arXiv:2601.02613v1 [cs.AR] 6 Jan 2026

用于自动调制分类的稀疏性感知流式脉冲神经网络加速器与输出通道数据流

Kuilian Yang，*IEEE* 学生会员，Li Zhang、Ahmed M. Eltawil，*IEEE* 高级会员，以及 Khaled Nabil Salama，*IEEE* 高级会员

摘要——**包括 5G、新兴的6G网络以及物联网（IoT）的大规模部署在内，无线通信技术的快速发展加剧了对高效频谱利用的需求。自动调制分类（AMC）在认知无线电系统中发挥着关键作用，可实时识别调制方案，以支持动态频谱接入和干扰抑制。尽管深度神经网络（DNN）具有较高的分类准确率，但其计算和能耗需求给实时边缘部署带来了挑战。脉冲神经网络（SNN）凭借其事件驱动特性，具有固有的能效优势；然而，在硬件资源受限的条件下，同时实现高吞吐量和低功耗仍面临挑战。本文提出一种面向AMC任务优化的稀疏性感知脉冲神经网络流式加速器。传统脉动阵列虽能利用稀疏性，却存在吞吐量较低的问题；流式架构虽能实现高吞吐量，却无法充分利用输入和权重的稀疏性。与之不同，我们的设计兼具两者优势。利用推理过程中卷积核固定这一特性，我们采用门控一对多乘积（GOAP）算法，仅对非零输入与权重的交集进行计算。额外迭代或空迭代会预先计算并嵌入推理数据流，从而消除动态数据读取，并实现全流水线、无需控制的层间执行。在现场可编程门阵列上实现后，我们的稀疏性感知输出通道数据流流式（SAOCDS）加速器在 RadioML 2016 数据集上达到 23.5 MS/s（约为基线吞吐量的两倍），同时降低了动态功耗，并保持了相当的分类准确率。这些结果表明，该加速器在边缘认知无线电系统中进行实时、低功耗部署具有巨大潜力。**

索引词—**自动调制分类（AMC）、认知无线电（CR）、边缘设备、深度学习、脉冲神经网络（SNN）、流式加速器、稀疏性感知数据流、门控一对全部乘积算法。**

I. 引言

THE 物联网 (IoT) 设备的快速普及改善了人们的日常生活，但也使其在不可信电磁环境中易遭受恶意攻击[1]。自动调制分类 (AMC) 是识别潜在攻击信号调制格式的重要工具，有助于频谱风险监测和威胁检测。自动调制分类的应用涵盖军事和民用领域，包括频谱管理、电子侦察和电子对抗措施 [2]。在现代通信系统中，尤其是采用软件定义无线电的系统，高效识别调制类型对于有效进行信号解调和提升整体系统性能至关重要。

基于深度学习的自动调制分类技术已在各种应用中展现出优异性能 [3], [4], [5]。然而，无线电信号的高采样率给远程处理带来了挑战，原因在于带宽限制。因此，在嵌入式边缘设备上直接部署 AMC，已成为实时系统中一种实用且高效的解决方案。尽管传统人工神经网络（ANN）具有很高的准确率，但其计算和存储需求使其不太适用于资源受限的边缘平台 [6]。相比之下，脉冲神经网络（SNN）具有显著的低功耗运行潜力，这得益于其事件驱动特性。在 SNN 中，二值输入特征图（IFM）可将稠密乘累加（MAC）运算替换为更简单的累加，从而降低计算复杂度——鉴于乘法器的硬件成本，这一点尤为重要。事件驱动行为还使 IFM 具有高度时间稀疏性。当这种稀疏性与卷积核中的空间稀疏性相结合时——后者可通过剪枝和量化等训练技术实现——SNN 为开发适用于边缘部署的高能效、高吞吐量实时 AMC 系统提供了有前景的途径。

由于脉冲神经网络具有独特的计算模式，中央处理器和图形处理器等传统计算单元往往难以高效执行脉冲神经网络。因此，专用数字硬件加速器应运而生，成为颇具前景的解决方案，主要分为脉动阵列加速器和流式架构两类。

脉动阵列加速器采用固定的同构处理单元（PE）网格，在不同脉冲神经网络层中执行计算。它们通常依靠集中式控制逻辑（如路由器和调度器）分发输入和权重，以实现基于累加的运算。STICKER [7], 、SNAP [8],、ESSA [9], 和 LoAS [10] 等设计利用空间稀疏性和/或时间稀疏性来跳过冗余计算。然而，逐层调度和频繁的内存访问限制了层间并行性，从而制约了吞吐量。

相比之下，流式架构以异构方式实例化每个脉冲神经网络层，并配备本地存储权重，从而实现层间的连续数据流，无需外部存储器

Kuilian Yang、Li Zhang、Ahmed M. Eltawil 和 Khaled Nabil Salama 任职于沙特阿拉伯图瓦尔 23955 的阿卜杜拉国王科技大学计算机、电气与数学科学及工程系（电子邮件：kuil-ian.yang@kaust.edu.sa；li.zhang@kaust.edu.sa；ahmed.eltawil@kaust.edu.sa；khaled.salama@kaust.edu.sa）。

访问。这消除了对全局控制逻辑的需求，从而降低延迟并提高吞吐量。S2N2 [11]以及其他基于 FINN 的实现 [12] exemplify this approach。然而，由于缺乏细粒度控制逻辑，这些设计通常无法有效跳过零权重或零输入，主要仅利用时间稀疏性，导致冗余计算并降低能效。

为克服现有架构在稀疏性利用与吞吐量之间的权衡问题，我们提出了一种面向基于SNN的AMC系统的稀疏性感知输出通道数据流流式加速器（SAOCDS）。

据我们所知，这是首个同时整合时间与空间稀疏性的全流水线流式脉冲神经网络加速器，无需额外控制逻辑即可实现自动工作负载均衡，从而在资源约束下实现高吞吐量。

所提出的设计在 RadioML 2016 数据集 [13], 上实现了 23.5 MS/s 的吞吐量，同时保持较低的动态功耗和具有竞争力的分类准确率。

本工作的主要贡献总结如下：

1) 我们提出了一种同时利用时间与空间稀疏性的流式架构，在保持高吞吐量的同时实现更高能效的计算。 2) 我们采用输出通道数据流策略，该策略能够自然地平衡计算工作负载，并简化层间数据传输。 3) 我们开发了一种轻量级错误检测算法，用于检测和处理潜在的逻辑冲突，确保层间数据流畅通无阻，且不会产生额外的控制开销。 4) 我们通过在 AMC 任务（例如 RadioML 2016）上的实现证明，在硬件资源有限的情况下，SAOCDS 加速器能够同时实现高吞吐量和具有竞争力的准确率，展示了其作为一种可行的新部署方案，在资源受限且对吞吐量敏感的场景中应用于脉冲神经网络。

本文其余部分安排如下：第二节回顾相关研究，并指出现有的稀疏性利用机会。第三节分析在流式架构中同时整合时间与空间稀疏性的关键挑战，随后介绍我们提出的稀疏性感知输出通道数据流流式脉冲神经网络（SAOCDS）加速器。第四节介绍用于自动调制分类的训练后脉冲神经网络架构，并报告其分类结果。第五节详细介绍我们的现场可编程门阵列实现与实验评估。最后，第六节总结全文。

II. 背景

本节首先综述现有的脉冲神经网络加速器设计，评估它们如何利用稀疏性并识别其局限。随后，我们将重点介绍提升效率的尚存机遇，这些机遇也为我们提出的设计提供了动机。

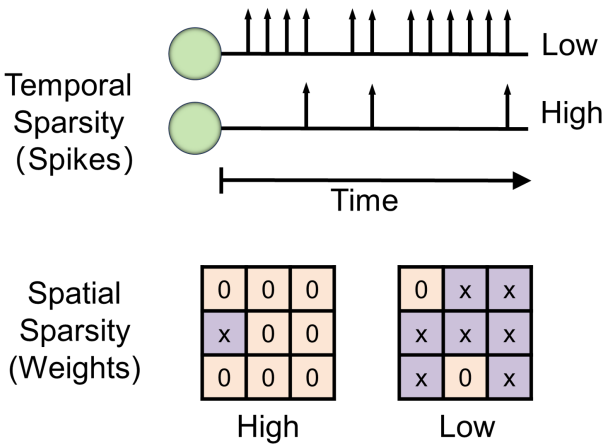

图 1. 脉冲神经网络中的时间稀疏性与空间稀疏性

A. 相关脉冲神经网络加速器研究

脉冲神经网络固有的稀疏性主要体现为两种形式（图 1）：时间稀疏性，即脉冲生成的事件驱动特性所产生的稀疏性——神经元仅在达到阈值时才会触发脉冲；以及空间稀疏性，指取值为零的突触权重，这通常是训练过程中进行剪枝或量化的结果。

已有多种加速器被提出，用于利用脉冲神经网络中的稀疏性。STICKER [7] 和 SNAP [8] 采用输入优先策略，通过坐标元数据压缩非零输入特征图元素，以检索对应的权重。然而，这会增加存储开销，因为 1 位输入脉冲需要多个位的元数据。ESSA [9] 通过扫描输入行，并采用压缩稀疏行格式（CSR）存储权重，利用时间与空间稀疏性。但大量元数据仍会带来相当大的开销。LoAS [10]采用两阶段位图编码，以简化输入特征图存储并减少冗余读取。然而，由于输出像素是逐个计算的，且多个时间步并行处理，因此需要负载均衡器，以便在各存储体之间均匀分配工作负载。

所有这些设计都采用输入优先策略，利用运行时元数据定位对应的权重。这会因输入流传输期间的实时解码而增加延迟和逻辑开销。相比之下， [14] 提出了一种基于门控一对全乘积（GOAP）算法的权重优先策略，该算法遍历非零权重，并利用预计算元数据获取相关输入。由于推理期间权重固定，这种方法减少了运行时计算。然而，与以往设计一样，它仍基于传统脉动阵列，因此层间并行性受限，需要频繁访问存储器，且仍依赖路由器和调度器等控制组件。尽管这些稀疏性感知加速器能够跳过冗余操作并提高能效，但受其底层架构所限，仍无法充分发挥流式吞吐量。

为提升推理性能，一些脉冲神经网络加速器采用了基于FINN的流式架构 [12],

[11], 其中，每一层都被实例化为专用硬件模块，权重存储在本地。这使得层间数据流可以直接传递，无需将中间结果写入全局存储器，从而降低存储器延迟，并免去了运行时解码器。尽管由于实例化所有层而增加了资源使用量，但这种方法能够实现较高的吞吐量。然而，由于缺乏利用稀疏性的控制逻辑，所有输入都会被统一处理，而不论它们是否对输出有实质性贡献。因此，尽管该架构具有较高的吞吐量，仍可能执行冗余计算，限制其效率。

总而言之，当前的加速器设计在计算效率与吞吐量之间进行权衡：控制逻辑可以跳过冗余操作，但可能引入开销，从而降低速度。因此，将层间并行性与无效操作的稀疏性感知跳过相结合，以维持高吞吐量，仍然是一项关键挑战。

B. 稀疏性利用的机遇

通过分析现有流式架构脉冲神经网络加速器的运行原理 [12], [11],，我们得出了以下观察结果：

1) 输入–权重对的有效性：由于矩阵–向量乘法在任一操作数为零时都会产生零，因此零输入或零权重会使相应的计算变得冗余。因此，流式架构面临的主要挑战是，在仅使用最少控制逻辑的情况下，识别有效的（非零）输入–权重对。

2) 数据读取顺序的影响：在推理过程中，IFM 是动态的，而卷积核（权重）保持不变。由于所有权重值均已预先知晓，权重优先方案可以利用预计算元数据定位非零权重，从而避免实时解码、额外的解码器或存储器读取。相比之下，输入优先方法需要对 IFM 元数据进行实时解码，以读取相应的权重，这使硬件设计更加复杂。

3) 输入与权重的获取开销差异：尽管在流式架构中，输入特征图和权重都存储在本地存储器（例如静态随机存取存储器或暂存存储器）中，但输入特征图通常采用二进制格式，而硬件中的权重则以定点格式（例如16位）存储。因此，获取输入特征图通常比读取权重更有效率。

基于这些观察，我们旨在将时间与空间稀疏性结合应用于脉冲神经网络流式架构，以提升效率，同时保持高吞吐量。

III. 稀疏性感知输出通道数据流流式架构（SAOCDS）加速器

鉴于射频任务对高吞吐量的要求，我们采用流式架构加速器。根据上一节所发现的稀疏性利用机会，我们首次提出一种全流水线流式脉冲神经网络加速器，将时间与空间稀疏性结合起来。该设计可在处理单元之间自动实现负载均衡，并在资源约束下仍保持高吞吐量。本节首先分析在流式架构中结合这两种稀疏性所面临的挑战。基于上述分析，我们介绍所提出的稀疏性感知加速器，并说明如何将稀疏性利用算法嵌入其流式数据流中。

A. 流式架构中时空稀疏性的结合所面临的挑战

设计流式架构时，通常会对所有输入处理单元应用统一操作，以平衡工作负载并尽量减少控制逻辑的需求。这种方法旨在减少条件操作的数量，尤其是嵌套条件语句，因为它们可能延长关键路径、降低基本计算单元的速度，并导致工作负载不平衡。然而，利用稀疏性必然涉及跳过不必要的操作。因此，一个关键挑战是如何在不引入嵌套条件语句的情况下利用时间与空间稀疏性。

在输入优先型流式架构中， [11], [12], 输入特征图迭代不进行稀疏性检测，因此无法实时利用时间稀疏性。引入即时检查以跳过值为零的输入，不仅会因通道级稀疏性变化造成工作负载不平衡，还需要为每条数据流配置控制逻辑；与脉动阵列等可重构设计相比，这会大幅增加硬件开销。因此，这类架构通常只在累加过程中利用时间稀疏性：输入脉冲控制累加器是否运行，从而使各处理单元之间的延迟保持均衡。尽管如此，输入特征图读取和权重选择仍会访问所有输入（包括零值）及其权重，导致冗余数据传输。因此，时间与空间稀疏性带来的综合收益仍受到严重限制，而试图全面整合稀疏性所需的开销又使这种方法难以实用。

另一种方案是采用权重优先算法。现有基于 GOAP [14] 的工作仍依赖于经过控制逻辑增强的脉动阵列架构，并利用非零权重元数据定位对应的输入。在路由器和调度器的协助下，GOAP 可仅将非零输入–权重对分派给处理单元，从而在不强制采用统一数据流的情况下实现均衡执行。

然而，将此类控制逻辑直接移植到流式架构中——为保持高并行度而在每条数据通路中复制一份——会带来诸多重大挑战：(1) 由于每条通路都要复制控制逻辑，硬件资源使用量和逻辑延迟都会急剧增加（不同于可重构阵列中的共享逻辑）；(2) 并行通路可能会争用内存访问，从而导致冲突或行为错误；(3) 由于输入具有实时性和不可预测性，各通路之间不同的稀疏模式会造成工作负载不均衡，进而导致延迟不平衡和同步问题（部分处理单元负载过重，而其他处理单元处于空闲状态）。

这些问题表明，将权重优先的稀疏性方法从可重构阵列迁移到流式架构并非易事。这需要精心设计数据流，在最大限度简化控制逻辑的同时，保持较高的并行度、平衡各处理单元的工作负载，并确保各层之间的流式数据流保持一致。

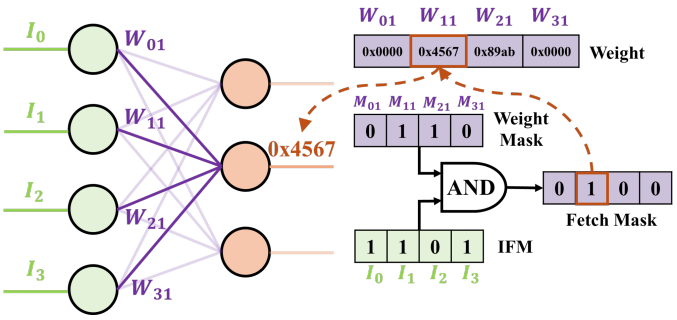

图2. 全连接（FC）层的权重掩码。

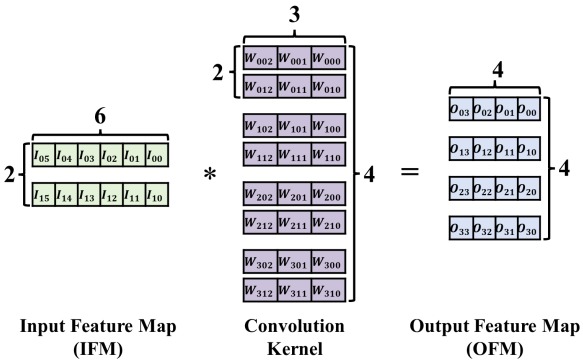

图3。卷积运算的简单示例

为满足这些要求，我们提出一种混合方案：预先计算并压缩非零权重（因为权重在推理时是固定的），然后在迭代过程中仅遍历非零权重；随后，在累加过程中利用时间稀疏性，将输入脉冲用作使能信号，以跳过不必要的运算。该设计结合了数据压缩与稀疏性感知跳过，同时保留了流式架构在简洁性和吞吐量方面的优势。

B. 全连接层中的稀疏性利用

脉冲神经网络主要由两种类型的层组成：全连接（FC）层和卷积层。全连接层执行矩阵-向量乘法和累加操作。为处理这些计算，采用了矩阵-向量阈值单元（MVTU）。每个MVTU包含多个处理单元（PE），每个处理单元包含若干条单指令多数据（SIMD）通道，以实现矩阵运算的并行处理。

在全连接层中，每个权重对应神经元之间的唯一连接，并且每个权重在每个时间步中只使用一次。为了优化计算，必须识别非零的输入-权重对。这种情况类似于逻辑“与”运算，只有当两个操作数都处于激活状态时才需要进行计算。

权重通常以定点格式（例如16位）存储。仅为判断权重是否为零而提取权重，会导致不必要的内存访问。鉴于卷积核权重在推理过程中保持不变，我们可以对其进行预处理，以识别非零值。一种有效的方法是实现权重掩码（WM），如图2所示。

在这种方法中，每个权重都与一个1位掩码相关联，用于指示其是否为非零值。与原始的16位权重存储相比，这种方法的存储开销仅为1/16。当神经元接收来自前一层的输入时，输入的二值特性使其能够直接充当掩码。通过对输入和权重掩码执行逻辑与运算，我们可以得到提取掩码（FM），用于确定需要提取和处理的权重。

与传统方法相比，传统方法会获取并处理与有效输入对应的所有权重，而该技术可显著降低计算负载。例如，在图 2 中，传统方法会根据 IFM 获取并处理三个权重，其中包括两个零值。相比之下，WM 方法根据 FM 仅获取并处理一个非零权重，从而同时减少内存访问和计算量。所需的额外硬件极少，仅涉及一个 4 位与门，实现起来十分简单。

随着网络复杂度的增加，这种设计的优势也愈发明显，可显著提升效率和性能。类似的策略已被应用于人工神经网络 [15] 中，以利用神经网络中的稀疏性，从而减少存储使用量和计算需求。

C. 卷积层中的稀疏性利用

与全连接层不同，全连接层中的每个权重在每个时间步仅使用一次，而卷积层由于卷积运算的特性，会在多个输入像素之间重复使用权重。虽然权重掩码技术仍可减少部分硬件资源利用率和不必要的数据读取，但由于逻辑与运算的重复执行，该技术在此场景下的效率较低。因此，我们需要针对卷积层开发一种高效策略，在不产生显著开销的情况下有效利用稀疏性。

为便于介绍后续算法，我们给出了一个卷积运算的简化示例，其维度如下所示，如图3所示。

- **输入特征图（IFM）：** (1, 6, 2)，分别表示高度（H）、宽度（W）和输入通道数（IC）。
- **卷积核：** (1, 3, 2, 4)，分别对应高度（H）、宽度（W）、输入通道数（IC）和输出通道数（OC）。
- **输出特征图（OFM）：** (1, 4, 4)，分别表示高度（H）、宽度（W）和输出通道数（OC）。

在此示例中，输入特征图（IFM）的尺寸是在填充后指定的，步幅设置为 1。输入特征图（IFM）、输出特征图（OFM）和卷积核的高度均设置为 1，这与射频输入信号中常见的长输入数组的特性相一致。

1) 门控一对全部乘积算法：基于此前的观察，我们发现，在推理过程中，卷积核权重保持不变，因此按照权重优先级进行读取效率更高，因为输入特征图中的数据通常规模较小。因此，利用固定的稀疏权重来

识别对应的输入特征图条目，并判断输入是否为零，以决定是否进行累加，是一种可行的解决方案。

此方法称为门控一对多乘积（GOAP）算法 [14]

表I 滑动窗口（SW）方法与门控一对多乘积（GOAP）方法在输入读取次数、权重读取次数和累加次数方面的比较。

方法	SW	GOAP
# 输入读取次数	$3 \times 2 \times 4 = 24$	$4 \times 1 \times 3 \times 4 = 48$
# 权重读取次数	$3 \times 2 \times 4 \times 4 = 96$	$3 \times 4 = 12$
# 累加	$(4+3+3+2) \times 4 = 48$	$(2+2+2) \times 4 = 24$

与滑动窗口（SW）方法不同，GOAP 方法仅关注卷积核中的非零权重。它会遍历这些非零权重，然后读取输入特征图。因此，输出数据流的生成方式也会相应改变。

2) SW 与 GOAP 方法的数据流比较： 根据加速器的配置，每个处理单元（PE）会自动从存储器中读取上一时间步的膜电位，并应用衰减因子（α），以模拟漏积分发放神经元的漏电特性。随后，输入特征图数据流被馈入每个处理单元，处理单元再从存储器中读取相应的突触权重。处理单元对输入进行加权累加；当达到预设阈值时，便判定神经元是否应当发放。若应发放，则将膜电位软复位。完成此过程后，当前时间步更新后的膜电位会写回存储器。MVTU模块集成了矩阵-向量乘法、累加和漏积分发放神经元模型仿真等功能，从而简化了计算流水线。

以图3中的卷积示例为例，图4展示了单个通道下 SW 和 GOAP 方法的工作原理，并突出了两者的差异。为公平比较，时间与空间稀疏性均设为 50%，并且为简化起见，假设所有四个卷积核具有相同的权重分布。

在图4(a)中，处理单元接收一个输入特征图流，其维度为$(H,\ W,\ IC) = (1,\ 3,\ 2)$。它根据索引读取相同尺寸的对应权重，并沿输入通道进行累加，以计算单个输出像素。所有输出通道并行执行此操作。随着滑动窗口移动，输出像素随之逐像素移动。由于滑动窗口每次移动时都要读取每个卷积核，因此每个输出通道共进行 $3 \times 2 \times 4 = 24$ 次权重读取，所有通道共进行96次权重读取。同样，对于四个相同尺寸的滑动窗口，输入共读取24次。由于所有输出通道并行使用输入特征图，因此总读取次数仍为24。关于累加，由于SW算法仅利用时间稀疏性——在输入特征图非零时进行累加——每个输出通道需要进行 $4+3+3+2 = 12$ 次累加。输入读取次数总计为，图3所示卷积运算的权重读取次数和累加次数汇总于表I。

相反，图4(b)展示了 GOAP 方法。例如，考虑一个位于坐标 (0，1) 处的非零权重 a，该坐标表示第一个输入通道和第二个水平位置。分析滑动窗口操作可知，只有第一个输入通道中位置 1–4 处的输入特征图值会与该权重 a 相乘。这个范围称为该权重 a 的使能图（ EM ）。因此，每个非零权重只读取一次，其使能图会读取四个对应的输入值。考虑到时间稀疏性，权重 a仅执行两次累加。同样，权重 b和 c 也各产生两次累加。因此，对于一个输出通道，GOAP算法执行 12 次输入读取、3 次权重读取和 6 次累加。所有输出通道的输入读取次数、权重读取次数和累加次数总计如表I所示。

比较这些结果可知，GOAP 方法读取输入数据的频率是 SW 方法的两倍。然而，它所需的权重读取次数仅为八分之一，执行的累加次数也只有一半。这种效率源于 GOAP 同时利用了时间与空间稀疏性，而 SW 仅依赖时间稀疏性。值得注意的是，在此设计中，每个输入的位长度为 1，而每个权重的位长度为 16。因此，SW 读取的数据总量为 1,560 位，而 GOAP 仅读取 240 位，仅为 SW 总量的 15.4%。再加上累加次数减少，GOAP 方法显著降低了数据读取和累加相关的运算与功耗。

3) 权重压缩格式：将权重稀疏性纳入神经网络设计，需要为稀疏数据选择合适的存储格式。常见格式包括坐标格式（COO）和压缩稀疏行格式（CSR）。这些格式均使用三个数组来存储非零值及其对应的元数据。

在 COO格式中，第一个数组存储非零值，第二个数组记录这些值的行索引，第三个数组则包含列索引。相反，在 CSR格式中，第一个数组存储非零值，第二个数组（称为索引指针）指示每一行在数据数组和索引数组中的起始位置，第三个数组存储非零值的列索引。

以图 5 中的示例为例，该图展示了一个维度为 $(H, W, IC, OC) = (1, 3, 2, 2)$ 的卷积核，分别以 COO 和 CSR 格式存储。虽然这些压缩格式是为二维矩阵设计的，但我们的卷积核有四个维度。尽管 $H = 1$ 在此示例中形成了三维张量，但将每个输出通道的压缩矩阵作为独立组件存储，会给硬件实现带来挑战。具体而言，细粒度内存分离较为困难，尤其是在不同输出通道的非零值数量不同时。硬件通常会根据最大组件分配空间，可能导致空间利用率低下。为解决这一问题，我们将输入通道和输出通道索引合并到行索引中，从而有效地将三维张量转换为二维矩阵。

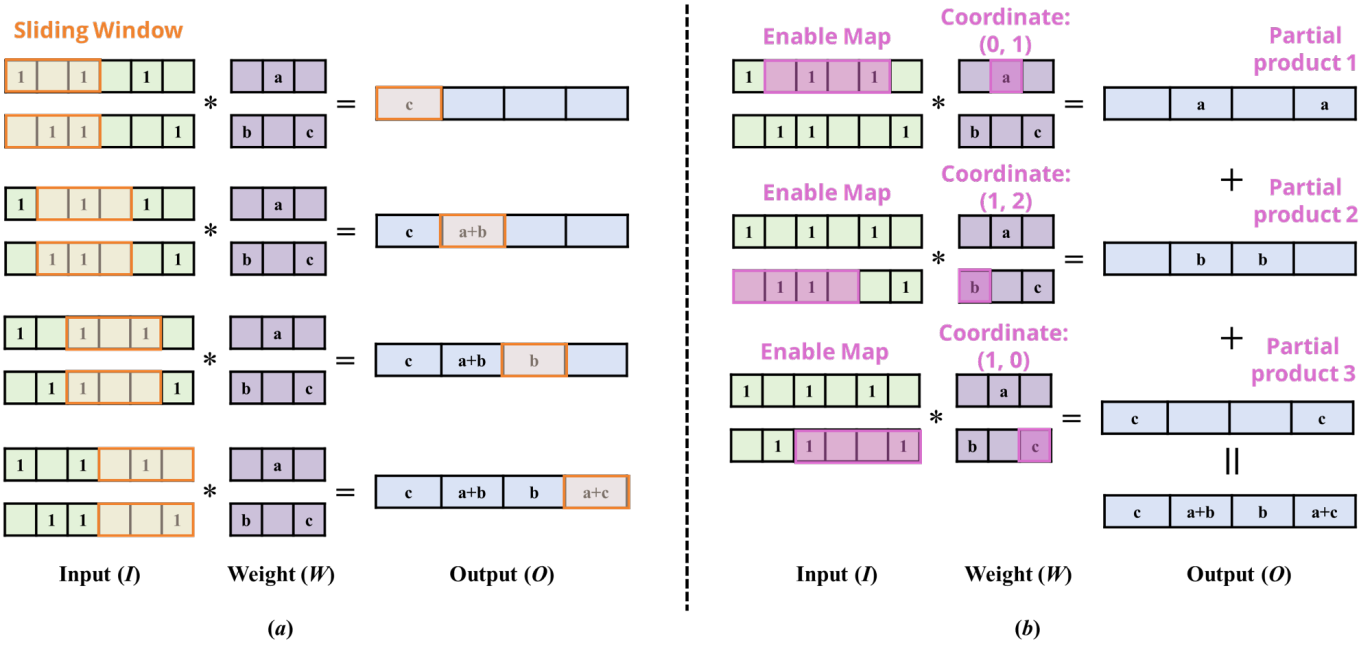

图4. 卷积中单个通道的 (a) 滑动窗口（SW）和 (b) 门控一对全部乘积（GOAP）算法的工作原理对比层

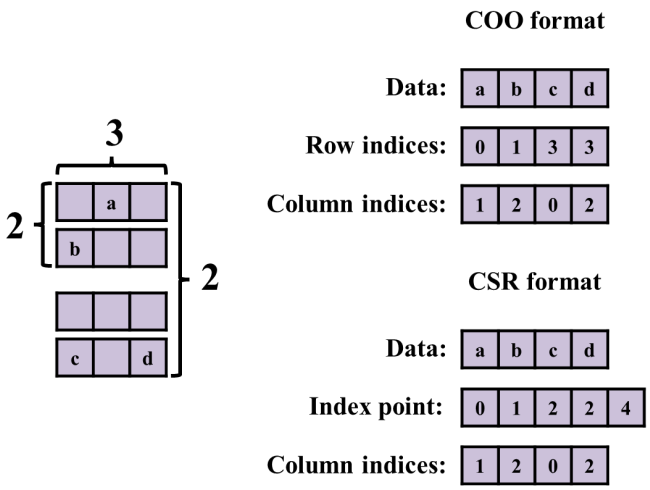

图5. 坐标格式（COO）和压缩稀疏行格式（CSR）中的稀疏矩阵存储示例。

$$IC_index = W.RI \ \% \ \#IC \tag{1}$$

$$OC_index = W.RI \ / \ \#IC \tag{2}$$

COO格式具有更高的灵活性，可直接访问行索引和列索引，而无需遍历相邻行，从而简化控制逻辑。因此，我们的设计采用 COO 格式。每个非零权重数据($W.D$)都对应一个行索引($W.RI$)和一个列索引($W.CI$)。输入通道索引（IC index）和输出通道索引（OC index）可通过以下方式计算：(1)(2)。列索引直接对应输入通道中非零权重的列索引。

为评估 COO 存储的开销，表II列出了 $W.D$、 $W.RI$ 和 $W.CI$ 的位宽，以及每个非零项的“总长度”。“数量”列表示采用稠密（100%）存储时每层权重的总数。

例如，在第 2 层中，卷积滤波器尺寸为 $1\times11\times16\times32$，在稠密存储下产生 5,632 个权重（每个 16 位）。在稀疏COO格式中，每个非零权重使用 9 位行索引和 4 位列索引进行编码，总计 29 位。因此，在稀疏性密度为 X时，所需存储空间为 $29\times5632\times X = 163328X\ bits$。将其与稠密存储进行比较可知，第 2 层的“盈亏平衡”密度约为 55.2%；当实际密度低于该阈值时，稀疏存储在位数方面更高效。其他层的类似计算结果汇总于表中。

然而，在实际应用中，片上存储器分配约束可能会削弱这些节省。对于我们的目标FPGA，BRAM字必须与受支持的宽度对齐（例如 1、2、4、9、18、36 位），且最小可分配单元为半个BRAM块（$18kbits$）。因此，稀疏权重数据通常需要填充或向上取整至最近的分配块。由于我们未采用定制的位打包或索引拼接（以避免增加解码逻辑和延迟），在中等密度下，稀疏性带来的理论存储节省可能会减少，甚至完全消失。

4）稀疏性感知输出通道数据流（*SAOCDS*）算法：基于对 GOAP 方法和压缩格式的数据流分析，我们可以推导出稀疏性感知输出通道数据流（SAOCDS）算法。数据流如图 6 所示。

在每个加速器层中，我们按以下步骤进行。首先，

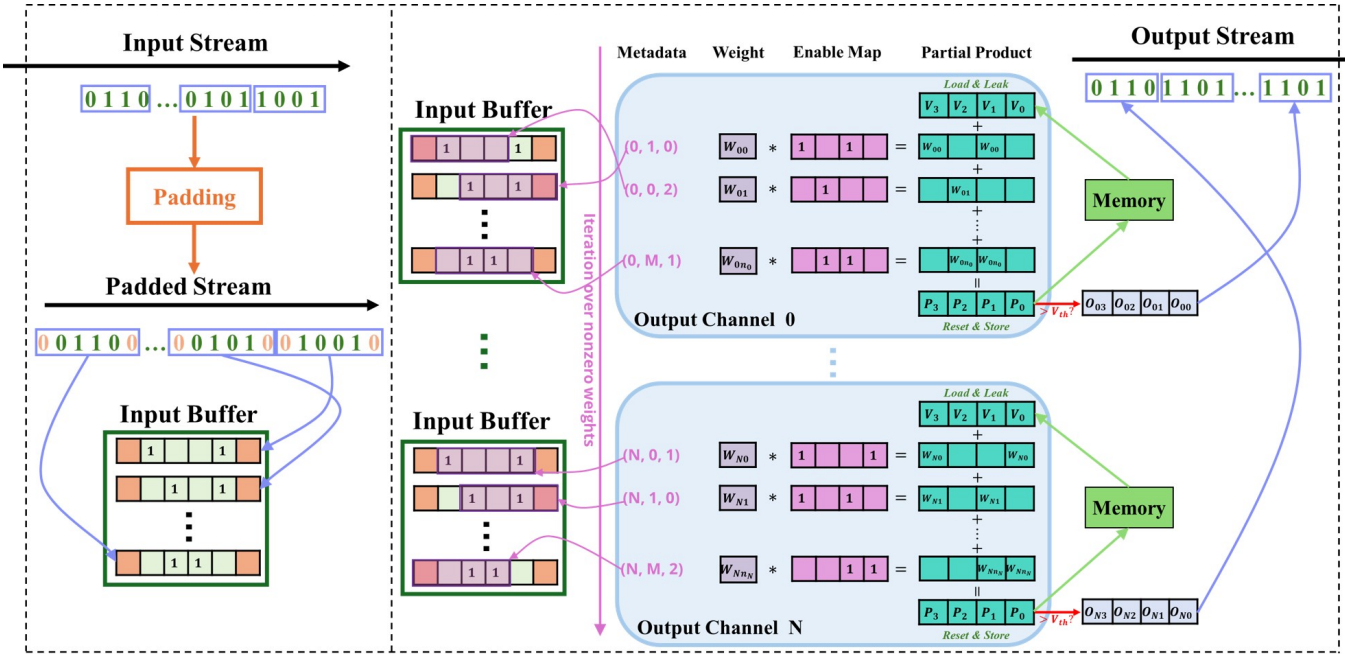

图6. SAOCDS加速器中单层的数据流

表II 三个卷积层的 COO 开销与稠密存储对比。

层 No.	W. D	W. RI	W. CI	总计长度	数量密度		稠密总计 Bit	COO 总计 Bit	盈亏-平衡密度
L1	16	5	4	25	352	X	5632	$8800X$	64.00%
L2	16	9	4	29	5632	X	90112	$163328X$	55.17%
L3	16	11	3	30	10240	X	163840	$307200X$	53.33%

我们按照输入通道顺序读取前一层的脉冲信号，然后对其进行填充并存入输入缓冲区，以便在后续迭代中重复使用（如图6左侧所示）。接着，我们按照输出通道顺序遍历所有非零权重，利用以COO格式存储的权重数据及其元数据进行累加。最后，我们计算膜电位并生成输出脉冲（如图6右侧所示）。详细的迭代步骤和变量计算见算法1。

对于每个时间步，我们首先初始化迭代变量（第 2–5 行）。然后遍历以压缩格式存储的非零权重（第 6 行）。对于每个权重，我们解码其对应的输出通道索引 oc （第 7 行）、下一个输出通道的索引 $next_oc$ （第 8 行）以及输入通道索引 ic （第 9 行）。如果当前输入通道索引小于输入通道总数，我们就从输入缓冲区读取新输入通道的数据（第 10–13 行）。如果当前 oc 与前一个不同，则表示出现了新的输出通道，我们必须读取该输出通道对应的新神经元的膜电位（状态）（第 14–17 行）。接下来，我们遍历该权重使能图中的输出像素 oi。如果对应值为 1，则将部分乘积累加到神经元的膜电位中；否则，跳过累加（第 18–22 行）。请注意，使能图的维度与单个输出通道相同，且每个非零权重都会产生一个部分乘积。处理完一个非零权重后，如果$next\ oc$ 不等于 oc，则表示当前输出通道的累加阶段结束。此时，将神经元的膜电位与其阈值进行比较，以确定是否发放脉冲。如果发放，则向该像素输出一个脉冲，并执行软复位。随后将更新后的膜电位写回存储器（第 23–26 行）。最后，更新前一个输出通道索引$pre\ OC$ （第 27 行），至此完成一次非零权重迭代（第 6–28 行）。处理完当前时间步的所有非零权重后，该时间步的计算即告结束，随后进入下一个时间步。

通过遍历非零权重，我们可以利用空间稀疏性，而无需额外的控制逻辑或实时解码器来获取元数据。利用输入信号门控累加，使我们能够通过跳过不必要的计算来利用时间稀疏性。此外，采用输出通道数据流——即每层按照输出通道索引的固定顺序，每次传输一个输出通道——可使前一层的输出与下一层的输入直接连接，无需任何额外的控制逻辑。此外，由于每个非零权重的累加维度与输出通道的维度（也就是使能图的维度）相匹配，因此工作负载天然均衡。而且，神经元状态仅在输出通道切换时加载和存储，从而消除了同一输出通道内频繁的内存访问。总之，我们的流式数据流融合了时间与空间稀疏性，无需额外的控制逻辑，并确保每个非零权重对应的工作负载分布均衡。

算法1 稀疏性感知输出通道
数据流流式处理（SAOCDS）脉冲神经网络加速器，适用于一个卷积层。

输入： 输入特征图按输入通道顺序依次流入。
输出： 输出特征图按输出通道顺序依次生成，输出通道-通道，传递到下一层。

对于 $t = 0$ 到 $T - 1$ **执行**
  $IC_read \leftarrow 0$
  $pre_oc \leftarrow OC$
  $oc \leftarrow 0$
  $next_oc \leftarrow 0$
  对于 $nnz = 0$ 到 $NNZ - 1$ **执行**
    $oc \leftarrow \left\lfloor \frac{W[nnz].RI}{IC} \right\rfloor$
    $next_oc \leftarrow \left\lfloor \frac{W[nnz+1].RI}{IC} \right\rfloor$
    $ic \leftarrow W[nnz].RI\%IC$
    如果 $IC_read < IC$ **则**
      输入 $I[ic]$
      $IC_read \leftarrow IC_read + 1$
    结束条件判断
    如果 $oc \neq pre_oc$ **则**
      加载 $V[oc]$
      衰减 $V[oc]$
    结束条件判断
    对于 $oi = 0$ 到 $OI - 1$ **执行**
      如果 $I[ic][oi + W[nnz].CI] == 1$ **则**
        $V[oc][oi] \leftarrow V[oc][oi] + W[nnz].D$
      结束如果
    结束循环
    如果 $next_oc \neq oc$ **则**
      输出 $O[oc]$
      存储 $V[oc]$
    结束条件判断
    $pre_oc \leftarrow oc$
  结束循环
结束循环

D. 稀疏性补充算法

将稀疏性纳入我们的设计可提高计算效率，但也会在迭代执行过程中带来潜在的逻辑挑战。稀疏度较高时可能会引发一些特定问题，需要谨慎处理。

1) 空迭代： 在我们的架构中，前一层按数据通道逐个输出数据。然而，对于某些输入通道，卷积核可能没有非零权重。例如，考虑一个仅在输入通道 0 和 2 上具有非零权重的卷积核。在第一次迭代中，系统读取输入通道 0 的数据，由于存在相应的非零权重，计算会按预期进行。在第二次迭代中，系统读取输入通道 1 的数据，但卷积核在该通道上没有非零权重。此外，非零权重对应输入通道 2，而该通道的数据尚未更新或不可用（在算法 1 中，$IC\ read = 1$ while $ic = 2$）。此时尝试进行计算可能会导致逻辑错误。为避免这种情况，我们将此类迭代指定为空迭代，在此期间不执行任何计算。系统会在必要的输入数据可用后恢复

算法 2 SAOCDS算法 以及补充
单个卷积层的稀疏性处理机制。

输入： 按输入通道依次传输输入特征图数据流。
输出： 按输出通道依次生成输出特征图，并传输至下一层。

$REPS \leftarrow NNZ$ +#额外 I +#空 L
对 $t = 0$ 至 $T - 1$ **执行**
  $IC_read \leftarrow 0$
  $pre_oc \leftarrow OC$
  $oc \leftarrow 0$
  $nnz \leftarrow 0$
  对 $reps = 0$ 至 $REPS - 1$ **执行**
    $nnz_oc \leftarrow \left\lfloor \frac{W[nnz].RI}{IC} \right\rfloor$
    $nnz_next_oc \leftarrow \left\lfloor \frac{W[nnz+1].RI}{IC} \right\rfloor$
    如果 $IC_read < IC$ **则**
      输入 $I[ic]$
      $IC_read \leftarrow IC_read + 1$
    结束如果
    如果 $oc \neq nnz_oc$ **则**
      加载 $V[oc]$
      衰减 $V[oc]$
      输出 $O[oc]$
      存储 $V[oc]$
      $oc \leftarrow oc + 1$
    else
      $ic \leftarrow W[nnz].RI\%IC$
      如果 $ic < IC_read$ **那么**
        如果 $oc \neq pre_oc$ **则**
          加载 $V[oc]$
          衰减 $V[oc]$
        结束如果
        对于 $oi = 0$ 到 $OI - 1$ **执行**
          如果 $I[ic][oi + W[nnz].CI] == 1$ **则**
            $V[oc][oi] \leftarrow V[oc][oi] + W[nnz].D$
          结束条件判断
        结束循环
        如果 $nnz_next_oc \neq oc$ **，则**
          输出 $O[oc]$
          存储 $V[oc]$
          $pre_oc \leftarrow oc$
          $oc \leftarrow oc + 1$
        else
          $pre_oc \leftarrow oc$
        结束条件判断
        $nnz \leftarrow nnz + 1$
      结束 if
    结束 if
  结束 for
结束 for

操作（算法2，第 22 行的条件语句）。

请注意，空迭代仅发生在计算第一个输出通道时。对于后续输出通道，会重用所有通道的输入数据，从而避免这一问题。

2）额外迭代：我们的算法遍历卷积核中的非零值。然而，较高的稀疏性可能导致某些输出通道没有非零权重，尤其是在输出通道维度较小时。在算法1中，从一个输出通道切换到另一个输出通道时，当前输出通道的状态会写回状态变量数组，随后加载并衰减下一个输出通道的状态，然后进行累加。如果某个输出通道没有非零权重，该通道就不会被处理，其状态也不会衰减，从而导致不一致。

为解决这一问题，我们引入“额外迭代”，以更新此类通道的状态和输出值。必须将对已存储非零权重的遍历与流式加速器内部的迭代相互解耦。修改后的算法如下（算法 2，第 14–19 行）：

- 设当前输出通道为 oc，当前非零权重所属的输出通道为 $nnz\ oc$。
- 如果 $nnz\ oc = oc$，则执行标准迭代。
- 如果 $nnz\ oc \neq oc$，则对输出通道 oc 执行额外迭代：– 加载该通道的状态。– 对状态应用衰减。– 写回更新后的状态。– 输出中间通道的值。– $oc = oc + 1$

这种方法确保所有输出通道无论稀疏性如何，其状态都能得到适当更新，并生成输出，从而保持整个网络的一致性。

需要特别指出的是，无论需要多少次空迭代或额外迭代，前一层始终按通道输出特征图。鉴于卷积核的分布固定，我们能够预测空迭代或额外迭代的发生情况及其位置，从而预先计算迭代总次数（算法 2）。

包含补充稀疏性处理机制的完整算法见算法 2。空迭代与额外迭代可确保在极端稀疏情况下数据流正确，同时开销极小。当稀疏性低于 90% 时，这些迭代的次数少于十次；稀疏性为 95% 时，它们占所有迭代次数的比例不到 10%，且迭代总次数降至完全密集情况下的大约 5%。总之，我们以几乎可以忽略不计的开销解决了稀疏性可能引发的问题。

IV. AMC系统与SNN模型

本节介绍用于构建 AMC系统 的已训练脉冲神经网络模型，报告其分类结果，并描述后续的模型压缩和稀疏性增强过程。

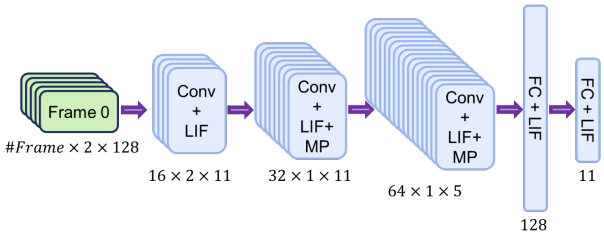

图 7. 脉冲神经网络分类器的架构。图中使用的缩写：$Conv$ 表示卷积层，MP 表示最大池化层，FC 表示全连接层，LIF 表示漏电积分发放神经元模型。

A. 数据集

我们使用 RadioML 2016.10A 数据集 [13],这一自动调制分类研究中广泛采用的基准 [13]，[16]，[17]，验证 AMC 系统的功能。该合成数据集由 GNU Radio 生成，涵盖 11 种调制方案——包括 8 种数字类型和 3 种模拟类型——信噪比范围为 -20 dB 至 18 dB，间隔为 2 dB。每个样本由一个 2×128 矩阵构成，表示射频信号的同相（I）和正交（Q）分量。

为使这些样本适用于脉冲神经网络处理，我们采用 [12] 中所述的基于Sigma-Delta的转换方案。该方法对原始浮点IQ数据进行过采样并应用低通滤波器，得到维度为（2, 128×OSR）的二进制输出，其中OSR表示过采样率。随后，将二进制数据重塑为（2，128，OSR）张量，使SNN能够在每个时间步处理一帧数据，输入尺寸为（2，128），共处理T = OSR个时间步。

*B. SNN*模型与软件分类结果

$$\begin{cases} U_t = \alpha \cdot U_{t-1} + W \cdot I - \theta \cdot S_{t-1} \\ \quad \begin{cases} S_t = 1, & if \quad U_t > U_{th0} \\ S_t = 0, & if \quad U_t \leq U_{th0} \end{cases} \end{cases} \tag{3}$$

SNN分类器模型的架构如图 7所示，并使用 PyTorch框架实现。详细的仿真设置见 [12]。神经元动力学采用漏电积分发放（LIF）模型，如公式3所定义，其中 U_t 表示时间步 t时的膜电位，α 为衰减因子，θ 为软复位因子，W 为权重矩阵，S 表示脉冲函数，U_{th0} 为神经元的发放阈值。在每个时间步，膜电位按因子 α成比例衰减。如果上一个时间步产生了脉冲，则通过减去 θ对膜电位进行软复位。随后，膜电位累加IFM与卷积核的乘积，以确定当前时间步是否发放脉冲。为确保在现场可编程门阵列上的硬件实现准确无误，将 α、θ和 U_{th0}视为每个神经元的可训练参数。

图 8 展示了我们的模型在不同信噪比下的分类准确率。当信噪比低于 0 dB 时，准确率相对较低，平均准确率约为

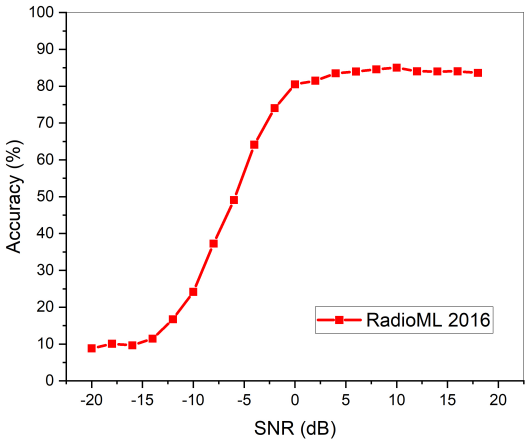

图 8. RadioML 2016 上不同信噪比（SNR）下的分类准确率

57%。然而，当信噪比高于 0 dB 时，模型的准确率始终超过 80%，峰值约为 85%。这些结果与先前针对同一数据集的研究结果相当 [17], [13], [18]。

C. 剪枝与量化

图 9 展示了原始训练模型所有层中权重绝对值的分布。为优化模型的硬件实现，我们采用剪枝和量化技术来减小模型规模并降低计算复杂度。

1) 剪枝：剪枝是模型压缩的一种有效技术。我们采用细粒度剪枝 [19],，将单个权重置为零，同时不改变网络架构。这种方法在减少活跃参数数量的同时，保留了模型的结构。

分析图 9 可知，除第一层外，所有层的权重都显著集中在零附近。这种分布表明，即使不造成明显的准确率损失，也具有很大的剪枝潜力。第一层在零附近的峰值不那么明显，这可能是因为其权重数量有限。

为在压缩模型的同时保持分类准确率，我们采用*L1*非结构化剪枝，移除绝对值最小的权重。具体而言，我们采用为期100个训练轮次的三阶段训练计划：最初20%的训练轮次用于学习基础特征；随后60%的训练轮次通过迭代剪除不太重要的权重来减小模型规模；最后20%的训练轮次则用于微调，以恢复剪枝可能导致的性能下降。这种方法符合模型压缩领域的成熟实践，即在训练过程中融入剪枝，以平衡效率与准确率。

2）量化：在 PyTorch 中，模型权重以 32位浮点数 存储，其精度高于推理所需。为优化 FPGA部署，我们将这些权重量化为16位定点数。为防止准确率下降，我们在训练过程中采用学习步长量化（LSQ）——这是一种量化感知方法，将量化步长视为可训练参数，并通过使用直通估计器的反向传播进行优化。

使用LSQ训练时，会在前向传播和反向传播过程中模拟16位量化，同时保留全精度权重用于梯度更新。收敛后，我们应用学习到的步长：对每个权重进行缩放、裁剪至16位范围，并舍入为相应的定点表示。随后，将最终量化后的权重存储在现场可编程门阵列上，并在推理过程中使用整数运算。这种方法在降低模型大小、简化硬件实现的同时，能够保持模型准确率。详细结果见第五节。

V. FPGA 实现结果

该硬件架构使用 Xilinx Vitis 高层次综合（HLS）工具开发，并通过 Vivado Design Suite 实现。我们在 Xilinx Virtex-7 FPGA VC709 平台上实现了图 7 所示的五层脉冲神经网络分类器。为便于高效部署硬件，原始 PyTorch 模型中的所有浮点权重均被量化为 16 位定点表示。此外，还通过细粒度剪枝引入了权重稀疏性。

A. 累加效率分析

表 III 不同空间稀疏度级别下的累加次数比

空间稀疏性	累加次数比			
	第 1 层	第 2 层	第 3 层	第 4 层
0%	100.00%	100.00%	100.00%	100.00%
10%	89.70%	89.73%	89.74%	90.00%
20%	79.83%	79.96%	79.95%	79.96%
30%	69.80%	69.65%	70.02%	70.13%
40%	59.87%	59.92%	59.77%	59.85%
50%	49.85%	49.91%	49.80%	50.10%
60%	39.74%	39.03%	40.19%	40.14%
70%	29.80%	30.39%	29.47%	29.97%
80%	20.01%	19.72%	20.02%	20.07%
90%	9.89%	9.44%	10.14%	9.79%

表 III 展示了与基于 FINN 的基准设计 [12] 相比，我们的稀疏性感知设计在减少累加操作方面的有效性。在输入相同的情况下，随着空间稀疏性从 0% 增至 90%，第 1 至第 4 层的累加次数比例均持续且显著地降低。

这一趋势表明，与仅利用时间稀疏性相比，同时利用时间与空间稀疏性能够显著提升累加效率。值得注意的是，累加操作的减少比例与空间稀疏性程度密切吻合，凸显了空间稀疏性在提升 SNN 推理效率方面的关键作用。第 5 层

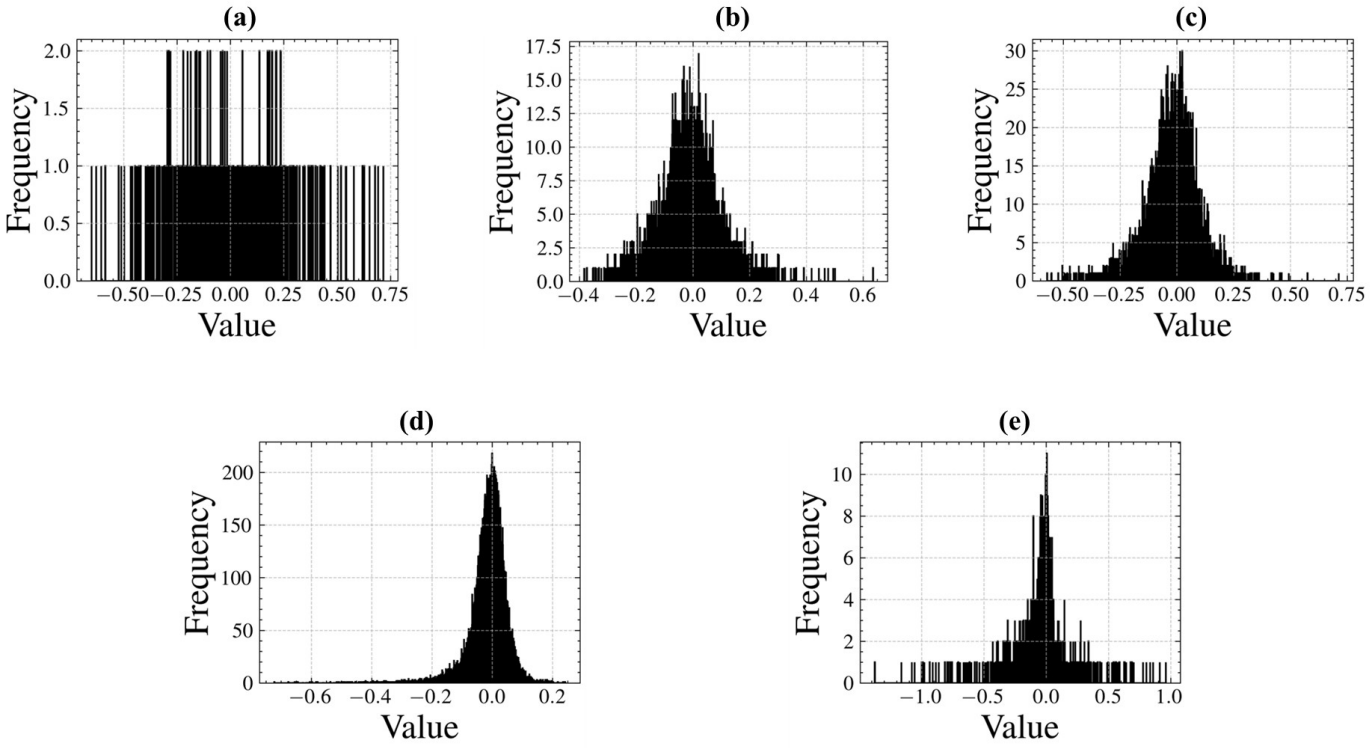

图9. 各层的权重值分布。(a)-(e)分别表示第1至第5层的权重值分布。在每个直方图中，“频率”表示落在各数值范围区间内的权重数量。

由于其维度有限，累加次数的变化较大，结果的可靠性也因此受到影响，因此未纳入本次分析。总体而言，这些结果凸显了将稀疏性感知机制融入 SNN 硬件加速器所带来的显著优势。

B. 与已有工作的比较

$$FoM = \#LUT * \frac{Dynamic\ Power}{Throughput} \tag{4}$$

表 IV 展示了 RadioML 2016 加速器的实现结果。硬件利用率通过查找表 (LUT)、触发器 (FF)、块 RAM (BRAM) 和数字信号处理单元 (DSP) 的数量来表征。

表 IV 报告了动态功耗和静态功耗，以表征逻辑活动以及基线 FPGA 的功耗。动态功耗通过工具 Vivado Design Suite，利用加速器设计的周期精确仿真生成的 SAIF（开关活动交换格式）文件进行估算，从而确保基于实际开关活动进行准确计算。电压/温度角分析假设采用商用级器件、最大工艺变化、25 °C 环境温度和标准电源电压。

吞吐量以在最大工作频率（F_{max}）下每秒处理的输入 I/Q 采样数量（ S/s ）来衡量，其中每个数据集帧包含 128 个 I/Q 采样对。为公平比较不同设计的功耗效率，我们使用测得的动态功耗和吞吐量计算每个采样的能耗。为将硬件资源利用率纳入评估，我们定义了性能指标（FoM）（见公式 4）；该指标同时偏好较低的查找表用量和动态功耗，以及较高的吞吐量。因此，较小的性能指标表示加速器在整体性能、功耗和资源使用量方面更高效。

在 FPGA 资源指标中，我们采用查找表利用率作为衡量逻辑资源消耗的主要指标。之所以这样选择，是因为查找表（以及与之关联的寄存器）构成了 FPGA 中的可配置逻辑结构——它们可实现任意组合逻辑和时序逻辑，是芯片上定制逻辑的基本构建模块。相比之下，触发器主要反映存储或流水线寄存器，而非逻辑复杂度；块 RAM 主要用于衡量存储容量，而非逻辑运算；在我们的设计中，DSP 模块仅用于辅助神经元状态更新（例如膜电位衰减），并不用于核心累加路径。因此，查找表数量最直接地反映了定制逻辑的规模，是我们进行比较时最具代表性的逻辑资源消耗指标。

关于分类准确率，需要注意的是，RadioML 2016 数据集上的平均准确率通常较低 [13],[16], [12] （约为 57%），尤其是在低信噪比条件下。在 [17],中，报告的准确率是在高信噪比下达到的最高值。相比之下， [12] 中报告的结果以及我们的设计结果，均是在量化和剪枝后于高信噪比下获得的。

对于提出的4层人工神经网络模型，采用了 [20], 16位量化。然而，硬件利用率，尤其是

表 IV 各种以往加速器设计的资源利用率、性能和准确率比较。

设计	模型	平台	量子比特	#查找表	#FF	#块随机存取存储器	#DSP	动态功耗 (W)	静态功耗 (W)	吞吐量 (兆样本/秒)	每样本能耗 (纳焦/秒)	Fmax (兆赫)	准确率	延迟 (μs)	FoM (皮焦耳/秒)
[20]	4-ANN	ZCU104	16 位	89512	57726	N/A	1116	**0.254**	**0.593**	4.78	53.14	70	N/A	N/A	4756.4
[17]	TFM	AXU3EG	16 位	**20060**	**11511**	167	**115**	0.6	N/A	N/A	N/A	N/A	低	N/A	N/A
[17]	TFM	AXU3EG	32 位	47629	36239	251	261	0.6	N/A	0.704	852.27	N/A	**0.86**	500	40592.9
[12]	5-脉冲神经网络	Virtex 709	16 位	74578	54289	**63.5**	118	1.146	1.063	11.45	100.09	**137**	**0.86**	**454.85**	7464.3
SAOCDS 100	5-脉冲神经网络	Virtex 709	16 位	82859	44906	96.5	297	0.473	1.038	**23.5**	**20.13**	**137**	**0.86**	3246.42	**1667.8**
SAOCDS 50	5-脉冲神经网络	Virtex 709	16 位	84467	44815	85.5	297	0.493	1.039	**23.5**	20.98	**137**	0.85	1640.98	1772
SAOCDS 15	5-脉冲神经网络	Virtex 709	16 位	85671	44991	76.5	297	0.552	1.040	**23.5**	23.49	**137**	0.85	497.94	2012.4

查找表、触发器和数字信号处理器的利用率仍然相对较高。尽管专门使用数字信号处理器能够在现场可编程门阵列上高效执行乘加运算，并有助于实现相对较低的动态功耗，但数字信号处理器也是一种稀缺且宝贵的资源。由于吞吐量较低，每个采样的能耗仍然较高，表明功耗效率不高。因此，尽管报告的性能指标看起来颇具优势，但其忽略了数字信号处理器的大量使用。

来自 [17]的两种基于 Transformer 的实现无需量化即可达到较高的仿真准确率。然而，应用 16位量化后，其准确率显著下降，使得有效的调制识别难以实现，尽管延迟有所改善。虽然这些设计的硬件利用率相对较低，但由于 Transformer 的计算复杂度较高，尤其是在缺少量化和剪枝的情况下，其吞吐量受到严重限制。因此，其性能指标值和每样本能耗的竞争力较弱。值得一提的是，前三种加速器均未采用流式架构，因此吞吐量相对较低。

来自 [12] 的先前流式架构采用基于 FINN 的流式方法，实现了相对较高的吞吐量和更优的性能指标。请注意，在我们的实验中，硬件利用率与 [12]略有不同，因为我们采用了更大的部分乘积位宽，以增强抗溢出能力。此外，通过使用数字信号处理器实现节点特定衰减因子，确保了板载推理准确率，而累加操作则无需数字信号处理器。得益于较高的并行度，该设计也保持了较低的延迟。

需要注意的是，在流式架构中，内存带宽很少会成为瓶颈。中间激活通过片上先进先出队列在各层之间传递，权重则存储在片上或通过综合生成，因此片外流量极少。片上访问量不大，因为每个本地存储块都能独立处理低速率访问。因此，片外和片上存储器都不需要达到脉动阵列所需的高带宽，所以流式架构通常不会报告内存带宽。

在所有层的均匀密度分别为 15%、50% 和 100% 的 SAOCDS 设计中，该架构能够灵活处理输入特征图和卷积核中的稀疏性。值得注意的是，即使在密度为 100%（卷积核不存在稀疏性）的情况下，SAOCDS-100 消耗的动态功耗也仅为 [12] 中设计的 41%。这得益于数据读取和存储需求的减少。该架构的流式特性有助于提升吞吐量，并显著改善性能指标。

然而，与基于 FINN 的流式架构相比，SAOCDS-100 的延迟更高，主要原因是它需要遍历非零权重。随着稀疏性增加，延迟显著降低，同时仍保持出色的性能指标。在稀疏性为 85% 时，我们的设计实现了与 [12] 相当的延迟，而功耗仅为其 48.2%。

总而言之，与采用 RadioML 2016 数据集的先前自动调制分类研究相比，我们的 SAOCDS 设计有效利用了输入特征图和卷积核中的稀疏性，始终实现最高吞吐量，并以硬件资源仅略有增加为代价，达到了较低的功耗。使用稠密卷积核时，延迟较高，但会随卷积核密度成比例变化，并随着稀疏性增加而显著降低。在空间稀疏性极高时，SAOCDS 在保持最高吞吐量、具竞争力的分类准确率和低功耗的同时，实现了极低延迟。

C. 不同权重密度下的详细比较

比较不同空间密度水平下的性能，可以更深入地了解我们加速器的效率。我们纳入了基于 FINN 的基准设计 [12]，该设计采用相同的脉冲神经网络架构和输入特征图数据流，以便进行直接比较。表V总结了结果：每个条目都注明了加速器及其逐层密度。单个数值表示全部五层的密度均匀；一组数值（例如“25-20-15-20-25”）则指定逐层权重密度。在训练期间，我们采用层特定剪枝率来达到这些目标密度，并确保部署时每一层都达到指定的密度。

1) 资源利用率分析：SAOCDS 设计在不同空间稀疏度下的查找表和触发器用量保持稳定，而块随机存取存储器用量则随稀疏性增加而减少。这是因为逻辑资源经过综合后用于支持单次迭代，而稀疏性只会影响迭代次数，不会影响每次迭代所需的资源。稀疏性较高时需要存储的非零权重更少，因此块随机存取存储器的消耗也相应减少。

此外，SAOCDS由于在处理潜在衰减操作时具有更高的并行度，因此使用了更多数字信号处理器。具体而言，它在所有层中执行了297次衰减操作，而基线为118次，这证实了数字信号处理器资源会随着计算并行度的提升而按比例增加。

表V 不同空间稀疏度级别下SAOCDS的资源利用率、性能和准确率比较。

设计	#LUT	#FF	#块随机存取存储器	#DSP	动态功耗 (W)	静态功耗 (W)	吞吐量 (MS/s)	每样本能耗 (nJ/S)	Fmax (MHz)	延迟 (μS)	准确率 (%)	FoM (μJ/S)
[12] 100	**74578**	54289	63.5	**118**	1.146	1.063	11.45	100.09	137	454.85	**100**	7464.3
SAOCDS 100	82859	44906	96.5	297	0.473	1.038	**23.5**	20.13	137	3246.42	**100**	1667.8
SAOCDS 75	83322	44902	89.5	297	0.432	1.036	**23.5**	18.38	137	2460.18	99.98	1531.7
SAOCDS 50	84467	**44815**	85.5	297	0.493	1.039	**23.5**	20.98	137	1640.98	99.51	1772.0
SAOCDS 25	83868	44901	79.0	297	0.481	1.038	**23.5**	20.47	137	822.10	99.22	1716.6
SAOCDS 20	83514	44915	78.5	297	0.541	1.040	**23.5**	23.02	137	658.90	99.17	1922.6
SAOCDS 15	85671	44991	76.5	297	0.552	1.040	**23.5**	23.49	137	497.94	97.64	2012.4
SAOCDS 10	84175	44945	**75.0**	297	0.473	1.037	**23.5**	20.13	137	**453.14**	93.33	1694.3
SAOCDS 5	83572	45060	75.5	297	**0.361**	**1.033**	**23.5**	**15.36**	137	**453.14**	73.19	**1283.8**
SAOCDS 25-20-15-20-25	84808	44969	77.0	297	0.615	1.043	**23.5**	26.17	137	500.82	98.19	2219.4
SAOCDS 20-15-10-15-20	84297	45011	76.5	297	0.579	1.041	**23.5**	24.64	137	**453.14**	95.68	2076.9

2）功耗、延迟与吞吐量分析：SAOCDS在大多数稀疏度下的功耗相对稳定，但有两个显著例外：均匀密度为5%时功耗骤降，而混合逐层密度下功耗上升。这是因为稀疏性只影响迭代次数，不影响每次迭代的资源使用量。现场可编程门阵列的静态功耗在所有设计中几乎保持不变（1.033–1.043 W），这意味着动态功耗的变化主要反映了逻辑复杂度和资源使用量。

延迟大致随密度的降低而成比例下降，这与基于 GOAP 的卷积设计的表现一致。由于第三个卷积层的迭代次数最多，其延迟主导了流式加速器的总延迟。然而，在稀疏度极高时（例如 5%），瓶颈转移至全连接层，延迟因此趋于平稳。与 GOAP 不同，全连接层采用的 WM 方法会跳过冗余操作以降低功耗，但无法降低延迟。这会导致流水线停顿，最终降低整体功耗。

在混合密度设计中，逐层工作负载的平衡性更好，因此可以让更多逻辑单元同时运行。由于不会发生阻塞，卷积层仍是瓶颈，平均利用率提高，功耗也随之增加。

由于流式架构可确保完全流水线化的并行执行，因此所有稀疏度下的吞吐量都保持一致。由于关键路径不受稀疏性影响，吞吐量保持稳定，并且高于 [12] ，这是因为最慢流水线阶段的实现效率更高。

3）准确率分析：在分类准确率对比中，我们将压缩模型与其对应的原始 PyTorch 模型进行比较，而非与真实标签进行比较。

在训练期间进行联合剪枝与量化时，适度的稀疏性对分类准确率影响甚微。即使均匀密度为 10%，模型仍能保持较高的准确率，但密度进一步降低时，性能会显著下降。采用逐层密度后，可以在较大的层中实现更高的稀疏性，从而有助于平衡工作负载、减少延迟瓶颈并保持准确率。例如，与基线 [12],相比，SAOCDS 25-20-15-20-25 配置以不到 2% 的准确率损失和仅为基线 53.7% 的功耗，实现了接近最优的延迟，同时保持了相近的资源使用量和2× 吞吐量——在所评估的设计中，这代表了最佳的整体权衡。

VI. 结论

本文提出了一种稀疏性感知输出通道数据流流式推理脉冲神经网络加速器，针对高吞吐量、低功耗的实时应用进行了优化，例如认知无线电系统中的自动调制分类。据我们所知，这是首个同时集成时间与空间稀疏性的全流水线流式脉冲神经网络架构。该加速器仅遍历卷积核中的非零权重，并且仅在相应输入非零时执行累加，从而消除了不必要的权重读取和累加操作。通过这种方式，我们的加速器实现了处理单元之间的自动负载均衡，并能在硬件资源受限的情况下支持高吞吐量。

推理期间卷积核的固定特性使得可以预先计算潜在的空迭代与额外迭代，从而实现层间无缝流式处理，避免了传统脉动阵列加速器通常存在的路由器与调度器开销。该方案在现场可编程门阵列上实现后，所提出的 SAOCDS 加速器实现了 23.5 MS/s 的吞吐量，在 RadioML 2016 数据集上的表现优于此前报告的加速器。

与基于 FINN 的流式脉冲神经网络加速器基线相比，所提出的设计实现了 2× 更高的吞吐量；在卷积核密度为 100% 时，其动态功耗仅为基线的 41.3%，而逻辑硬件利用率仅增加了 11.1%。随着卷积核稀疏性增加，该加速器仍能保持较高的吞吐量和较低的动态功耗，且延迟显著降低——在空间稀疏性较高时甚至低于基线。此外，在广泛的稀疏度范围内，分类准确率始终相对较高，展现了该设计的鲁棒性和效率。

参考文献

[1] X. Hao、Z. Feng、S. Yang、M. Wang 和 L. Jiao，《通过元学习实现自动调制分类》，*IEEE* 物联网期刊，第 10 卷，第 14 期，第 12276–12292 页，2023 年。

[2] T. Huynh-The、Q.-V. Pham、T.-V. Nguyen、T. T. Nguyen、R. Ruby、M. Zeng 和 D.-S. Kim，《自动调制分类：深度架构综述》，*IEEE Access*，第 9 卷，第 142950–142971 页，2021 年。[3] F. Meng、P. Chen、L. Wu 和 X. Wang，《自动调制分类：一种深度学习方法》，*IEEE* 车辆技术汇刊，第 67 卷，第 11 期，第 10760–10772 页，2018 年。[4] S. Tridgell、D. Boland、P. H. Leong 和 S. Siddhartha，《实时自动调制分类》，载于*2019*年国际现场可编程技术会议（*ICFPT*），第 299–302 页，IEEE，2019 年。[5] Y. Wang、L. Guo、Y. Zhao、J. Yang、B. Adebisi、H. Gacanin 和 G. Gui，《边缘设备上的自动调制分类分布式学习》，*IEEE* 无线通信快报，第 9 卷，第 12 期，第 2177–2181 页，2020 年。[6] F. Restuccia 和 T. Melodia，《大数据变小：通过射频环路中的深度学习实现实时频谱驱动嵌入式无线网络》，载于*IEEE INFOCOM 2019—IEEE* 计算机通信会议，第 2152–2160 页，IEEE，2019 年。[7] Z. Yuan、Y. Liu、J. Yue、Y. Yang、J. Wang、X. Feng、J. Zhao、X. Li 和 H. Yang，《STICKER：一种适用于 65纳米 CMOS 卷积神经网络的节能型多稀疏度兼容加速器》，*IEEE* 固态电路期刊，第 55 卷，第 2 期，第 465–477 页，2019 年。[8] J.-F. Zhang、C.-E. Lee、C. Liu、Y. S. Shao、S. W. Keckler 和 Z. Zhang，《SNAP：一种用于非结构化稀疏深度神经网络推理的高效稀疏神经网络加速处理器》，*IEEE* 固态电路期刊，第 56 卷，第 2 期，第 636–647 页，2020 年。[9] Y. Kuang、X. Cui、Z. Wang、C. Zou、Y. Zhong、K. Liu、Z. Dai、D. Yu、Y. Wang 和 R. Huang，《ESSA：可编程高效稀疏脉冲神经网络加速器的设计》，*IEEE* 超大规模集成电路系统汇刊，第 30 卷，第 1641 页，2022 年 11 月，第 1631 页。[10] C. Fang、Z. Shen、Z. Wang、C. Wang、S. Zhao、F. Tian、J. Yang 和 M. Sawan，《一种采用三维计算阵列的节能型非结构化稀疏性感知深度脉冲神经网络加速器》，*IEEE* 固态电路期刊，2024 年。[11] A. Khodamoradi、K. Denolf 和 R. Kastner，《S2N2：一种用于流式脉冲神经网络的 FPGA 加速器》，载于*2021*年 *ACM/SIGDA* 国际现场可编程门阵列研讨会，第 194–205 页，2021 年。[12] W. Guo、K. Yang、H.-G. Stratigopoulos、H. Aboushady 和 K. N. Salama，《采用高效基于Σ-Δ的脉冲编码方案的端到端神经形态无线电分类系统》，*IEEE* 人工智能汇刊，第 5 卷，第 4 期，第 1869–1881 页，2023 年。[13] T. J. O'Shea 和 N. West，《使用 GNU Radio 生成无线电机器学习数据集》，载于*GNU Radio* 会议论文集，第 1 卷，2016 年。[14] H.-H. Lien 和 T.-S. Chang，《用于目标检测的稀疏压缩脉冲神经网络加速器》，*IEEE* 电路与系统汇刊 *I*：正规论文，第 69 卷，第 5 期，第 2060–2069 页，2022 年。[15] K. Hunter、L. Spracklen 和 S. Ahmad，《两种稀疏性胜过一种：释放稀疏-稀疏网络的性能优势》，神经形态计算与工程，第 2 卷，034004，2022 年 3 月，第 034004 页。[16] K. Jung、J. Woo 和 S. Mukhopadhyay，《一种采用混合机器学习的片上加速器，用于射频信号调制分类》，载于*2022*年 *IEEE/MTT-S* 国际微波研讨会（*IMS 2022*），第 487–490 页，IEEE，2022 年。[17] H. Wang、Z. Qi、Z. Li 和 X. Zhao，《MR-Transformer：用于调制识别的 FPGA 加速深度学习注意力模型》，*IEEE* 无线通信汇刊，2024 年。[18] K. Pijackova 和 T. Gotthans，《使用深度学习架构进行无线电调制分类》，载于*2021*年第*31*届国际 *Radioelektronika* 会议（*RADIOELEKTRONIKA*），第 1–5 页，IEEE，2021 年。[19] S. Han、J. Pool、J. Tran 和 W. Dally，《同时学习权重和连接以构建高效神经网络》，神经信息处理系统进展，第 28 卷，2015 年。[20] A. Emad、H. Mohamed、A. Farid、M. Hassan、R. Sayed、H. Aboushady 和 H. Mostafa，《用于射频频谱监测的深度学习调制识别》，载于*2021*年 *IEEE* 国际电路与系统研讨会（*ISCAS*），第 1–5 页，IEEE，2021 年。[21] Q. Fournier、G. M. Caron 和 D. Aloise，《关于更快、更轻量 Transformer 的实用综述》，*ACM* 计算调查，第 55 卷，第 14s 期，第 1–40 页，2023 年。