13 %，在喷注子结构分类上提升 0.94%。

索引词—**动态稀疏性、现场可编程门阵列、神经网络、查找表**

I. 引言

深度神经网络（DNN）显著提升了我们解决各种数据格式模式识别问题的能力，这些数据格式包括图像、视频、语音、音频和文本 [1]。现场可编程门阵列（FPGA）为部署深度神经网络提供了独特的实现平台，因为它们能够更好地与网络和视频解码器等其他系统组件集成，支持将信号处理与深度神经网络相结合，并且与图形处理器相比，能耗更低、延迟更短，尤其适用于实时推理任务。

在人工神经网络中，神经元的输出由下式给出

$$y = \sigma\left(\sum_{k=1}^{N} w_k x_k + b\right) \tag{1}$$

其中， N 是输入的数量， x_i 是输入， w_i 是权重， b 是偏置项， $\sigma(\cdot)$是激活函数。基于查找表（LUT）的深度神经网络（LUT-DNN）利用现场可编程门阵列原生组件，实现卓越的面积效率和低延迟。这是通过将图 1 （公式 1 中的乘法、求和和激活运算合并到单个查找表中）实现的。

近期方法，包括 LogicNets [2],PolyLUT [3],和 NeuraLUT [4], 引入了 先验 权重稀疏性，通过限制每个神经元的扇入， F。换言之，它们将权重向量 $W = \{w_1, w_2, \ldots, w_N\} \in \mathbb{R}^N$限制为先验确定、稀疏，且最多包含 F个非零值（$0 < F \ll N$）。

现有基于LUT的深度神经网络的准确率从根本上受到可扩展性有限和随机稀疏性欠佳的制约。为了使真值表大小保持在可控范围内，输入位宽 β 和扇入 F 都必须较小，这显著限制了网络的表征能力。此外，普遍采用随机分配的稀疏连接，往往会导致输入选择效率低下，进一步影响准确率。

尽管以往的架构增强尝试通过提升单个神经元的表达能力来提高准确率，例如用多项式函数 [3] 或子网络结构 [4]，取代线性运算，但这些方法仍受制于一个根本限制：LUT 大小会随 βF呈指数级增长，因此无法解决可扩展性瓶颈。与此同时，现有的稀疏性优化技术，包括训练后剪枝 [5–8]和训练期间稀疏化方法 [9–12], ，都是为传统深度神经网络设计的。它们通常在模型或层级别进行操作，无法应对基于查找表的网络特有的神经元级扇入约束。这些局限凸显了一个亟待填补的关键空白，也带来了创新机遇。

本文介绍了 SparseLUT，这是一种通过架构和训练算法层面优化来应对上述独特挑战的架构。图 2 分别展示了这两种优化。

首先，在架构层面，我们通过一个 A 输入加法器组合 A 个 PolyLUT子神经元，以增加神经元

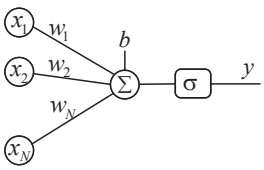


图 1：扇入为 N 的神经元计算。查找表深度神经网络通常选择输入的一个随机子集（大小为 $F \ll N$）。SparseLUT 是一种在最大化准确率的同时选择输入的训练方案。

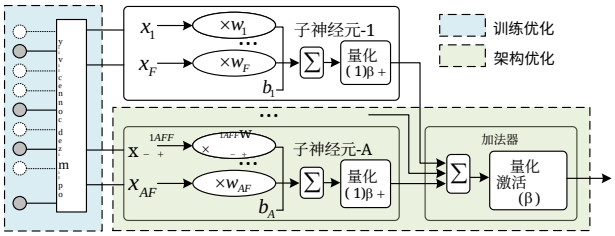


图 2：SparseLUT 的优化策略

扇入。如图2右侧所示，我们的方法重新构建了神经元计算：将权重乘法、累加、批归一化（BN）和量化激活分为两个阶段。第一阶段与 PolyLUT 类似，但不包含批归一化，并针对每个子神经元重复执行。批归一化则在结果累加后执行，并在必要时再次量化所得激活。这样，我们可以更充分地利用现场可编程门阵列逻辑资源，其中查找表成本 $\mathcal{O}(2^{\beta F A})$ 降低至 $\mathcal{O}(A \times 2^{\beta F} + 2^{A(\beta+1)})$。

其次，我们通过仔细选择非零值，提出了一种非贪心稀疏化策略 w_k，如公式 1 所示。具体而言，图3将在第 III 节中详细介绍，展示了我们的训练方法：在训练过程中逐步调整活跃连接，最终达到目标扇入$F = 2$。随着训练的进行，为活跃连接少于 F 的神经元添加新连接（情况 1）；而活跃连接超过 F 的神经元则停用连接（情况 2）。在整个训练过程中，所有神经元都将达到目标扇入。

本工作的主要贡献如下：

1）在计算机体系结构层面，我们提出了对 Poly- LUT 框架进行高效重构与扩展的方法 [3],该框架集成了 A 通过一个 A输入加法器组合的 PolyLUT 子神经元，从而提升了可扩展性和准确率。2）在工具层面，我们提出了首个用于基于查找表的神经网络的非贪心训练算法，在训练过程中优化神经元输入连接性。该方法不会产生额外的查找表或布线资源开销，因为它仅修改了 F 图1中所示输入的选择策略。3）这两种优化彼此正交。在评估中，我们首先将所提出的架构优化应用于原始 PolyLUT。在准确率相当的情况下，我们的方法将 MNIST 和喷注子结构分8×类任务中的查找表消耗量分别降低了 49× 和 132×，同时将延迟分别降低了 16×和 1 [2],。此外，与 LogicNets、 [3],PolyLUT [4], 和 NeuraLUT 等基线 LUT-DNN 模型相比，我们提出的连接性优化在不产生任何硬件或延迟开销的情况下，持续提升了准确率。与随机稀疏性相比，该方法在 MNIST 上的准确率最高提升 2.13%，在喷注子结构分类任务上的准确率提升 0.94%。

本文是对早期一篇会议论文的扩展 [13]；主要区别在于工具层面的非贪心训练算法（第 III-B节）。本文其余部分的组织结构如下：第 II 节回顾相关工作，包括基于LUT的深度神经网络和深度稀疏化技术。第III 节详细介绍所提出的方法，包括架构层面的优化和非贪心稀疏化策略。第 IV节展示实验结果，说明与现有方法相比在准确率和资源利用率方面的提升。最后，第 V 节总结全文并概述未来的研究方向。

II. 背景

现场可编程门阵列上的机器学习推理可通过多种方法实现。最直接的方法是基于 DSP 的推理，例如 hls4ml [14],，它使开发者能够使用 TensorFlow 或 PyTorch 等流行框架在 Python 中训练模型，并自动生成可综合的高层次综合代码。这种方法充分利用现场可编程门阵列的各类资源，包括 DSP 模块。另一类方法是基于 XNOR 的推理，例如二值神经网络 [15], ，它采用 1 位量化，以高效的 XNOR 运算取代乘法器。纯基于查找表的方法还可进一步分为基于树的方法、基于可微查找表的方法和传统的基于查找表的深度神经网络方法。例如，TreeLUT [16] 仅使用查找表实现梯度提升决策树，而不依赖块随机存取存储器或 DSP；可微无权重神经网络（DWN） [17] 则完全由映射到查找表的无权重节点构成神经网络。本工作主要针对传统的基于深度神经网络的查找表设计，并在评估部分与其他类别的方法进行比较。

A. 基于查找表的传统深度神经网络方法

LogicNets [2] 是首个基于查找表的深度神经网络架构，它对每个神经元的输入和输出进行量化，并将神经元的传递函数（即，密集连接的线性层和激活函数）封装在查找表中。该方法枚举神经元输入的所有可能组合，并根据神经元的权重和偏置确定相应的输出。通过使用布尔表达式替代人口计数运算，大幅降低了计算开销。在 LogicNets 的基础上，Andronic等人提出的 Poly-LUT [3], 通过引入分段多项式函数，进一步提高了准确率，并减少了所需的层数。随后，为继续提升每个查找表的表示能力，NeuraLUT [4] 采用网络中网络技术 [18]，从而将整个子网络映射到单个查找表中，实现更深的神经网络和更高的准确率。

图 4展示了这些基于查找表的深度神经网络方法背后的核心思想。如顶部所示，最多从 F 个输入中随机选择 N 个当前层节点，连接到下一层的每个神经元（此处仅展示了下一层的一个神经元）。此外，

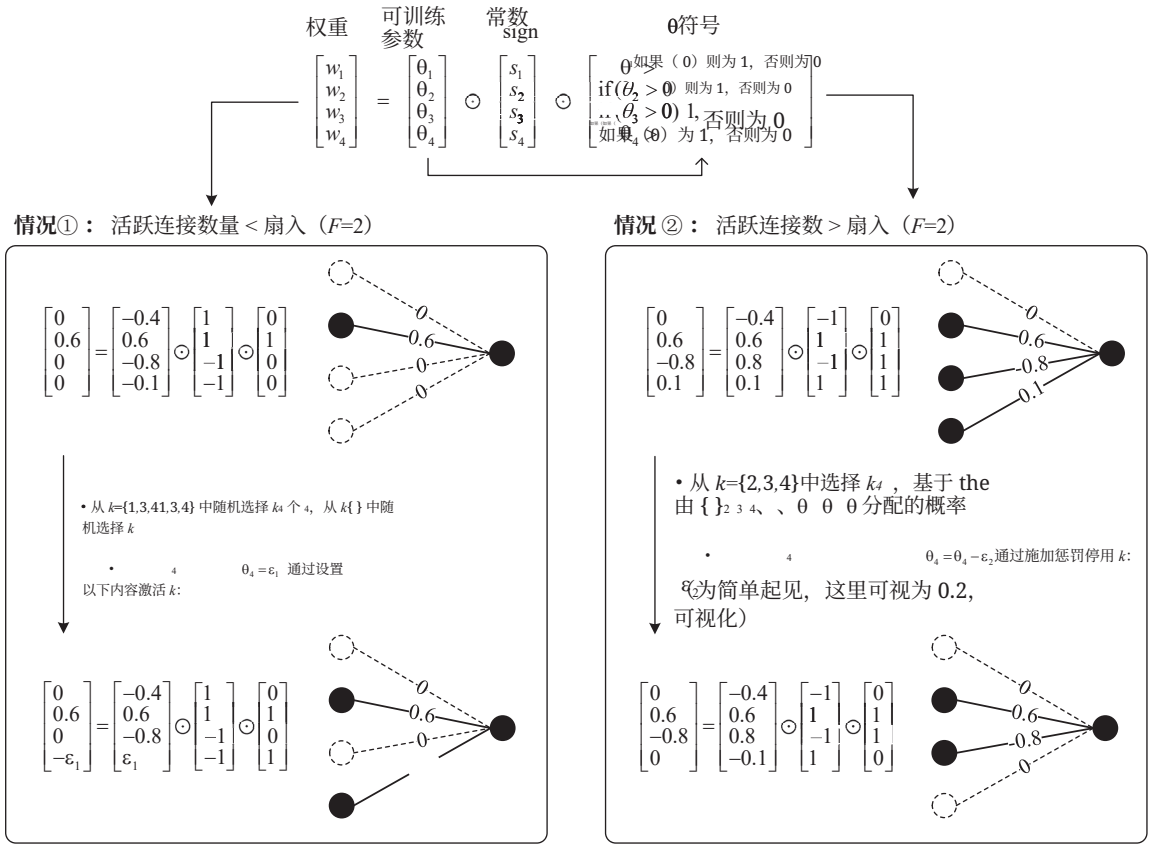


图 3：示例迭代，展示 SparseLUT 如何调整神经元连接，使其达到目标扇入 2：引入新连接，将扇入从 1 增至 2；移除连接，将扇入从 3 降至 2。请注意， ϵ_2 参数通常设为小得多的值，因此需要多次迭代才能停用一个连接。

表 I：动态稀疏化方法汇总。

方法	训练	丢弃（剪枝） ↘	再生长 ↗	先验固定扇入约束
GMP [9]	稠密到稀疏	幅值	✗	✗
DeepR [10]	稀疏到稀疏	随机	随机	✗
RigL [11]	稀疏到稀疏	幅值	梯度	✗
SRigL [12]	稀疏到稀疏	幅值	梯度	✗
DWN [17]	稀疏到稀疏	幅值	✗	✓*
SparseLUT（我们的方法）	灵活	随机+幅值	随机	✓

* DWN 是一种特殊情况，专为无权重神经网络设计，不适用于通用的加权深度神经网络。

每个神经元输入和输出的位宽均量化为 β，而神经元传递函数中的其余参数则保持全精度。因此，训练后，可在硬件中使用 βF 个输入实现将输入向量$[x_1, x_2, \ldots, x_F]$映射到输出节点的传递函数，因此，其实现需要$\mathcal{O}(2^{\beta F})$个查找表来构成预计算真值表。

B. 稀疏化

实际上，对于大多数稠密深度神经网络而言，相当一部分权重是冗余的，可以将其置零，而对准确率的影响极小 [8]。关于神经网络稀疏化（即网络剪枝）的研究可以追溯到20世纪80年代末以及 20 世纪 90 年代中期。这些技术从一个预训练的稠密连接模型开始，执行权重移除，即识别出不重要的权重并将其置零。人们采用了各种方法作为移除准则。例如，1989 年的早期研究“最优脑损伤” [5], 根据海森矩阵的对角元素确定并移除不显著的权重。1995 年，Thimm等 [6]证明，根据权重幅值移除权重是一种简单而有效的方法。Gale 等 [7] 进一步研究了其他剪枝准则，并将其应用于在 WMT2014 英译德数据集上训练的大规模 Transformer，以及在 ImageNet 上训练的 ResNet-50。他们得出结论，正则化L_0 和变分 Dropout [19]等复杂技术 [20]能够实现

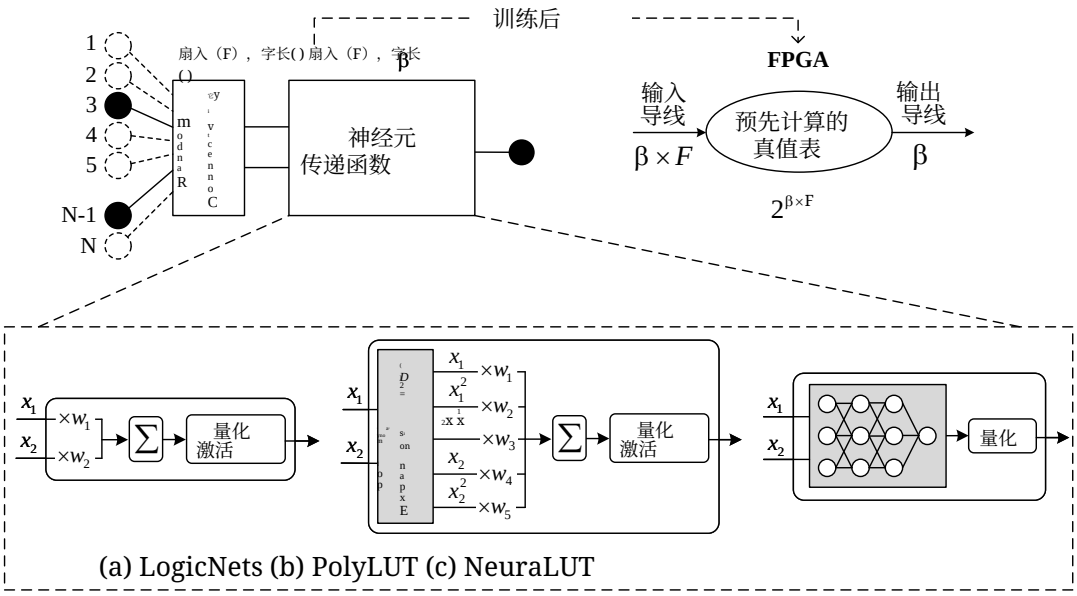


图 4：具有稀疏连接的广义查找表深度神经网络架构示意图。

与基于幅值的剪枝相比，准确率-稀疏性权衡相近。对于查找表深度神经网络，PolyLUT [21]和 NeuraLUT-Assemble [22] 提出了一种基于权重排序的训练后剪枝策略，以取代随机连接。具体而言，训练完成后，对每个神经元的权重分支按其扇入进行排序，并按重要性从高到低选择排名靠前的分支。

Gale 等人 [7] 还发现，仅通过剪枝学得的非结构化稀疏架构，其性能不如通过联合稀疏化与优化训练得到的模型。因此，较新的方法从头开始，而非从预训练模型出发，也就是说，它们（1）从原始初始训练条件开始；（2）通过同时进行权重移除和权重再生（可选）来更新连接性。这些方法又可分为两个子类，其代表方法列于表 I：（1）训练从稠密模型开始，并在训练过程中逐步移除连接，以达到所需的稀疏性 [9]。（2）训练从稀疏模型开始，以尽量减少训练内存开销；在训练过程中，根据权重幅值、梯度或随机准则移除并重新生长连接 [10–12]。

渐进式幅值剪枝 (GMP) [9],会在训练过程中，从稠密的初始解中移除连接，直至达到所需的稀疏性。它从稠密网络开始，移除幅值最小的权重。其稀疏性会单调递减，直至达到目标稀疏性。

相比之下，还有一些方法采用神经网络的动态重连 [10–12]。这一策略受到人脑的启发：大脑的大部分体积由白质构成，也就是神经元之间的连接网络。在大脑中，突触连接具有高度动态性，突触会持续形成并重组，尤其是在学习过程中 [23–26]。

深度重连（DeepR） [10]提出了一种用于动态重连的随机帧架。在初始化时，DeepR 使用连接数量有限的网络，这些连接是随机生成的。在训练过程中，每次迭代时，如果连接的权重更新导致符号改变，就会移除该连接。随后随机激活一个新连接，以维持目标稀疏性。我们的工作扩展了这种随机游走机制，更多细节将在后续章节中介绍。Rigged Lottery（RigL） [11] 会移除幅值最小的权重，并在训练过程中依据梯度幅值重新生长连接。每次迭代中，移除的连接数与重新生长的连接数相等，从而保持稀疏性水平不变。RigL 旨在处理层级的标准非结构化稀疏约束。结构化 RigL（SRigL） [12]通过引入常数扇入的结构化稀疏性扩展了 RigL，从而实现紧凑的权重矩阵表示。具体而言，训练完成后，每个矩阵中的非零权重呈矩形模式。SRigL 达到的目标扇入是平均值，而非在每个神经元层面强制实现。

DWN 的逐元素无权重映射并不直接适用于我们的基于查找表的加权推理流程。DWN [17] 是一种无权重神经网络模型，其可学习映射机制会在训练期间根据权重大小选择固定数量的连接。具体而言，其计算不涉及加权求和、累加、非线性激活或标准反向传播。相反，它提供了一种可微机制，可决定输入直接路由至查找表的方式，从而无需传统权重即可进行训练。因此，尽管 DWN 提供了一种富有启发性的在线可学习映射，但其无权重的逐元素形式与传统的基于查找表的深度神经网络推理所依赖的加权计算存在根本差异，尤其是对于采用定制架构和训练流程的网络（例如 PolyLUT 中的多项式形式以及网络中网络

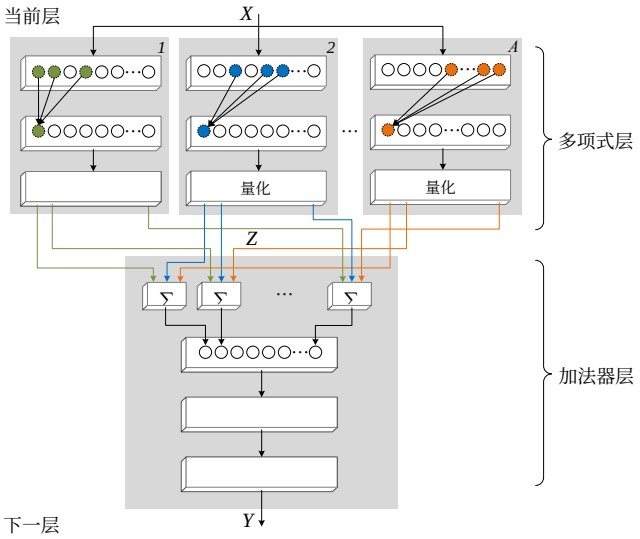


图 5：优化后架构的单层框图。

结构（在 NeuraLUT 中）。因此，现有文献尚未展示将其直接集成到传统加权深度神经网络训练流程中的方法。

III. 方法论

A. DNN 架构优化

图5展示了我们提出的深度神经网络架构。与图4相比，子神经元的扇入 F 保持不变，但神经元的总扇入在输出端增加了A 倍。这是通过对 A 个独立且并行的随机连接 Poly 层求和实现的。Poly 层与加法器层之间的连接性固定，以便对每个神经元执行加法运算。为阐明其增强机制，我们引入了式（2）中详述的公式。

$$\sum_{i=0}^{AF-1} w_i x_i + b = \sum_{a=0}^{A-1} \left(\sum_{i=0}^{F-1} w_{(aF+i)} x_{(aF+i)} + b_a \right) \tag{2}$$

在计算过程中，诸如修正线性单元（ReLU）之类的激活函数输出的位数可能比输入少一位，因为其输出为非负数。为避免加法器层发生溢出，我们将其内部字长增加一位（至 $\beta+1$），如图2所示。

B. 动态稀疏化

1）模型表示：与只专注于更新网络权重和偏置的传统深度神经网络不同，SparseLUT 还会在一组预定义约束下执行权重移除和重新生长。其得以实现的一个关键因素是权重表示。我们采用 DeepR 的方法 [10] ，其中网络中的每个连接 k 由两个部分表示：

- 一个**可训练的连接参数** $\theta_k \in \mathbb{R}$，同时控制连接的幅值及其状态（激活或未激活）。
- 一个**预定义的、不可训练的常数符号** $s_k \in \{-1, 1\}$，经过随机初始化，用于编码连接的固定极性。

参见图3，我们用 $W = \{w_1, w_2, \ldots, w_N\} \in \mathbb{R}^N$ 表示公式1中神经元的权重向量。

算法1概述了 SparseLUT 中确定每个神经元有效权重向量的过程。算法从初始权重向量 $W_0 \in \mathbb{R}^N$开始，其中每个元素的幅值用作 θ_k的初始值。这确保了初始条件下的密集连接，所有对应于 θ_k 的连接均为非负。网络的初始稀疏性由 $|W_0|$ 与二进制稀疏掩码向量*is con*的逐元素乘积决定，其中仅将元素 F_i 设为 1，其余元素均设为 0。最后，为得到有效权重向量 W，对乘积$\theta_k \odot s_k$应用指示函数 $\mathbb{1}(\cdot)$，以控制每个连接的激活状态。

算法 1： 权重映射过程

输入： W 的权重维度，初始扇入为 F_i。

1 $W_0 \leftarrow$ 从以下分布中随机初始化权重 W_0 标准正态分布，且具有相同的维度（与 W 相同）；
2 $is_con \leftarrow$ 生成一个二值稀疏掩码 $is\ con$ _ 使每个输出神经元都连接到 F_i 随机选取的输入神经元；
3 $\theta \leftarrow |W_0| \odot is_con$ （将初始 θ 值计算为逐元素乘积）；
4 $sign \in \{-1, +1\}$ 随机初始化权重符号，采用均匀概率；
5 $W \leftarrow \theta \cdot sign \cdot \mathbb{1}(\theta > 0)$ （计算最终权重，其中，$\mathbb{1}(\cdot)$ 是将负值设为 θ 的指示函数，使其变为零）；
6 **返回**最终权重向量 W

2）稀疏训练： SparseLUT 训练过程（如算法2所述）是一种非贪心算法。每次迭代都会更新连接参数，并调整网络的稀疏性，以强制执行目标扇入 F_o 训练期间的稀疏性。

在每个反向传播训练步 [27], 中，仅更新活跃连接的参数 θ_k （$\theta_k > 0$），具体依据第 6 行中的规则进行更新。给定学习率为 η，项$\eta \frac{\partial E(\theta)}{\partial \theta_k}$ 表示误差函数的导数，而$\eta\alpha$ 是正则化项（α 是任意正则化函数）。最后一项 ηv_k最初由 DeepR 引入， [10],在参数空间中实现了随机游走方案。这里， $v_k \sim N(0, G^2)$ 表示一个与 θ_k维度相同的噪声矩阵，其均值为0，标准差为 G。完成此次更新后，任何 θ_k 变为负值的连接都会被停用。

对于每个神经元，现有活跃连接数与目标扇入之间的差值计算为 R。负的 R 表示活跃连接数小于目标扇入，因此会激活 $|R|$ 个从非活跃连接中均匀随机选择的连接 k'

算法 2： SparseLUT 训练过程

输入： 目标扇入 F_o，稀疏到稠密训练起始点 T，学习率 η，正则化系数 α，以及噪声 $v_k \sim N(0, G^2)$。

输出： 特征掩码 $\mathcal{M}$

1 **初始化：**
2 $W \leftarrow$ 定义于算法1；
3 **训练：**
4 **对**每个训练步 t**执行**
5 **对** 所有活跃连接 k **(**$\theta_k > 0$**) 执行**
6 更新 $\theta_k \leftarrow \theta_k - \eta \frac{\partial E(\theta)}{\partial \theta_k} - \eta\alpha + \eta v_k$；7 **如果** $\theta_k < 0$ **则** 将连接 k 设为非活跃；
8 $R \leftarrow$ 活跃连接数量 - F_o；
9 **如果** $R < 0$ **则**
10 以 k' 选择 $|R|$非激活连接均匀概率并激活它们；
11 设置 $\theta_{k'} \leftarrow \epsilon_1$；
12 **else**
13 **如果** $t < T$ **则**
14 对所有活跃连接的 θ 值进行排名；
15 选择 $|R|$ 活跃连接 k' 中具有最低排名概率的连接，并应用惩罚；
16 更新 $\theta_{k'} \leftarrow \theta_{k'} - \epsilon_2$；
17 **else**
18 对所有活跃连接的 θ 值进行排名；
19 选择 $|R|$ 活跃连接 k' 中具有排名概率最低的，并停用它们；
20 设置 $\theta_{k'} \leftarrow 0$；
21 **返回** 特征掩码 $\mathcal{M}$，其中 $\mathcal{M}_k = 1$ 若 $\theta_k > 0$，否则 $\mathcal{M}_k = 0$；

（第 10 行）。这些连接会以较小的值 ϵ_1 初始化，使它们能够重新参与反向传播过程，而不会在重新激活后立即主导更新。

相反，正值 R 表示 $|R|$ 需要停用活跃连接以满足扇入约束。不同于表 I中的方法，SparseLUT 在此停用过程中采用两阶段训练策略：

1) 渐进式稀疏化阶段：在训练的早期阶段（$t < T$），每次迭代都会从 $\theta_{k'}$ 中减去一个较小的惩罚项 ϵ_2（第 16 行）。这一渐进式惩罚会更大程度地减小较小参数的值，其依据是权重较小的连接对神经元输出的贡献较小。连接收敛的速度会受到 ϵ_2的显著影响。当$\theta_{k'}$ 变为负值时，这些连接将被消除。具体而言，较大的 ϵ_2 值可能会导致权重相对较大的节点立即改变状态。这一阶段放宽了传统方法所采用的严格要求，即被丢弃的连接数与再生长的连接数必须始终相等，传统方法包括 DeepR 等再生长方法 [10],RigL [11],和 SRigL [12]。我们将在后文展示，训练过程中的这种更大灵活性能够持续提升准确率。

2）微调阶段：在训练的后期阶段（$t \geq T$），模型会严格执行扇入约束。当连接被直接停用时 $|R| > 0$（第 20 行）。此阶段专注于通过定期重新连接被消除的连接，并选择数量相同的替代连接，来对剩余连接进行微调，从而维持每个神经元的扇入约束。

完成训练过程后，算法最终生成的输出为特征掩码 $\mathcal{M}$，其中每个元素 $\mathcal{M}_k$ 表示连接 k的状态（第 21 行）。

如算法 2 所示，SparseLUT 的计算成本主要来自连接搜索，具体而言，是剪枝和再生长步骤。当 $R < 0$（再生长）时，会从所有非活跃候选项（$N - F \approx N$）中进行选择；而在 $R > 0$（剪枝）时，则对活跃候选项进行排名。根据经验，在较长的训练周期中，活跃候选项的数量会迅速稳定在接近目标扇入的值F_o 。因此，排序开销主要取决于 F_o。由此，训练时间主要受模型大小、训练轮数以及 F_o 的影响。

C. 工具流程

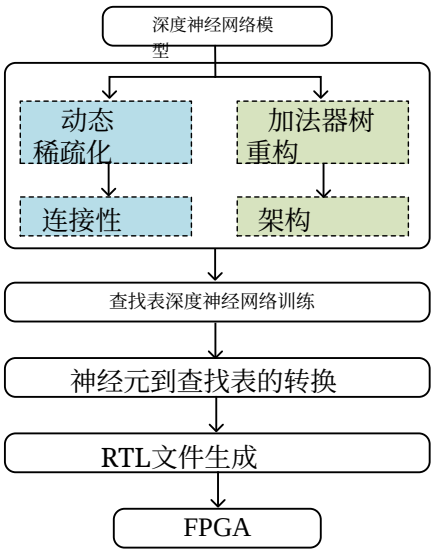


图 6：SparseLUT 的工作流程。

图6展示了所提出的 SparseLUT 框架的整体工作流程，该框架可直接应用于现有的基于查找表的深度神经网络设计 [2–4]。该流程从与标准查找表深度神经网络相同的模型配置开始，从而与其训练流程完全兼容。在流程左侧，SparseLUT 会生成一个连接性特征掩码 $\mathcal{M}$，该掩码符合现有查找表深度神经网络的结构要求，并可直接替换

图4中所示的随机生成连接。此外，网络架构经过修改，将扇入因子 A 设为可配置参数。借此可使用加法树结构重建网络拓扑。

在此之后，使用 PyTorch 离线进行查找表深度神经网络训练 [28]。通过与 Brevitas 库集成 [29]，实现量化感知训练（QAT），确保训练后的模型适用于硬件部署。训练完成后，模型权重将转换为查找表。该转换首先分析量化激活值，以确定每个神经元的输入范围。对于子神经元层，将根据参数生成所有可能的输入组合 β 和 F；对于加法器层，则使用 β 和 A生成组合。随后，将这些输入向量传播通过各自的层，以获得输出值，从而形成构成查找表内容的输入-输出对。

随后，这些查找表用于生成 Verilog 中的寄存器传输级（RTL）代码，以描述每个神经元实现的布尔函数。最后，使用 AMD/Xilinx Vivado 工具链将设计综合到硬件上 [30]。

IV. 结果

A. 数据集

我们使用两个广泛采用的数据集评估所提出的 SparseLUT 设计，用于超低延迟推理。这些数据集与现有技术的评估基准一致，包括 LogicNets [2], PolyLUT [3], 和 Neu-raLUT [4]。此外，为了评估优化后的连接性在更复杂任务上的泛化能力，我们还基于 CIFAR-10 数据集对分类准确率进行了消融研究。由于上述所有查找表深度神经网络方法均未开展任何基于 CIFAR-10 的实验，且本文重点并非探索最优超参数搜索空间，因此将 MNIST 所采用的设置直接应用于 CIFAR-10 任务。

1) 手写数字识别：在医疗保健、医学成像和实时目标跟踪等对时序要求严格的领域，低延迟至关重要。这些应用凸显了快速、准确地做出决策的必要性，即使是极短的延迟也可能造成重大影响。遗憾的是，目前没有专门针对低延迟图像分类任务的公开数据集，因此我们采用 MNIST [31]，将本研究与其他基于LUT的深度神经网络进行比较。MNIST 是用于手写数字识别任务的数据集，以 28×28 个像素作为输入，并输出 10 个类别。

2)喷注子结构分类：对于欧洲核子研究中心大型强子对撞机（大型强子对撞机）等物理实验而言，实时决策往往至关重要。喷注子结构分类（JSC）是其中一项需要高吞吐量数据处理的应用。先前研究 [32–35] 采用现场可编程门阵列上的神经网络来执行此任务，以实现实时推理。我们也使用根据参考文献 [33] 构建的 JSC数据集来评估我们的工作。该数据集以 16 个子结构属性为输入，以 5 种喷注类型为输出。在此任务中，基于现场可编程门阵列的分类必须采用流水线处理，以应对 40 MHz 的数据速率，同时确保响应延迟低于一微秒。

3) 目标识别：与手写数字识别类似，真实场景中的目标识别也要求低延迟，以便快速、可靠地做出决策。为评估我们的方法，我们采用了 CIFAR-10 [36]数据集。与 MNIST 相比，该数据集提供了更复杂的基准。CIFAR-10 包含来自 10 个不同目标类别的32×32 张彩色图像，因此分类任务更具挑战性。

我们强调，SparseLUT 中提出的架构优化与连接性训练优化彼此正交，可以独立应用。为开展消融研究，我们首先评估仅采用架构优化的模型。为与我们已发表工作中使用的术语保持一致， [13], 我们将这个仅包含架构优化的变体称为 PolyLUT-Add。

B. 架构优化评估

我们使用 AdamW 作为优化器， [28],对较小的模型／数据集（JSC-M Lite）训练 1000 轮，对（JSC-XL 和 HDR）训练 500 轮。对于（JSC-XL、JSC-M Lite）和 MNIST，迷你批次大小分别设为 1024 和 128。我们沿用 PolyLUT 中的这些配置， [3]以确保评估的一致性。

硬件评估使用 Vivado 2020.1，针对 xcvu9p-flgb2104-2-i FPGA 器件进行编译；配置 Flow_PerfOptimized_高设置，以在上下文外（OOC）模式下进行综合，并将默认值用于布局布线设置。

1) PolyLUT-Add 与原始 PolyLUT 的比较：如表II所示，对于 $A = 2$，在 MNIST 和喷注子结构分类基准测试上，PolyLUT-Add 相较于 PolyLUT 的准确率分别提高了 2.7% 、 0.6% 和 2.3%，而 LUT 大小仅增加 2–3× 。相比之下，要通过增加扇入来使 PolyLUT 获得类似的准确率提升， F会导致 LUT 大小增加 256–1024× 。这些结果表明，PolyLUT-Add 在准确率与硬件成本之间实现了更高效的权衡。此外，值得注意的是，寄存器传输级生成时间成本也与查找表条目数相关；因此，直接增加扇入会导致寄存器传输级生成时间成本呈指数级增长。（寄存器传输级生成时间是在配备 Intel(R) Core(TM) i7-10700F @2.9GHz 和 64GB 内存的台式机上测量的。）

2) PolyLUT-Add 与更深、更宽的 PolyLUT 对比：进一步测试了 PolyLUT-Add 与 PolyLUT 另外两种变体之间的准确率比较（基线 PolyLUT 设置如表III所示）：

1) **PolyLUT-Deeper**：本方法探究增加网络深度的影响。我们将深度因子记为 $\mathbb{D}$。然后 $\mathbb{D}\times$ 将层数应用于表III中的模型。例如，对于 JSC-M Lite，如果 $\mathbb{D} = 2$，则

表 II：PolyLUT 与 PolyLUT-Add 的准确率和硬件结果比较（$\mathbb{D}=1, \mathbb{W}=1$）

模型	度 D	模型	扇入 $(F \times A)$	准确率(%)↑	查找表条目↓	LUT（占 1182240 的百分比）	FF（占 2364480 的百分比）	F_max（MHz）	延迟 RTL（周期）	生成（小时）
HDR	1	PolyLUT	6	93.8	2^{12}	3.43	0.12	378	6	1.40
			10	96.1	$2^{12} \times 256$		–			–
		PolyLUT-Add	6×2	96.5	$2^{12} \times 2 + 2^6$	12.69	0.12	378	6	3.00
			6×3	**96.6**	$2^{12} \times 3 + 2^9$	20.67	0.12	378	6	4.40
	2	PolyLUT	6	95.4	2^{12}	6.62	0.12	378	6	1.40
			10	97.3	$2^{12} \times 256$		–			–
		PolyLUT-Add	6×2	97.1	$2^{12} \times 2 + 2^6$	19.78	0.07	378	6	3.00
			6×3	**97.6**	$2^{12} \times 3 + 2^9$	31.36	0.07	378	6	4.50
JSC-XL	1	PolyLUT	3	74.5	2^{15}	19.55	0.07	235	5	2.10
			5	74.9	$2^{15} \times 1024$		–			–
		PolyLUT-Add	3×2	**75.1**	$2^{15} \times 2 + 2^{12}$	50.10	0.07	235	5	5.17
	2	PolyLUT	3	74.9	2^{15}	37.40	0.07	235	5	2.30
			5	75.2	$2^{15} \times 1024$		–			–
		PolyLUT-Add	3×2	**75.3**	$2^{15} \times 2 + 2^{12}$	89.60	0.07	235	5	5.24
JSC-M Lite	1	PolyLUT	4	71.6	2^{12}	0.97	0.01	646	3	0.16
			7	72.1	$2^{12} \times 512$		–			–
		PolyLUT-Add	4×2	72.2	$2^{12} \times 2 + 2^8$	2.62	0.01	488	3	0.35
			4×3	**72.3**	$2^{12} \times 3 + 2^{12}$	4.33	0.01	363	3	0.63
	2	PolyLUT	4	72.0	2^{12}	1.51	0.01	568	3	0.16
			6	72.5	$2^{12} \times 512$		–			–
		PolyLUT-Add	4×2	72.5	$2^{12} \times 2 + 2^8$	4.29	0.01	440	3	0.34
			4×3	**72.6**	$2^{12} \times 3 + 2^{12}$	6.57	0.01	373	3	0.64

–：由于超出现场可编程门阵列内存容量限制，省略了超高扇入设置的数据。

表 III：图 7 中 PolyLUT 基线模型的设置。

模型	每层神经元数	β	F	D
MNIST	256, 100, 100, 100, 100, 10	2	6	1,2
JSC-XL[1]	128, 64, 64, 64, 5	5	3	1,2
JSC-M Lite	64, 32, 5	3	4	1,2

当 $D=1$ 时，PolyLUT 与 LogicNets 等效。备注：
[1] $\beta_i = 7$， $F_i = 2$；

隐藏层数量加倍，这意味着每层神经元数变为 (64,64,32,32,5)。

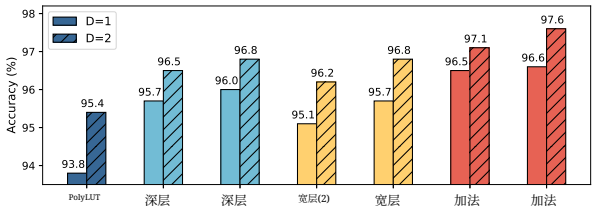
2) **PolyLUT-Wider**：本节研究更宽网络模型的影响。我们将宽度因子记为 $\mathbb{W}$。那么，$\mathbb{W}\times$ 表 III 中的模型采用每层神经元数。再次说明，对于 JSC-M Lite，若 $\mathbb{W}=2$，则每层神经元数变为 (128,64,5)。

图7 展示了上述配置下的准确率。在所有数据集上，无论是线性（$D=1$）还是非线性（$D=2$）情况，PolyLUT-Add 的准确率都高于所有基线。

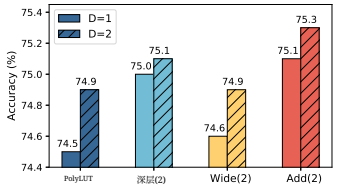
3）准确率相近时的比较：最后，我们研究了准确率相近时的面积和延迟。表IV展示了 PolyLUT-Add 与 PolyLUT 之间的比较。$A=2$ 以及更低的 F 用于所有 PolyLUT-Add 模型

（分别记为“HDR-Add2”、“JSC-XL-Add2”和“JSC-M Lite-Add2”，详细设置见表V ）。值得注意的是，PolyLUT分别应用于 HDR 和 JSC-XL 模型以及 $D=4$ 应用于 JSC-M Lite 模型，而 PolyLUT-Add 使用了更小的 D。

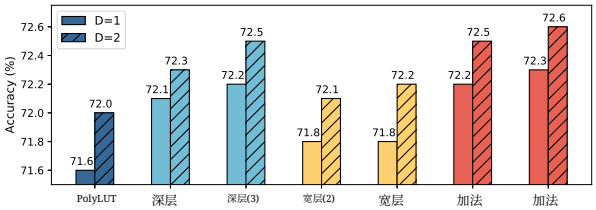
PolyLUT-Add 在 MNIST、JCS-XL 和 JSC-M Lite 基准测试中分别实现了 4.8×、6.5× 和 13.9× 的查找表缩减。至于延迟，他的方法



(a) HDR



(b) JSC-XL



(c) JSC-M Lite

图 7：不同模型的准确率结果。我们分别用 Deep($\mathbb{D}$)、Wide($\mathbb{W}$) 和 Add(A) 表示 “PolyLUT-Deeper” 、 “PolyLUT-Wider” 和 “PolyLUT-Add”。

分别在三个基准测试上实现了 1.6×、1.3× 和 1.2× 的降幅。这些显著的减少归因于更低的多项式次数 D 和更低的 F。

表 IV：PolyLUT-Add 与 PolyLUT 的比较结果。频率和面积数据采集自 Vivado 布局布线后的报告。

数据集	模型	准确率↑	LUT	FF	DSP	BRAM	F_max(MHz)↑	延迟(ns)↓
MNIST	**PolyLUT-Add（HDR-Add2, D=3）**	**96%**	**14810**	**2609**	**0**	**0**	**625**	**10**
	PolyLUT（HDR, D=4）[3]	**96%**	70673	4681	**0**	**0**	378	16
JSC（高精度）	**PolyLUT-Add (JSC-XL-Add2, D=3)**	**75%**	**36484**	**1209**	**0**	**0**	**315**	**16**
	PolyLUT（JSC-XL, D=4）[3]	**75%**	236541	2775	**0**	**0**	235	21
JSC（低精度）	**PolyLUT-Add（JSC-M Lite-Add2, D=3）**	**72%**	**895**	**189**	**0**	**0**	**750**	**4**
	PolyLUT（JSC-M Lite, D=6）[3]	**72%**	12436	773	**0**	**0**	646	5

∗：论文报告“LUT+FF”

C. 稀疏化评估

为了评估 SparseLUT 作为一种提高基于LUT的深度神经网络准确率的通用方法，我们在包括 LogicNets [2], PolyLUT [3], NeuraLUT [4],以及通过架构优化增强的 PolyLUT-Add 变体在内的多种最先进方法上测试了其性能。与 PolyLUT-Add 中使用的树结构扇入扩展技术类似，NeuraLUT-Assemble [22] 是 NeuraLUT 的一种扇入优化变体，也在本研究中进行了评估。我们比较了随机连接与 SparseLUT 提供的优化后的连接的有效性。此外，NeuraLUT-Assemble 还报告了使用静态稀疏化的结果；静态稀疏化是一种基于权重排序的训练后剪枝策略，其性能也纳入了我们的评估。

在搜索优化后的稀疏连接时，所有模式均采用相同的训练设置，以确保一致性。使用 PyTorch 中的 AdamW优化器 [28] ，并对每个模型进行 300 轮训练。阈值 T 设为 240 轮，相当于总训练时长的 80%。采用超参数 $\epsilon_1 = 10^{-12}$ 和 $10^{-5} \leq \epsilon_2 \leq 10^{-4}$。 F_i 设为 N ，以使训练从稠密模型开始。主要目标是获得优化后的稀疏模式 $\mathcal{M}$。采用全精度训练，以加快收敛速度，并更准确地识别关键连接。本节所呈现的结果基于这些配置，不过，进一步优化或采用其他训练技术，可能会提升特定应用的性能。

该实验包括两个步骤：(1) 推导优化后的连接性特征掩码 $\mathcal{M}$。(2) 使用 $\mathcal{M}$ 替换基线模型中的原始随机连接，并重新训练基线模型。

为确保公平比较，步骤 2 中的配置和训练超参数与原论文中报告并汇总于表V中的基线设置保持一致。本节中的所有结果均通过重新训练模型获得。

在步骤 1 中，我们使用全连接基线评估三种稀疏模式，如表 VI所示：

- **随机稀疏性**：反映现有技术目前采用的方法 [2–4, 13]。
- **DeepR***：原始的 DeepR [10] 无法直接应用于我们的问题。我们开发了一个修订版 DeepR*，支持固定扇入，并将其用作比较基线。
- **SparseLUT**：反映所提出的方法。

随后，我们以 MNIST数据集为对象开展了一项案例研究，重点进行了两项分析：（1）使用热力图可视化 SparseLUT 学习到的连接性，以评估其质量；（2）比较采用学习得到的连接性与随机连接性的查找表深度神经网络模型的测试准确率。

1) 连接分布可视化： 在测试其在查找表深度神经网络中的性能之前，我们缺乏用于评估连接质量的真实标注。然而，对于 MNIST数据集，手写数字通常出现在中心位置 1，因此可以合理地预期，最重要的连接会集中在中心区域。

我们采用了一个全连接模型和三个稀疏模型（随机、DeepR*、SparseLUT），其层神经元数量 {256, 100, 100, 100, 10}与表V中的 HDR-5L 模型一致。训练后，我们沿第二个轴计算第一层权重矩阵 $|W| \in \mathbb{R}^{784\times256}$ 绝对值的平均值，并将其重塑为 $W_{ave} \in \mathbb{R}^{28\times28}$，使其与输入图像尺寸相匹配。

图8 以热力图的形式可视化了 W_{ave}。全连接模型（最后一个子图）显示，较大的权重集中在中心区域，这与我们的假设一致。

图8 中的前三个子图展示了稀疏模型的热力图，其中每行的 $F = 6$ 个元素$|W|$ 非零。随机稀疏模型呈现均匀分布，体现了非结构化连接。DeepR*呈现出连接集中于中心区域的趋势，表明模型针对该任务进行了有意义的自适应。SparseLUT 的热力图与全连接模型更为相似，说明 SparseLUT 能有效学习优化后的连接模式，根据数据集的结构优先连接中心区域。

2) 与基线方法的比较：图 9 进一步比较了五种模型在三种稀疏配置下采用不同连接模式时的测试准确率：PolyLUT（$D = 1$）、PolyLUT（$D = 2$）、PolyLUT-Add（$D = 1$）、PolyLUT-Add（$D = 2$）和 NeuraLUT。三种稀疏配置分别为：随机稀疏性（含五个随机种子的箱线图）、DeepR*和 SparseLUT。对于 DeepR* 和 SparseLUT，每种模型优化后的准确率点均单独绘制。

结果表明，DeepR* 和 SparseLUT 的表现都优于随机稀疏性，但 SparseLUT 的优势更为显著。DeepR* 在某些情况下相较于随机稀疏性仅取得有限改进，例如 PolyLUT

[1]手写数字数据集的可视化：



表 V：基线模型的设置。

数据集	方法	模型	每层神经元数	β	F	D
MNIST	PolyLUT	HDR	256, 100, 100, 100, 100, 10	2	6	1,2
	PolyLUT-Add	HDR-Add2	256, 100, 100, 100, 100, 10	2	4	1,2
	NeuraLUT	HDR-5L	256, 100, 100, 100, 10	2	6	—
	NeuraLUT-Assemble	HDR-5L-0.1	2160, 360, 2160, 360, 60, 10	1	6	—
JSC（高精度）	PolyLUT	JSC-XL[1]	128、64、64、64、5	5	3	1,2
	PolyLUT-Add	JSC-XL-加法2[2]	128, 64, 64, 64, 5	5	2	1,2
	NeuraLUT	JSC-5L[1]	128, 128, 128, 64, 5	4	3	—
	NeuraLUT-Assemble	jsc-cernbox[3]	320、160、80、40、20、10	4	2	—
JSC（准确率低）	PolyLUT	JSC-M Lite	64、32、5	3	4	1,2
	PolyLUT-Add	JSC-M Lite-Add2	64, 32, 5	3	2	1,2
	NeuraLUT	JSC-2L	32, 5	4	3	—
CIFAR-10	PolyLUT	HDR	256, 100, 100, 100, 100, 10	2	6	1,2
	PolyLUT-Add	HDR-Add2	256, 100, 100, 100, 100, 10	2	4	1,2
	NeuraLUT	HDR-5L	256, 100, 100, 100, 10	2	6	—
	NeuraLUT-Assemble	HDR-5L-0.1	2160, 360, 2160, 360, 60, 10	1	6	—

PolyLUT 与 $D = 1$ 等同于 LogicNets。备注：[1] $\beta_i = 7$、$F_i = 2$；
[2] $\beta_i = 7$、$F_i = 1$；[3] $\beta_i = 8$、$F_i = 1$；

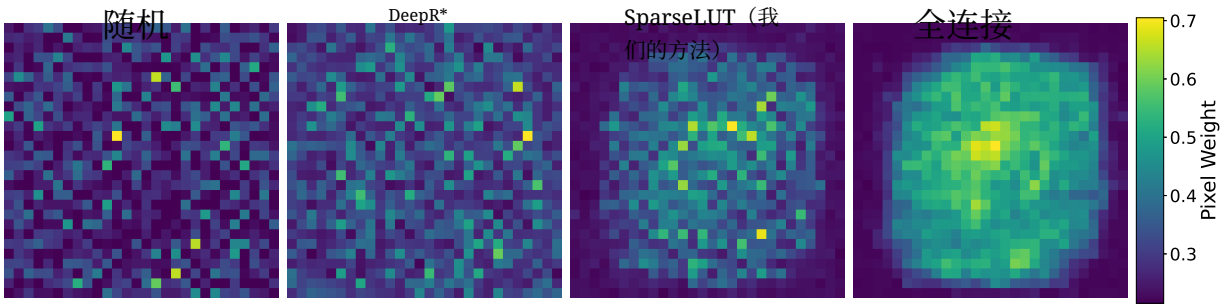


图 8：三种稀疏模式（随机稀疏性、DeepR* 和 SparseLUT）下第一层平均权重矩阵的热力图，并以全连接模式作为基线。

表 VI：稀疏模式

Mode	固定扇入约束
全连接	—
随机稀疏性	✓
DeepR* [10]	✓
SparseLUT（我们的方法）	✓

（$D = 1$）和 PolyLUT-Add（$D = 2$），其准确率点虽略高于均值线，却与随机稀疏性的箱形区域重叠。相比之下，SparseLUT 始终带来显著提升，超出随机稀疏性范围 1.4%–2.1%。这一性能差距凸显了 SparseLUT 的非贪心训练方法的有效性：该方法在早期阶段提供了更大的参数搜索空间，使模型能够探索更广泛的稀疏模式。

3) 测试准确率分析：在本小节中，我们展示了 SparseLUT 在多个基线模型上的性能。对于 MNIST 和 JSC 数据集，随机稀疏性下的准确率数值取自参考论文和开源实现。如表 VII 所示，SparseLUT 在所有基线模型上均持续优于随机稀疏配置，以及静态稀疏化

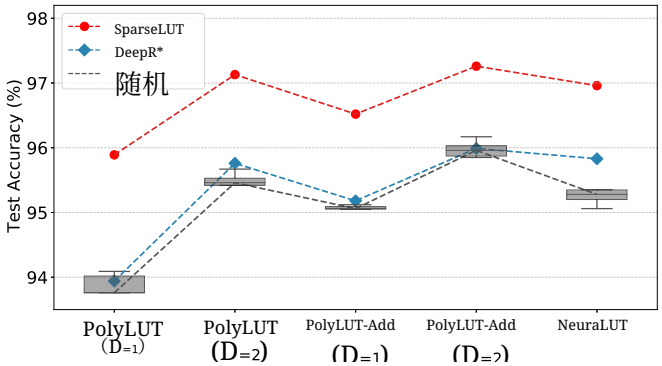


图 9：采用随机稀疏性（5 个随机种子）、DeepR* 和 SparseLUT 的五个模型的测试准确率。

增强后的 NeuraLUT-Assemble 模型的准确率（左侧报告随机准确率，右侧报告优化后的准确率）。

为进一步评估所提出的连接优化对模型准确率的普适性，表 VII报告了在 CIFAR-10 数据集上的结果。比较结果表明，在所评估的模型中，SparseLUT 优于随机稀疏性。CIFAR-10 上相对较低的准确率可归因于现有基于查找表的深度神经网络架构设计较为简单

表VII：与先前研究报告的准确率进行比较。准确率$^{+opt}$ 表示通过 SparseLUT 优化连接性后的准确率。

数据集	设置	模型	准确率	准确率$^{+opt}$	稠密
MNIST	HDR(D=1) [3]	PolyLUT	93.76%	**95.89%**	98.55%
	HDR(D=2) [3]	PolyLUT	95.42%	**97.13%**	
	HDR-Add2(D=1) [13]	PolyLUT-Add	95.09%	**96.52%**	
	HDR-Add2(D=2) [13]	PolyLUT-Add	95.87%	**97.26%**	
	HDR-5L [4]	NeuraLUT	95.20%	**96.96%**	98.61%
	HDR-5L-0.1 [22]	NeuraLUT-Assemble	95.52%, 97.90%	**98.10%**	98.87%
JSC（高精度）	JSC-XL(D=1) [3]	PolyLUT	74.48%	**74.65%**	75.46%
	JSC-XL(D=2) [3]	PolyLUT	74.94%	**75.01%**	
	JSC-XL-Add2(D=1) [13]	PolyLUT-Add	74.64%	**74.74%**	
	JSC-XL=加法2(D=2) [13]	PolyLUT-Add	74.98%	**75.04%**	
	JSC-5L [4]	NeuraLUT	74.93%	**74.98%**	75.31%
	jsc-cernbox [22]	NeuraLUT-Assemble	74.85%，75.00%	**75.03%**	75.26%
JSC（低精度）	JSC-M Lite(D=1) [3]	PolyLUT	71.65%	**72.10%**	72.48%
	JSC-M Lite(D=2) [3]	PolyLUT	71.98%	**72.15%**	
	JSC-M Lite-加法2(D=1) [13]	PolyLUT-Add	71.53%	**71.96%**	
	JSC-M Lite-加法2(D=2) [13]	PolyLUT-Add	71.90%	**72.24%**	
	JSC-2L [4]	NeuraLUT	72.01%	**72.95%**	73.34%
CIFAR-10	HDR(D=1) [3]	PolyLUT	38.83%	**40.99%**	60.95%
	HDR(D=2) [3]	PolyLUT	42.66%	**46.10%**	
	HDR-Add2(D=1) [13]	PolyLUT-Add	40.81%	**42.74%**	
	HDR-Add2(D=2) [13]	PolyLUT-Add	44.27%	**46.82%**	
	HDR-5L [4]	NeuraLUT	42.98%	**46.38%**	61.38%
	HDR-5L-0.1 [22]	NeuraLUT-Assemble	44.84%	**46.27%**	62.51%

LogicNets [2] 种情况与所有基线情况等效，且具有 $D = 1$。

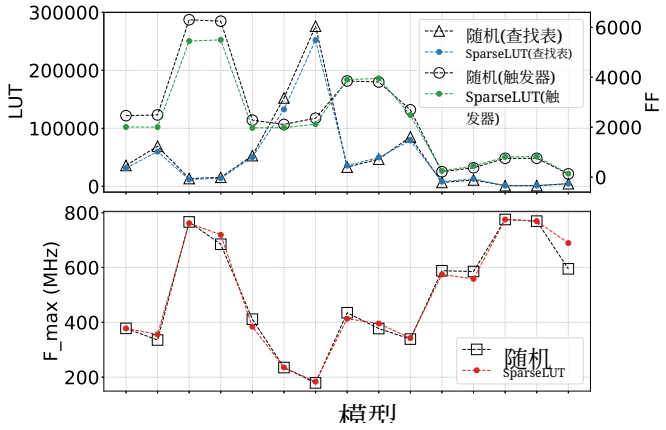


图10：模型布局布线后的硬件比较（从左到右的顺序与表 VII 相同，即从 PolyLUT 到 NeuraLUT），比较是否采用 SparseLUT 的连接性。

。开发能够在 CIFAR-10 等复杂任务乃至更大规模基准测试上达到实用准确率（例如超过 90%）的基于查找表的深度神经网络，仍然是 FPGA社区面临的一项开放挑战。

为了评估通过优化稀疏性所能实现的准确率提升上限，我们采用 LogicNets 的全连接模型（即，多项式次数 $D = 1$），作为公平的比较基准，用于比较 PolyLUT（$D = 1$）和 PolyLUT-Add（$D = 1$）配置。对于采用 $D > 2$ 或 Neura-LUT 网络中网络架构的模型，全连接实现会导致参数数量呈指数级增长，使此类实现不切实际。因此，表 VII 最后一列中这些配置的稠密准确率被视为合理的参考指标，而非上限。

结果凸显了不同数据集和模型之间准确率差值（δ =稠密准确率 - 随机稀疏准确率）的显著差异。对于 MNIST 和 CIFAR-10，SparseLUT 实现了大幅准确率提升，这归因于较大的 δ（超过 3%）。相比之下，对于 JSC数据集中的大多数模型，这一 δ 要小得多，约为 0.8%，这解释了优化后的连接带来的准确率提升相对有限。JSC数据集中的一个显著例外是 NeuraLUT 上的 JSC-2L模型（最后一行），其 δ 为 1.33%。SparseLUT 将其准确率从 72.01% 大幅提升至 72.95%，这可能得益于 JSC-2L模型更高的字长（$\beta = 4$），相比之下，其他 JSC-M Lite模型的字长为 3 位（$\beta = 3$）。

这些发现表明，SparseLUT 的有效性与底层模型的连接优化潜力密切相关，这一点体现在 δ 密集配置与稀疏配置之间的比较中。这凸显了数据集特征和模型架构在决定使用 Sparse-LUT 可实现的准确率提升幅度方面所起的关键作用。

表VIII：广义查找表深度神经网络与其他超低延迟机器学习技术的比较。

数据集	模型	准确率↑	准确率$^{+opt}$ ↑	查找表	FF	数字信号处理器	块随机存取存储器	F_{max}(MHz)↑	延迟(ns)↓
MNIST	PolyLUT（HDR, D=2） [3]	95.42%	97.13%	59790	2003	**0**	**0**	356	17
	PolyLUT-Add（HDR-Add2, D=2） [13]	95.87%	97.26%	13890	2768	**0**	**0**	635	8
	NeuraLUT（HDR-5L） [4]	95.20%	96.96%	50271	1973	**0**	**0**	384	13
	NeuraLUT-Assemble（HDR-5L-0.1） [22]	97.90%	**98.10%**	5076	725	**0**	**0**	863	**2.1**
	DWN [17]	97.8%	-	**2092**	1757	**0**	**0**	**873**	9.2
	DWN^{arxiv} [37]	**98.3%**	-	4082	3385	**0**	**0**	827	6
	TreeLUT [16]	96.6%	-	4478	**597**	**0**	**0**	791	2.5
	PolyLUT^{extension} [21]	97.5%	-	75131	4668	**0**	**0**	353	17
	FINN [38]	96%	-	91131	-	**0**	5	200	310
	hls4ml [14]	95%	-	260092	165513	**0**	345	200	190
JSC（高精度）	PolyLUT（JSC-XL, D=2） [3]	74.94%	75.01%	252077	2121	**0**	**0**	183	27
	PolyLUT-Add (JSC-XL-Add2, D=2) [13]	74.98%	**75.04%**	49766	1976	**0**	**0**	395	13
	NeuraLUT（JSC-5L） [4]	74.93%	74.98%	79012	2485	**0**	**0**	342	15
	NeuraLUT-Assemble（jsc-cernbox） [22]	75.00%	75.03%	8556	1335	**0**	**0**	352	5.7
	Duarte 等人 [33]	75%	-	88797*		954	**0**	200	75
	TreeLUT [16]	75.6%	-	**2234**	**347**	**0**	**0**	735	**2.7**
	DWN^{arxiv} [37]	**76.3%**	-	4972	3305	**0**	**0**	**827**	7.3
	PolyLUT^{extension} [21]	75.1%	-	246071	12384	**0**	**0**	203	25
	Fahim等人 [35]	76.2%	-	63251	4394	38	**0**	200	45
JSC（低精度）	LogicNets [2]	71.65%	72.10%	7995	249	**0**	**0**	574	5
	PolyLUT（JSC-M Lite, D=2） [3]	71.98%	72.15%	12855	447	**0**	**0**	558	5
	PolyLUT-Add（JSC-M Lite-Add2, D=2） [13]	71。90%	72.24%	1197	410	**0**	**0**	769	4
	NeuraLUT (JSC-2L) [4]	72.01%	**72.95%**	4679	140	**0**	**0**	689	3
	DWN^{arxiv} [37]	**74.0%**	-	**110**	**72**	**0**	**0**	**1094**	**1.5**
	PolyLUT^{extension} [21]	72.5%	-	10169	631	**0**	**0**	598	5

1. ∗：论文报告“LUT+触发器” 2. DWN^{arxiv} 是包含更新结果的 DWN arXiv 版本，PolyLUT^{extension} 是包含更新结果的 PolyLUT 期刊扩展版本。

SparseLUT 不会改变生成的 RTL设计中的查找表条目数量；图 10展示了 Vivado 在相同时钟周期约束下的布局布线结果，包括面积（查找表和触发器（FF））以及最大频率（最大频率）。值得注意的是，SparseLUT 实现了相近的硬件消耗，且最大频率比较显示没有速度损失。因此，我们得出结论：SparseLUT 所实现的准确率提升并未带来额外的硬件和延迟开销。

表VIII 将上述讨论的连接优化模型与其他低延迟机器学习方法进行了比较。由于 DWN 在 CIFAR-10 上的准确率为 57.42%，而所有传统的基于LUT的深度神经网络的准确率均低于 50%（见表 VII），因此现阶段评估其硬件成本并无意义。因此，探索这两种方法的潜力仍是 FPGA 社区未来研究的一个开放方向。

表IX：任务执行时间

Task	Time
连通性搜索	2.6h
查找表深度神经网络训练（PolyLUT）	3.9h
RTL 生成	1.4h
综合、布局与布线	2.5h

4) 运行时间结果：最后，表IX详细列出了所涉及各项任务的执行时间，并表明连接性搜索的运行时间不会显著影响整体系统性能。为说明这一点，我们测量了在 MNIST 任务上实现 PolyLUT ($D = 2$) 模型的运行时间。综合、布局和布线在配备 Intel Xeon E-2356G 中央处理器和 94GB 随机存取存储器的机器上进行，而所有其他进程则在配备 Intel Core i7-10700F 中央处理器、 64GB 随机存取存储器和 NVIDIA RTX 2060 Super 图形处理器的系统上执行。

五、结论

本文介绍了 SparseLUT，这是一种综合性框架，通过两种正交优化提升基于查找表的深层神经网络（DNN）在现场可编程门阵列上的部署：架构增强和基于训练的连接优化。

在架构层面，我们提出了一种高效结构，通过加法器组合多个 PolyLUT子神经元，在不牺牲准确率的前提下，显著降低了硬件成本和延迟。实验结果表明，在 MNIST 和喷注子结构分类等多种基准测试中，该方法可节省 2.0×–13.9× 个查找表，并将延迟降低 1.2×–1.6× 。

此外，我们提出了一种非贪心动态稀疏训练算法，可在固定扇入约束下优化每个神经元的连接模式，且不会增加查找表或布线开销。通过智能地剪枝并重新生长连接，SparseLUT 的准确率始终优于现有的查找表深度神经网络，例如 LogicNets [2], PolyLUT [3],和 NeuraLUT [4], 在 MNIST 上最高可提升 2.13%，在喷注子结构分类上最高可提升 0.94%。

参考文献

[1] Yann LeCun、Yoshua Bengio 和 Geoffrey Hinton。深度学习。《自然》，521(7553):436，2015。[2] Yaman Umuroglu、Yash Akhauri、Nicholas James Fraser 和 Michaela Blott。LogicNets：面向极高吞吐量应用的神经网络与电路协同设计。载于*2020*年第*30*届国际现场可编程逻辑与应用会议（*FPL*），第291–297页。IEEE，2020。[3] Marta Andronic 和 George A Constantinides。PolyLUT：用于超低延迟现场可编程门阵列查找表推理的分段多项式学习。载于*2023*年国际现场可编程技术会议（*ICFPT*），第60–68页。IEEE，2023。[4] Marta Andronic 和 George A Constantinides。NeuraLUT：将神经网络密度隐藏于可通过布尔综合实现的函数中。载于*2024*年第*34*届国际现场可编程逻辑与应用会议（*FPL*），第140–148页。IEEE，2024。[5] Yann LeCun、John Denker 和 Sara Solla。最优脑损伤。神经信息处理系统进展，第2卷，1989。[6] Georg Thimm 和 Emile Fiesler。评估剪枝方法。载于国际人工神经网络研讨会论文集，第20–25页，1995。[7] Trevor Gale、Erich Elsen 和 Sara Hooker。深度神经网络中的稀疏性现状。*arXiv* 预印本 *arXiv:1902.09574*，2019。[8] Vivienne Sze、Yu-Hsin Chen、Tien-Ju Yang 和 Joel S Emer。深度神经网络的高效处理。Springer，2020。[9] Michael Zhu 和 Suyog Gupta。要不要剪枝：探究剪枝在模型压缩中的有效性。*arXiv* 预印本 *arXiv:1710.01878*，2017。[10] Guillaume Bellec、David Kappel、Wolfgang Maass 和 Robert Legenstein。深度重连：训练极稀疏的深度网络。*arXiv* 预印本 *arXiv:1711.05136*，2017。[11] Utku Evci、Trevor Gale、Jacob Menick、Pablo Samuel Castro 和 Erich Elsen。操纵彩票：让所有彩票券都成为赢家。载于国际机器学习大会，第2943–2952页。PMLR，2020。[12] Mike Lasby、Anna Golubeva、Utku Evci、Mihai Nica 和 Yani Ioannou。具有结构化稀疏性的动态稀疏训练。*arXiv* 预印本 *arXiv:2305.02299*，2023。[13] Binglei Lou、Richard Rademacher、David Boland 和 Philip H.W. Leong。PolyLUT-Add：基于现场可编程门阵列的宽输入查找表推理。*2024*年第*34*届国际现场可编程逻辑与应用会议（*FPL*），第149–155页，2024。[14] Jennifer Ngadiuba、Vladimir Loncar、Maurizio Pierini、Sioni Summers、Giuseppe Di Guglielmo、Javier Duarte、Philip Harris、Dylan Rankin、Sergo Jindariani、Mia Liu 等。使用 hls4ml 将现场可编程门阵列上的深度神经网络压缩为二值和三值精度。《机器学习：科学与技术》，2(1):015001，2020。[15] Md Maruf Hossain Shuvo、Syed Kamrul Islam、Jianlin Cheng 和 Bashir I Morshed。高效加速资源受限边缘设备上的深度学习推理：综述。*IEEE* 会议录，111(1):42–91，2022年。[16] Alireza Khataei 和 Kia Bazargan。TreeLUT：一种高效的深度神经网络替代方案，利用梯度提升决策树加速推理。发表于*2025* 年 *ACM/SIGDA* 国际现场可编程门阵列研讨会论文集，FPGA ’25，2025年。[17] Alan T. L. Bacellar、Zachary Susskind、Mauricio Breternitz Jr.、Eugene John、Lizy K. John、Priscila M. V. Lima 和 Felipe M. G. França。可微无权神经网络。发表于第 *41* 届国际机器学习大会（*ICML*）论文集，2024年。[18]Min Lin、Qiang Chen 和 Shuicheng Yan。网络中的网络。发表于*2014* 年国际学习表征会议（*ICLR*），第 1–10 页，加拿大艾伯塔省班夫，2014年4月。[19] Christos Louizos、Max Welling 和 Diederik P Kingma。通过正则化学习稀疏神经网络。*arXiv* 预印本 *arXiv:1712.01312*，2017年。[20] Dmitry Molchanov、Arsenii Ashukha 和 Dmitry Vetrov。变分 Dropout 使深度神经网络变得稀疏。发表于国际机器学习大会，第 2498–2507 页。PMLR，2017年。[21]Marta Andronic、Jiawen Li 和 George A. Constantinides。PolyLUT：通过硬件感知结构化剪枝实现超低延迟多项式推理。*IEEE* 计算机汇刊，74(9):3181–3194，2025年。[22] Marta Andronic 和 George A. Constantinides。NeuraLUT-assemble：硬件感知地组装子神经网络，以实现高效的查找表推理。发表于*2025* 年 *IEEE* 第 *33* 届年度国际现场可编程定制计算机研讨会（*FCCM*），第 208–216 页。IEEE，2025年。[23] Anthony JGD Holtmaat、Joshua T Trachtenberg、Linda Wilbrecht、Gordon M Shepherd、Xiaoqun Zhang、Graham W Knott 和 Karel Svoboda。活体新皮层中短暂和持久存在的树突棘。神经元，45(2):279–291，2005年。[24]Dan D Stettler、Homare Yamahachi、Wu Li、Winfried Denk 和 Charles D Gilbert。成年视觉皮层中的轴突和突触膨体具有高度动态性。神经元，49(6):877–887，2006年。[25]Alessio Attardo、James E Fitzgerald 和 Mark J Schnitzer。活体成年海马 CA1 区树突棘的非持久性。《自然》，523(7562):592–596，2015年。[26]Anna R Chambers 和 Simon Rumpel。由不稳定组件构成的稳定大脑：新兴概念及其对神经计算的启示。《神经科学》，357:172–184，2017年。[27] Shiliang Sun、Zehui Cao 等。机器学习视角下的优化方法综述。*IEEE* 控制论汇刊，50(8):3668–3681，2019年。[28] Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Ze ming Lin、Natalia Gimelshein、Luca Antiga、Alban Desmaison、Andreas Kopf、Edward Yang、Zachary DeVito、Martin Raison、Alykhan Tejani、Sasank Chilamkurthy，

Benoit Steiner、Lu Fang、Junjie Bai 和 Soumith Chintala。PyTorch：一种命令式风格的高性能深度学习库。发表于神经信息处理系统进展第 *32* 卷，第 8024–8035 页。Curran Associates 公司，2019年。[29] Alessandro Pappalardo。Xilinx/brevitas，2023年。[30] Xilinx 公司。《Vivado 设计套件用户指南》（UG949），2020年。[31] Li Deng。用于机器学习研究的 MNIST 手写数字图像数据库 [网络精选]。*IEEE* 信号处理杂志，29(6):141–142，2012年。[32] Jennifer Ngadiuba、Vladimir Loncar、Maurizio Pierini、Sioni Summers、Giuseppe Di Guglielmo、Javier Duarte、Philip Harris、Dylan Rankin、Sergo Jindariani、Mia Liu 等。利用 hls4ml 将现场可编程门阵列上的深度神经网络压缩至二值和三值精度。《机器学习：科学与技术》，2(1):015001，2020年。[33] Javier Duarte、Song Han、Philip Harris、Sergo Jindariani、Edward Kreinar、Benjamin Kreis、Jennifer Ngadiuba、Maurizio Pierini、Ryan Rivera、Nhan Tran 等。用于粒子物理的现场可编程门阵列深度神经网络快速推理。《仪器学杂志》，13(07):P07027，2018年。[34]Claudionor N Coelho、Aki Kuusela、Shan Li、Hao Zhuang、J ennifer Ngadiuba、Thea Klaeboe Aarrestad、Vladimir Loncar、Maurizio Pierini、Adrian Alan Pol 和 Sioni Summers。面向粒子探测器边缘低延迟推理的深度神经网络自动异构量化。《自然·机器智能》，3(8):675–686，2021年。[35] Farah Fahim、Benjamin Hawks、Christian Herwig、James Hirschauer、Sergo Jindariani、Nhan Tran、Luca P Carloni、Giuseppe Di Guglielmo、Philip Harris、Jeffrey Krupa 等。hls4ml：一种开源协同设计工作流，助力打造科学领域的低功耗机器学习设备。*arXiv* 预印本 *arXiv:2103.05579*，2021年。[36] Alex Krizhevsky。从小型图像中学习多层特征。技术报告，多伦多大学，2009年。[37] Alan TL Bacellar、Zachary Susskind、Mauricio Breternitz Jr、Eugene John、Lizy K John、Priscila Lima 和 Felipe MG França。可微无权神经网络。*arXiv* 预印本 *arXiv:2410.11112*，2024年。[38] Yaman Umuroglu、Nicholas J Fraser、Giulio Gambardella、Michaela Blott、Philip Leong、Magnus Jahre 和 Kees Vissers。FINN：一种用于快速、可扩展二值化神经网络推理的框架。发表于*2017* 年 *ACM/SIGDA* 国际现场可编程门阵列研讨会论文集，第 65–74 页，2017年。