arXiv:2301.01454v1 [cs.AR] 4 Jan 2023

基于现场可编程门阵列的高准确率、低延迟、高效率合成孔径雷达自动目标识别

Bingyi Zhang*、Rajgopal Kannan†、Viktor Prasanna*、Carl Busart†
*南加州大学 †DEVCOM美国陆军研究实验室
*{bingyizh，prasanna}@usc.edu †{rajgopal.kannan.civ，carl.e.busart.civ}@army.mil

摘要——**合成孔径雷达（SAR）自动目标识别（ATR）是遥感图像识别的关键技术。用于 SAR ATR 的最先进卷积神经网络（CNN）计算开销高且内存占用大，因此不适合部署在资源受限平台上，例如小型/微型卫星。**

本文提出一种基于 GNN 的模型与架构协同设计方案，并将其部署于现场可编程门阵列上，以解决上述问题。模型设计：我们为 SAR ATR 设计了一种新颖的图神经网络（GNN）。所提出的 GNN 模型采用 GraphSAGE 层算子和注意力机制，在计算开销近乎为 $1/100$ 的情况下，达到与最先进方法相当的准确率。随后，我们提出一种包含权重剪枝和输入剪枝的剪枝方法。通过套索回归进行权重剪枝，可在准确率不下降的情况下减少大部分参数；输入剪枝则能在准确率仅略微下降的情况下去除大部分输入像素。架构设计：为了充分释放所提出模型的计算并行度，我们开发了一种新颖的统一硬件架构，可执行多种计算内核（特征聚合、特征变换、图池化）。所提出的硬件设计采用 Scatter-Gather范式，高效处理各种计算内核中的不规则计算模式。我们在嵌入式现场可编程门阵列（AMD Xilinx ZCU104）上部署所提出的设计，并使用 MSTAR数据集评估其性能。与最先进的卷积神经网络相比，所提出的 GNN 在计算开销为 $1/3258$ 、模型大小为 $1/83$ 的情况下，达到相当的准确率。与最先进的中央处理器/图形处理器相比，我们的现场可编程门阵列加速器实现了 $14.8\times/2.5\times$ 的加速比（延迟），能效提高了 $62\times/39\times$。

索引词—**合成孔径雷达自动目标识别、图神经网络（GNN）、硬件架构**

I. 引言

合成孔径雷达（SAR）能够在各种天气条件下获取遥感数据，以观测地表目标。SAR 已广泛应用于农业 [1], [2], 文明 [3], [4], 等实际领域。SAR 自动目标识别（ATR）是对 SAR 图像中的目标进行分类的关键技术。卷积神经网络（CNN） [5]–[9] 已被广泛研究用于 SAR 自动目标识别，因为 CNN 能够从图像中提取具有判别性的特征。然而，基于 CNN 的方法 [5]–[9] 存在两个问题：（1）计算开销高：为实现高准确率，研究人员 [5]–[9] 开发了计算复杂度高的大型 CNN 模型；（2）内存需求大：这些大型 CNN 模型的参数数量庞大，需要占用大量内存。因此，不适合在资源受限平台（如小型/微型卫星 [10]–[14]）上部署大型 CNN。

上述问题的原因是：（1）CNN 中的卷积运算量大；（2）CNN 难以利用 SAR 图像中的数据稀疏性，因为 CNN 需要将整幅图像作为输入。在 SAR 图像中（图 1），只有少量像素属于目标（定义为感兴趣像素，即 POI），通过应用固定阈值 [15]即可轻松提取。然而，提取出的 POI 结构不规则，难以用 CNN 处理，而图神经网络（GNN）则提供了新的解决途径。直观而言，我们可以利用 POI 构建图，并使用 GNN 对图进行分类。值得庆幸的是，已有研究证明 GNN 是强大的模型 [16]，能够依据图结构信息和顶点特征对图进行分类。因此，GNN [17]–[19] 已应用于许多图分类任务 [20]–[24]。近年来，GNN 也已成功应用于许多图像分类任务 [25]–[27]。受此启发，我们为 SAR 自动目标识别设计了一种新型 GNN 模型（第 III-A 节）。我们提出了一种用于 SAR 图像的图表示 $\mathcal{G}(\mathcal{V}, \mathcal{E})$。所提出的 GNN 模型能够从构建的图中提取目标的结构信息。为提高分类准确率，我们利用包括空间注意力和通道注意力在内的注意力机制，识别重要的顶点和特征。为进一步降低计算复杂度，我们通过套索回归训练 GNN 模型来进行权重剪枝，并剪除绝对值较小的 GNN 模型权重。借助 GNN 模型，我们进行输入剪枝（POI 提取）。通过消除数值较小的顶点，计算复杂度降低了 92。8%，且准确率损失较小（$< 0.17\%$）。

所提出的图神经网络具有以下优势：（1）即使不进行权重/输入剪枝，所提出的图神经网络也能在准确率相近的情况下，实现与最先进的卷积神经网络近乎相同的计算开销；（2）卷积神经网络虽然可以利用权重剪枝，但难以利用输入剪枝，因为卷积神经网络必须使用整幅图像作为输入。图神经网络则可以灵活地使用一小组输入像素作为输入。因此，尽管我们可以在先进的卷积神经网络加速器上加速卷积神经网络，但其延迟仍然显著（第 VI-D 节）。

尽管所提出的 GNN 轻量级，能够部署在资源受限平台上，但加速 GNN 仍颇具挑战。GNN 具有不规则的计算模式和异构的计算内核 [29]，，因此难以在通用处理器上高效部署。

剪枝后的 GNN 模型通过权重剪枝引入了额外的不规则性。此外，所提出的模型包含多种异构计算内核（特征聚合、特征变换、图池化），需要映射到加速器上。尽管已有许多 GNN 加速器[29]–[35]被提出，但它们都未利用权重矩阵的稀疏性，也未处理图池化，因此对于所提出的模型而言仍然效率低下。尽管所提出的 GNN 以较低的计算复杂度实现了较高的准确率，我们认为，必须通过细致的模型-架构协同设计，才能实现合成孔径雷达自动目标识别的低延迟执行。

因此，我们为所提出的 GNN 模型开发了一种新颖的统一硬件架构，并展示了如何将各种计算内核映射到所提出的加速器上。在加速器设计中，我们采用 Scatter-Gather范式，高效处理各种内核的不规则计算模式。据我们所知，这是首个面向合成孔径雷达自动目标识别的基于图神经网络的模型—架构协同设计。我们的主要贡献如下：

- 我们提出了一种用于合成孔径雷达自动目标识别的轻量级 GNN，其准确率与最先进的卷积神经网络相当，同时显著降低了计算复杂度。
- 我们通过权重剪枝和输入剪枝，大幅降低计算复杂度和模型权重数量。
- 我们设计了一种统一的硬件架构，可执行所提出模型中的各种计算内核。我们采用 Scatter-Gather范式来处理这些内核的不规则计算模式。
- 借助所提出的硬件映射策略，我们进一步优化了各种计算内核的负载均衡（第 V-A 节）。
- 我们在 Xilinx ZCU104 上部署了协同设计，并使用MSTAR数据集对其进行评估。与最先进的卷积神经网络相比，所提出的 GNN 在准确率相当的情况下，计算开销为 $1/3258$ 、模型大小为 $1/83$ 。与最先进的中央处理器/图形处理器相比，我们的 FPGA加速器实现了 $14.8\times/2.5\times$ 的加速比（延迟），能效提高了 $62\times/39\times$ 。

II. 背景与相关工作

A. 相关工作

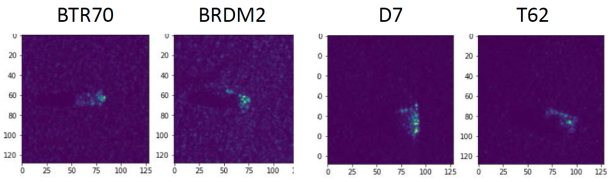

图 1：不同目标（车辆）的合成孔径雷达图像

合成孔径雷达自动目标识别旨在自动对给定合成孔径雷达图像中的目标进行分类（图 1）。为了实现高准确率，基于深度学习的方法已得到广泛研究。David[6] 证明，卷积神经网络的性能优于支持向量机等传统方法。TAI-SARNET [9]是一种卷积神经网络模型，通过结合空洞卷积和 Inception模块，实现了合成孔径雷达自动目标识别的高准确率。作者[8] 结合多视图特征，对合成孔径雷达图像中的目标进行分类。作者 [5] 利用空间注意力和通道注意力，提出了卷积块注意力模块。然而，最先进的卷积神经网络 [5], [8], [9] 计算开销较高，因此不适合部署在资源受限平台上。近期，作者[15] 将图神经网络应用于合成孔径雷达自动目标识别。他们按照像素灰度值递减的顺序连接像素，从合成孔径雷达图像构建图。然而，构建出的图丢失了目标的结构信息，因此对输入像素值的变化极为敏感。

表 I：符号说明

符号	说明	符号	描述
$\mathcal{G}(\mathcal{V},\ \mathcal{E},\ \boldsymbol{X}^0)$	输入图	v_i	i^{th} 顶点
$\mathcal{V}$	顶点集	e_{ij}	从 v_i 到 v_j的边
$\mathcal{E}$	边集	L	图神经网络层数
$\boldsymbol{h}_i^l$	v_i 在第 l 层的特征向量	$\mathcal{N}(i)$	v_i的邻居

B. 图神经网络

表 I 中定义了相关符号。图神经网络（GNN） [17]–[19] 被提出用于图上的表示学习 $\mathcal{G}(\mathcal{V},\ \mathcal{E},\ \boldsymbol{X}^0)$。GNN 能够从图的结构信息和顶点/边特征中学习，并将这些信息嵌入低维向量表示/图嵌入中（例如， $\boldsymbol{h}_i^L$是顶点 v_i的嵌入）。向量表示可用于许多下游任务，例如节点分类 [17], [18], 链接预测[36],图分类 [37], 等。GNN 遵循消息传递范式：顶点递归地从邻居聚合信息，例如：

GraphSAGE：GraphSAGE 在 [18] 中被提出，用于图上的归纳式表示学习。GraphSAGE 层遵循聚合-更新范式：

$$\begin{aligned}\text{aggregate:}\boldsymbol{z}_i^l &= \text{Mean}\left(\boldsymbol{h}_j^{l-1}: j\in\mathcal{N}(i)\cup\{i\}\right)\\ \text{update:}\boldsymbol{h}_i^l &= \text{ReLU}\left(\boldsymbol{z}_i^l\boldsymbol{W}_{\text{neighbor}}^l+\boldsymbol{b}_{\text{neighbor}}^l||\boldsymbol{h}_i^{l-1}\boldsymbol{W}_{\text{self}}^l+\boldsymbol{b}_{\text{self}}^l\right)\end{aligned}\tag{1}$$

III. 模型-架构协同设计

为在FPGA平台上实现准确、高效的合成孔径雷达自动目标识别，我们开展了全面的模型—架构协同设计。该协同设计包括：一种用于合成孔径雷达自动目标识别的新型GNN模型（第III-A节）、一种用于降低计算复杂度的剪枝策略（第III-B节）、一种用于高效执行所提出GNN的新型硬件设计（第III-C节），以及一种在各类计算内核之间保持负载均衡的策略（第V-A节）。我们硬件设计的核心创新在于，它能够执行所提出模型中的各类计算内核，并能高效处理由权重矩阵稀疏性导致的不规则计算模式。我们使用广泛采用的MSTAR数据集 [38]进行性能评估。我们关注多项性能

指标：（1）准确率：MSTAR数据集上的准确率；（2）计算复杂度：推理一幅SAR图像所需的总计算复杂度；（3）参数数量：模型中的参数总数；（4）延迟：推理一幅SAR图像所需的延迟；（5）能耗：推理一幅SAR图像所消耗的能量。

A. GNN 模型设计

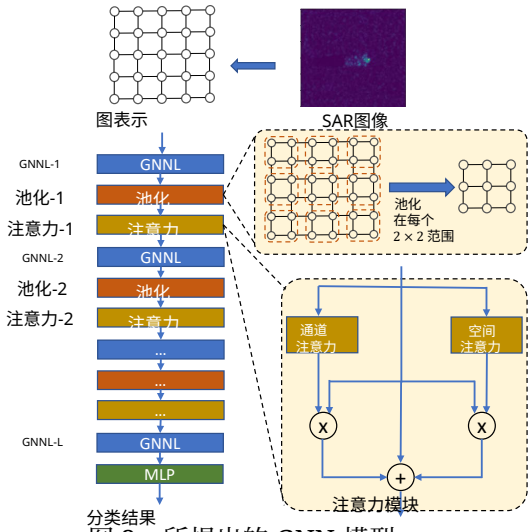

图 2：所提出的 GNN 模型

图表示：我们将一幅SAR图像表示为一个图$\mathcal{G}(\mathcal{V},\ \mathcal{E})$，并将每个像素视为一个顶点。每个像素/顶点通过边与其四个邻居（上、下、左、右）相连。顶点的特征是该像素的灰度值。这种图表示保留了目标的结构信息，可供GNN用于分类。此外，它也为输入剪枝提供了机会（第III-B节）。

GNN 模型：如图 2 所示，所提出的 GNN 模型由一系列层组成，包括图神经网络层（GNNL）、图池化层（池化）和注意力模块（注意力）。对于图神经网络层，我们采用GraphSAGE 层算子 [18]，这些算子已被证明在各种应用领域中能够取得出色的准确率。对于图池化层，由于输入图具有二维网格结构，我们采用与二维图像 CNN 类似的池化策略。在每个包含 s^2 个顶点的局部$s \times s$ 范围内，对 s^2 顶点执行池化算子（例如，最大值函数（）、最小值函数（））， 以获得一个输出顶点。图 2 展示了尺寸为 2×2 、步幅为 2 的池化操作。受 CNN 中注意力机制的启发 [39]，所提出的注意力模块由通道注意力模块和空间注意力模块组成。假设注意力模块的输入为 $\{\boldsymbol{h}_i :\ v_i \in \mathcal{G}\}$，其中 $\boldsymbol{h}_i \in \mathbb{R}^c$ 是 v_i 的特征向量， c 是特征向量的长度。通道注意力通过多层感知机计算每个特征的注意力分数 $\boldsymbol{F}_{ch}$ 。然后，将每个顶点与 $\boldsymbol{F}_{ch}$ 相乘，得到 $\{(\boldsymbol{h}_i)' :\ (\boldsymbol{h}_i)' = \boldsymbol{h}_i \otimes \boldsymbol{F}_{ch},\ v_i \in \mathcal{G}\}$，其中 $\otimes$ 表示逐元素乘法。空间注意力模块使用图神经网络层（GraphSAGE 层算子）计算每个顶点的注意力分数：

$$\{\alpha_i : v_i \in \mathcal{G}\} = \text{sigmoid}(\text{GNNL}(\{\boldsymbol{h}_i : v_i \in \mathcal{G}\})),$$

然后，将每个顶点特征向量乘以其注意力分数：$\{(\boldsymbol{h}_i)'' : (\boldsymbol{h}_i)'' = \alpha_i \boldsymbol{h}_i,\ v_i \in \mathcal{G}\}$。注意力模块的输出计算如下：

$$\{\boldsymbol{h}_i^{\text{output}} : \boldsymbol{h}_i^{\text{output}} = \boldsymbol{h}_i + (\boldsymbol{h}_i)' + (\boldsymbol{h}_i)'', v_i \in \mathcal{G}\} \quad (2)$$

经过GNNL-L后，所有特征向量都被展平为一个向量，并作为最后一个多层感知机（MLP）的输入，用于分类。

B. 网络剪枝

权重剪枝：为降低总计算复杂度，我们通过使用套索回归 [40]训练模型来进行权重剪枝。我们在损失函数中加入一个L1惩罚项：

$$\text{loss} = \sum_{i=1}^{N}(y_i - \text{Model}(\mathcal{G}_i))^2 + \lambda \sum_{w}^{W} |w|$$

惩罚项会导致权重收缩。模型中的部分权重变为零，并从模型中移除。训练后，我们设定阈值 I_{weight}，并剪除绝对值小于 I_{weight} 的权重。

输入剪枝：在合成孔径雷达图像中，目标之外的大多数像素的灰度值都可忽略。因此，在合成孔径雷达图像的图表示 $\mathcal{G}(\mathcal{V},\ \mathcal{E})$ 中，我们设定阈值I_{vertex}，并剪除灰度值小于 I_{vertex} 的顶点。与被剪除顶点相连的边也会被剪除。输入剪枝后，被移除的顶点在图池化层中仍保持原有位置，且不参与池化操作。例如，在局部2×2 范围内，如果一个顶点被剪除，池化算子将对其余三个顶点进行操作。对于输入到最后一个多层感知机的数据，被剪除顶点的特征向量会以零填充。

C. 架构设计

架构设计的目标是：(1) 支持所提出模型中的各种计算内核；(2) 处理由图神经网络层中的特征聚合以及权重矩阵的稀疏性所导致的不规则计算模式。图3展示了嵌入式FPGA平台上的所提出的架构设计。系统由一个应用处理单元（APU）和可编程逻辑区中的一个FPGA加速器组成。FPGA加速器执行GNN 模型的推理过程。在FPGA加速器中，权重/边缓冲区（WEB）用于存储模型权重和输入图的边；输入缓冲区（IB）用于存储输入顶点特征向量；结果缓冲区（RB）用于存储输出顶点特征向量。矩阵变换单元（MTU）执行矩阵变换，为下一层准备所需的数据布局。得益于所提出的轻量级模型，训练后的模型完全

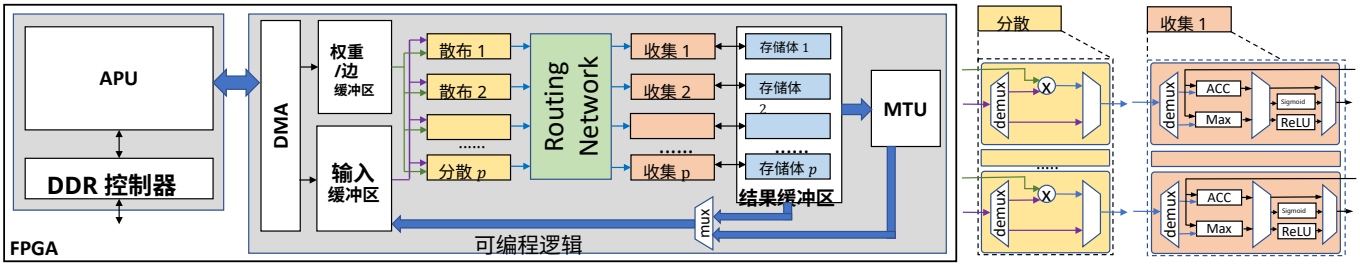

图 3：系统架构示意图

存储在权重缓冲区中，从而消除了运行时加载模型权重所产生的内存流量。

运行时：在运行时，APU 接收一幅输入合成孔径雷达图像，并将其转换为图表示。在转换过程中，灰度值小于 I_{vertex} 的像素会被剪枝。随后，APU 将输入图发送至加速器的输入缓冲区。加速器使用 Scatter-Gather 范式（SGP）执行每一层的计算。加速器利用每一层内的计算并行度。完成所有层的执行后，加速器将分类结果发送回 APU。

IV. 硬件映射

A. 计算内核

我们将计算内核分为两类：**顶点聚合内核（VAK）**：VAK 包括 (1) 特征聚合（在图神经网络层和空间注意力模块中）(2) 图池化。在 VAK 中，每个顶点都会将其特征向量传播给邻居或局部范围内的顶点（图池化）。**顶点更新内核（VUK）**：VUK 包括 (1) 特征更新（在图神经网络层和空间注意力模块中）(2) 注意力模块的通道注意力，(3) 最后的多层感知机。在 VUK 中，每个顶点的特征向量与权重矩阵相乘，以获得更新后的特征向量。由于进行了权重剪枝，权重矩阵具有较高的数据稀疏性（数据密度为 1%-33%）。

B. 使用 *Scatter-Gather* 范式进行内核映射

算法 1 Scatter-Gather 范式

当 未完成**时**
  散射单元：
  对于每条边 $e\langle src, dst, weight\rangle$ **执行**
    生成更新 $u \leftarrow$分散$(src.vector,\ e.weight)$
  结束循环
  Gather 单元：
  对于每个更新 $u\langle dst, vector\rangle$ **执行**
    更新顶点 $v_{dst} \leftarrow$ Gather$(u.vector)$
  结束 for
结束 while

该加速器设计基于 Scatter-Gather范式（算法1）。共有 p 条并行流水线。每条流水线由一个散射单元和一个 Gather 单元组成。路由网络根据索引 dst 将中间结果路由至目标位置。为了将 VAK 和 VUK

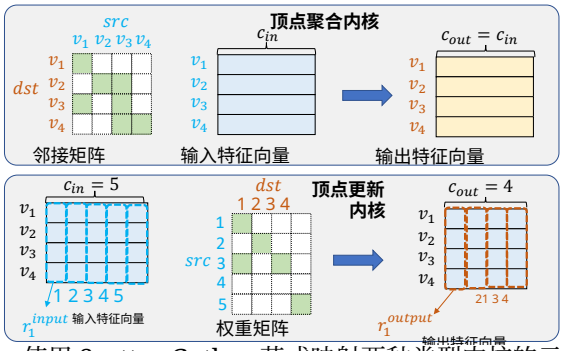

图 4：使用 Scatter-Gather 范式映射两种类型内核的示意图

映射到加速器上，我们提出了以下映射策略（示例如图4所示）：

VAK 映射：VAK 可以直接映射到加速器。对于每条边 $e\langle src, dst, weight\rangle$，散射单元从输入缓冲区加载 v_{src} 的特征向量，并生成一次更新 $u\langle dst, vector\rangle$。更新 $u\langle dst, vector\rangle$ 被路由到相应的 Gather 单元，Gather 单元将该更新应用于目标顶点 v_{dst}。

VUK 映射：对于 VUK，我们将一批顶点 $batch$ 分组，并将每个顶点的特征向量 $\{\boldsymbol{h}_i^{\text{input}}:\ v_i \in batch\}$ 与权重矩阵 $\boldsymbol{W}$ 同时相乘。输出特征向量为 $\{\boldsymbol{h}_i^{\text{output}}:\ \boldsymbol{h}_i^{\text{input}}\boldsymbol{W},\ v_i \in batch\}$。为应用 Scatter-Gather 范式，我们进行特征拼接。例如，我们将每个顶点 $\{\boldsymbol{h}_i(1)$ 的第一个特征：$v_i \in batch\}$ 拼接为一个向量 $\boldsymbol{r}_1^{input}$。该向量 $\boldsymbol{r}_1^{input}$ 的索引为 $src\,1$，因为它包含每个输入特征向量的 1st 特征。对于权重矩阵 $\boldsymbol{W}$，我们将权重矩阵中的每个非零元素表示为一条边 $e\langle src, dst, weight\rangle$。执行期间，对于每个非零权重 $e\langle src, dst, weight\rangle$，散射单元从输入缓冲区加载 $\boldsymbol{r}_{src}^{input}$，并生成一次更新 $u\langle dst, vector = e.weight \times \boldsymbol{r}_{src}^{input}\rangle$。然后，Gather 单元将更新 $u\langle dst, vector\rangle$ 应用于目标 $\boldsymbol{r}_{dst}^{output}$。$\boldsymbol{r}_{dst}^{output}$ 包含批次 $batch$ 中每个输出特征向量的 dst^{th} 特征。

请注意，VAK 和 VUK 的数据布局不同。在 VAK 中，输入/输出特征向量按顶点优先顺序存储；在 VUK 中，输入/输出特征向量按特征优先顺序存储。为在这两种数据布局之间切换，我们实现了一个矩阵变换单元（MTU）

以执行数据布局转换。

C. 硬件模块

散射/聚集单元：散射单元包含一个由 q个处理单元组成的阵列。每个处理单元都有一个乘法器，用于计算边/权重与顶点特征的乘积。与散射单元类似，收集单元包含一个由 q 个处理单元组成的阵列。每个处理单元都配有一个累加器（ACC）、一个最大值单元、一个 ReLU单元和一个 Sigmoid单元。多路复用器（MUX）和解复用器（DEMUX）为当前层选择数据通路。

路由网络：该路由网络采用硬件高效的蝶形网络 [41]实现。

Sigmoid单元：我们采用 Sigmoid函数的分段线性近似（PLA）[42] 。

V. 负载均衡与性能模型

A. 负载均衡

VAK 中的负载均衡：VAK 的工作负载均衡取决于如何将顶点划分到结果缓冲区的 p 个存储体中。如果图的度分布极不均衡，负载不均衡便是 GNN 中一个重要问题。得益于我们的图表示，图中顶点的度数范围为 0 到 4。我们采用贪心方法来保持 p 并行流水线的负载均衡。对于 VAK，度数相同 i $(0 \leqslant i \leqslant 4)$ 的目标顶点会均匀划分到结果缓冲区的 p 个存储体中。通过所提出的划分策略，每条流水线的工作负载都相同。图划分的开销很小 $\mathcal{O}(|\mathcal{V}|L_p)$，并由 APU 执行，其中 L_p 表示模型中的图池化层数量。所提出的划分算法可以轻松地通过 APU 上的多个线程并行化。

VUK 中的负载均衡：要执行 VUK，我们需要沿着 dst 维度划分权重矩阵（图4）。每个 Gather 单元负责累加一个划分的部分结果。为实现完美的负载均衡，每个划分应包含相同数量的非零元素。由于权重矩阵的划分属于离线过程，我们可以采用复杂度算法来寻找近乎最优的数据划分。在这项工作中，我们采用最长处理时间优先（LPT）算法，该算法已被证明能够达到 $4/3$ 近似比 [44]，从而逼近最优划分方案。

B. 性能模型

VAK建模：对于 VAK 内核，输入特征向量 c_{in} 的长度与输出特征向量 c_{out}的长度相同：$c_{\text{in}} = c_{\text{out}}$。每个时钟周期，散射单元或 Gather 单元可以处理 q个特征。 p 条并行流水线可以同时处理 p 条边。因此，VAK 内核的执行时间为：

$$t_{\text{VAK}} = \left\lceil \frac{|\mathcal{E}|}{p} \right\rceil \cdot \left\lceil \frac{c_{\text{in}}}{q} \right\rceil \tag{3}$$

VUK 建模：执行 VUK 时，加速器每次会将一批 q 个顶点分组，以充分利用散射单元和 Gather 单元。 p 条并行流水线可处理权重矩阵中的 p 个非零元素。因此，VUK 内核的执行时间为：

$$t_{\text{VUK}} = \left\lceil \frac{|\mathcal{V}|}{q} \right\rceil \cdot \left\lceil \frac{\text{nnz}(\boldsymbol{W})}{p} \right\rceil \tag{4}$$

其中，nnz($\boldsymbol{W}$) 是权重矩阵 $\boldsymbol{W}$中非零元素的数量。由于我们的加速器利用了每个内核内部的计算并行性，总执行时间等于所有内核的执行时间与预处理开销之和。

VI. 实现与实验结果

A. 实现细节与资源利用情况

我们在嵌入式现场可编程门阵列平台 Xilinx ZCU104 上实现了加速器。我们实现了 8 条流水线（8 个散射单元和 8 个 Gather 单元）。每个散射/聚集单元包含 16 个处理单元（PE）。在散射单元中，每个 PE 使用 3 个数字信号处理器；在 Gather 单元中，每个 PE 使用 7 个数字信号处理器。路由网络具有 8 个输入端口和 8 个输出端口。每个端口为 512 位，可接收/发送 16 个 32 位数据。APU 是一款运行频率为 1.3 GHz 的四核 ARM-A53 处理器。加速器采用高级综合开发。加速器使用了 1280 个数字信号处理器、96 个超大容量随机存取存储器、221 个块随机存取存储器和 178K 个查找表。加速器运行频率为 125 MHz。资源利用率和频率数据是在布局布线后报告的。

B. 基准测试与基准平台

基准测试：我们使用广泛采用的 MSTAR数据集进行实验。MSTAR数据集的设置遵循最先进的工作 [5], [6], [8], [9]。该数据集包含 10 类地面车辆的合成孔径雷达图像。训练集包含 2747 张图像，测试集包含 2427 张图像。每张合成孔径雷达图像的尺寸为 128×128，每个像素的灰度值表示合成孔径雷达信号的幅度。

表 II：各个平台的规格

平台	CPU AMD Ryzen 3990x	GPU Nvidia RTX3090	FPGA ZCU 104
发布年份	2020	2020	2018
工艺技术	台积电 7 nm	台积电 7 nm	台积电 16 nm
频率	2.9 GHz	1.7 GHz	125 MHz
片上存储器	256 MB L3缓存	6 MB L2缓存	4.8 MB

基准平台：如表 II 所示，我们将性能与最先进的中央处理器和图形处理器平台进行比较。在中央处理器平台和图形处理器平台上，我们使用 1.8.0 版本的 PyTorch Geometry (PyG) [45] 运行所提出的模型。对于中央处理器平台，PyG 使用 Intel MKL 作为后端；对于图形处理器平台，PyG 使用 CUDA 11.1 作为后端。为了利用中央处理器和图形处理器平台上权重矩阵的稀疏性，我们通过使用 torch.sspaddmm() 对 PyG 的 GraphSAGE 层[1]进行修改，以高效地实现特征向量与稀疏权重矩阵的乘法。

[1]https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/conv/sage_conv.html#SAGEConv

C. 准确率、计算复杂度、模型大小

权重/输入剪枝：SAR信号的幅值范围为 0 到 8。我们将 I_{vertex} 设为 0.1，因为它可以滤除大多数无关像素。我们比较准确率、计算量

	Type	准确率	浮点运算次数	参数量	模型大小
[5]	CNN	92.3%	$\frac{1}{12}\times$	0.5×10^6	16 Mb
[8]	CNN	97.97%	$\frac{1}{10}\times$	0.65×10^6	20.8 Mb
[9]	CNN	98.52%	$\frac{1}{3}\times$	2.1×10^6	67.2 Mb
[6]	CNN	99.3%	$1\times$ (6.94 GFLOPs)	2.5×10^6	80 Mb
本研究	GNN	99.09%	$\frac{1}{3258}\times$	0.03×10^6	0.96 Mb

计算复杂度、参数数量均优于最先进的工作 [5], [6], [8], [9]。与最先进的卷积神经网络 [6],相比，所提出的模型仅需$_{1\ 3258}$ 的计算复杂度和$_{1\ 83}$ 的参数数量，平均准确率却与之相当。

D. 延迟评估

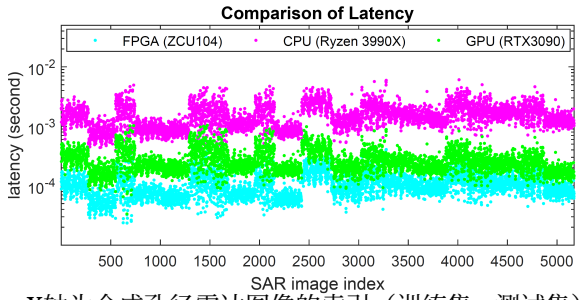

图5：X轴为合成孔径雷达图像的索引（训练集 +测试集）。Y轴为合成孔径雷达图像的推理延迟。

为比较不同平台的延迟，我们将批大小设为 1。FPGA加速器上的测量延迟为端到端延迟，从 APU 接收到合成孔径雷达图像开始，到 APU 从加速器获取分类结果为止，这意味着测得的延迟包含预处理开销。我们测量了训练集和测试集中所有图像的推理延迟。对比结果如图5所示。平均而言，就延迟而言，我们的 FPGA加速器比中央处理器和图形处理器平台快 $14.8\times$, $2.5\times$ 。由于我们采用了输入剪枝，图像经过输入剪枝后的图表示所含顶点数量各不相同。因此，推理延迟会随图像而波动。与中央处理器/图形处理器相比，我们的加速器延迟更低。这是因为中央处理器/图形处理器具有复杂的缓存层次结构，且缓存延迟较高（例如，中央处理器的缓存延迟较高：L3缓存为 32ns，L2缓存为 12ns）。因此，加载特征向量和权重矩阵会导致较大的延迟。相比之下，得益于定制的片上存储器组织方式，我们的 FPGA加速器能够在一个时钟周期内访问数据。此外，我们的 FPGA加速器采用分散-收集范式，高效处理各种计算内核中的不规则计算。

模型设计的影响：为了与最先进的卷积神经网络的推理延迟进行比较，我们在相同的

表III：ZCU 104 和图形处理器上的延迟比较

模型	[5]	[8]	[9]	[6]	所提出的模型
ZCU104	[Xilinx DPU] 0.88 毫秒	1.23 毫秒	3.09 毫秒	12.1 毫秒	[所提出的设计] 0.105 毫秒
图形处理器（RTX3090）	1.53 毫秒	2.5 毫秒	9.5 毫秒	31.2 毫秒	0.269 毫秒

FPGA平台（ZCU 104）上部署 AMD Xilinx DPU $115\times$ （2 * B4096 @ 300 MHz 配置），以执行 [5],[6], [8], [9] 中的卷积神经网络模型。AMD Xilinx DPU 是面向卷积神经网络的最先进的 FPGA 覆盖式加速器。平均推理延迟如表III所示。所提出的设计上的图神经网络（表III第 6 列）比 DPU 上的 [6]快 $115\times$ 。请注意，DPU 对权重和激活值采用 8位数据量化。我们的工作采用 32位浮点数据格式。DPU 通过处理 8 位数据实现了更高的计算并行度。

预处理开销：我们测量了 APU 上的预处理开销。对于一幅 SAR图像，APU 会通过输入剪枝（第 III-B 节）和图划分（第 V-A 节），将其转换为图表示（第 III-A 节）。每幅 SAR图像的平均预处理时间为 11.8 微秒，与总延迟相比可以忽略不计。

表IV：能耗对比

平台	推理速度	功耗	能耗（mJ/图像）
Ryzen 3990X	644（张图像/秒）	$26.5W$	41.1（毫焦/张图像）
Nvidia RTX3090	3717（张图像/秒）	$97W$	26.0（毫焦/张图像）
ZCU104	9500（张图像/秒）	$6.3W$	0.66（毫焦耳/张图像）

E. 能耗评估

表IV展示了各个平台的能耗对比。在 CPU 平台上，我们使用 PowerTOP [46]测量推理程序的功耗。在 GPU 平台上，我们使用 nvidia-smi [47] 命令工具测量功耗。对于 FPGA 板卡（ZCU 104），我们使用外部功率计测量其功耗。表IV中报告的数值是推理期间的平均功耗。结果表明，我们的 FPGA 加速器 $62\times$、 $39\times$ 的能效分别高于 CPU 和 GPU 平台。

VII. 结论

本文提出了一种面向合成孔径雷达自动目标识别的 FPGA 模型—架构协同设计新方法。所提出的轻量级图神经网络模型仅需 1/3258 计算复杂度和 1/83模型大小，即可达到与最先进模型相近的准确率。所提出的嵌入式FPGA平台加速器的延迟低于最先进的 CPU/GPU，且能耗显著更低。

致谢

本研究得到国家科学基金会（NSF）通过资助项目 OAC-1911229、CNS-2009057 的支持，部分经费由 DEVCOM陆军研究实验室（ARL）通过 ARL- USC 合作资助项目 DIRA-ECI:DEC21-CI-037 提供。作者 Bingyi Zhang 获得陆军研究实验室西部（ARL West）暑期研究计划的支持。

参考文献

[1] L. Landuyt、A. Van Wesemael、G. J.-P. Schumann、R. Hostache、N. E. Verhoest 和 F. M. Van Coillie，“基于合成孔径雷达的洪水制图：对既有方法的评估”，IEEE《地球科学与遥感汇刊》，第 57 卷，739，2018 2，第 722–页。[2] P. Zhan、W. Zhu 和 N. Li，“一种基于合成孔径雷达时间序列洪水信号的自动化水稻制图方法”，《环境遥感》，第 252 卷，第 112112 页，2021。[3] N. Li、Z. Guo、J. Zhao、L. Wu 和 Z. Guo，“利用星载合成孔径雷达表征黄河古河道：中国高分三号卫星案例研究”，IEEE《地球科学与遥感快报》，第 19 卷，第 1–5 页，2021。[4] T. Zhang、X. Zhang、J. Shi 和 S. Wei，“Hyperli-net：一种用于从合成孔径雷达图像中高精度、高速检测船舶的轻量级深度学习网络”，《国际摄影测量与遥感学会摄影测量与遥感杂志》，第 167 卷，第 123–153 页，2020。[5] M. Zhang、J. An、L. D. Yang、L. Wu、X. Q. Lu 等，“基于注意力机制的卷积神经网络用于 SAR 自动目标识别”，IEEE《地球科学与遥感快报》，第 19 卷，第 1–5 页，2020。[6] D. A. Morgan，“用于 SAR 图像自动目标识别的深度卷积神经网络”，载于《合成孔径雷达图像算法 XXII》，第 9475 卷。国际光学工程学会，2015，第 116–128 页。[7] J. Hu、L. Shen 和 G. Sun，“挤压-激励网络”，载于 IEEE计算机视觉与模式识别会议论文集，2018，第 7132–7141 页。[8] J. Pei、Y. Huang、W. Huo、Y. Zhang、J. Yang 和 T.-S. Yeo，“基于多视图深度学习框架的 SAR 自动目标识别”，IEEE《地球科学与遥感汇刊》，第 56 卷，第 4 期，第 2196–2210 页，2017。[9] Z. Ying、C. Xuan、Y. Zhai、B. Sun、J. Li、W. Deng、C. Mai、F. Wang、R. D. Labati、V. Piuri 等，“TAI-SARNET：面向小样本 SAR 自动目标识别的深度迁移空洞 Inception 卷积神经网络”，《传感器》，第 20 卷，第 6 期，第 1724 页，2020。[10] F. Bardi、W. Frodella、A. Ciampalini、S. Bianchini、C. Del Ventisette、G. Gigli、R. Fanti、S. Moretti、G. Basile 和 N. Casagli，“地基与卫星 SAR 数据在滑坡制图中的整合：圣弗拉特洛案例研究”，《地貌学》，第 223 卷，第 45–60 页，2014。[11] H. Septanto 和 O. Sudjana，“基于仿真的合成孔径雷达微卫星能量平衡分析”，载于第六届 LAPAN-IPB 卫星国际研讨会，第 11372 卷。国际光学与光子学学会，2019，第 113721W 页。[12] Y. Yokota、Y. Okada、K. Iribe、M. Tsuji、A. Ando 和 Y. Kunii，“日本小型卫星 ASNARO-2 搭载的新型 X 波段合成孔径雷达系统”，载于 2013年亚太合成孔径雷达会议（APSAR）会议论文集。IEEE，2013，第 81–83 页。[13] P. R. Akbar、H. Saito、M. Zhang、J. Hirokawa 和 M. Ando，“用于小型卫星搭载可展开合成孔径雷达天线的平行板缝隙阵列天线”，IEEE《天线与传播汇刊》，第 64 卷，第 5 期，第 1661–1671 页，2016。[14] K. Tanaka、H. Saito、H. Watanabe、K. Ijichi、K. Hirako 和 S. Shi-rasaka，“为按需观测的小型卫星研制的 1 k W 高功率 X 波段合成孔径雷达”，载于国际宇航大会论文集，IAC，第 2018 卷。国际宇航联合会（IAF），2018。[15] H. Zhu、N. Lin、H. Leung、R. Leung 和 S. Theodoidis，“基于像素灰度衰减和图卷积神经网络的 SAR 图像目标分类”，IEEE《传感器快报》，第 4 卷，第 6 期，第 1–4 页，2020。[16] K. Xu、W. Hu、J. Leskovec 和 S. Jegelka，“图神经网络的表达能力有多强？”，arXiv 预印本 arXiv:1810.00826，2018。[17] T. N. Kipf 和 M. Welling，“ 基于图卷积网络的半监督分类 ”，arXiv 预印本 arXiv:1609.02907，2016。[18] W. L. Hamilton、R. Ying 和 J. Leskovec，“大型图上的归纳式表示学习”，载于第31届神经信息处理系统国际会议论文集，2017，第 1025–1035 页。[19] P. Veli˘ckovi´c、G. Cucurull、A. Casanova、A. Romero、P. Lio 和 Y. Ben-gio，“图注意力网络”，arXiv 预印本 arXiv:1710.10903，2017。[20] V. Gligorijevi´c、P. D. Renfrew、T. Kosciolek、J. K. Leman、D. Berenberg、T. Vatanen、C. Chandler、B. C. Taylor、I. M. Fisk、H. Vlamakis 等，“利用图卷积网络进行基于结构的蛋白质功能预测”，《自然-通讯》，第 12 卷，第 1 期，第 1–14 页，2021。[21] H. Zhu、X. Du 和 Y. Yao，“Convsppis：利用带有特征图的集成卷积神经网络识别蛋白质-蛋白质相互作用位点”，Current Bioinformatics，第15卷，378，2020年4月，368页。[22] T. Zhao、Y. Hu、L. R. Valsdottir、T. Zang 和 J. Peng，“基于图卷积网络和深度神经网络识别药物-靶标相互作用”，Briefings in Bioinformatics，第22卷，第2期，第2141–2150页，2021年。[23] C. R. Qi、H. Su、K. Mo 和 L. J. Guibas，“PointNet：面向三维分类与分割的点集深度学习”，载于《IEEE计算机视觉与模式识别会议论文集》，2017年，第652–660页。[24] C. R. Qi、L. Yi、H. Su 和 L. J. Guibas，“PointNet++：度量空间中点集的深度层次特征学习”，《神经信息处理系统进展》，第30卷，2017年。[25] Y. Ding、X. Zhao、Z. Zhang、W. Cai、N. Yang 和 Y. Zhan，“用于高光谱图像分类的、采用ARMA滤波器和上下文感知学习的半监督局部保持稠密图神经网络”，IEEE《地球科学与遥感汇刊》，2021年。[26] Y. Ding、X. Zhao、Z. Zhang、W. Cai 和 N. Yang，“用于高光谱图像分类的图采样与聚合注意力网络”，IEEE《地球科学与遥感快报》，第19卷，第1–5页，2021年。[27] H. D. Nguyen、X.-S. Vu 和 D.-T. Le，“用于多标签图像分类的模块化图Transformer网络”，载于《AAAI人工智能会议论文集》，第35卷，第10期，2021年，第9092–9100页。[28] “Xilinx DPU。” [在线]。网址：https://github.com/Xilinx/Vitis-AI [29] M. Yan、L. Deng、X. Hu、L. Liang、Y. Feng、X. Ye、Z. Zhang、D. Fan 和 Y. Xie，“Hygcn：一种采用混合架构的GCN加速器”，载于《2020年IEEE高性能计算机体系结构国际研讨会（HPCA）》。IEEE，2020年，第15–29页。[30] B. Zhang、R. Kannan 和 V. Prasanna，“BoostGCN：一种优化FPGA上GCN推理的框架”，载于《2021年IEEE第29届现场可编程定制计算机器国际研讨会（FCCM）》。IEEE，2021年，第29–39页。[31] Y.-C. Lin、B. Zhang 和 V. Prasanna，“HP-GNN：在CPU-FPGA异构平台上实现高吞吐量GNN训练”，载于《2022年ACM/SIGDA现场可编程门阵列国际研讨会论文集》，2022年，第123–133页。[32] H. Zeng 和 V. Prasanna，“GraphAct：加速CPU-FPGA异构平台上的GCN训练”，载于《2020年ACM/SIGDA现场可编程门阵列国际研讨会论文集》，2020年，第255–265页。[33] B. Zhang、H. Zeng 和 V. Prasanna，“大规模GCN推理的硬件加速”，载于《2020年IEEE第31届专用系统、架构与处理器国际会议（ASAP）》。IEEE，2020年，第61–68页。[34] B. Zhang、S. R. Kuppannagari、R. Kannan 和 V. Prasanna，“CPU-FPGA异构平台上高效的基于邻居采样的GNN训练”，载于《2021年IEEE高性能极限计算会议（HPEC）》。IEEE，2021年，第1–7页。[35] H. Zhou、B. Zhang、R. Kannan、V. Prasanna 和 C. Busart，“面向FPGA上高性能时序GNN推理的模型-架构协同设计”，arXiv预印本arXiv:2203.05095，2022年。[36] M. Zhang 和 Y. Chen，“基于图神经网络的链接预测”，《神经信息处理系统进展》，第31卷，第5165–5175页，2018年。[37] R. Ying、J. You、C. Morris、X. Ren、W. L. Hamilton 和 J. Leskovec，“基于可微池化的层次图表示学习”，arXiv预印本 arXiv:1806.08804，2018年。[38] “MSTAR数据集。” [在线]。网址：https://www.sdms.afrl.af.mil/index.php?collection=public-data&page=public-data-list [39] S. Woo、J. Park、J.-Y. Lee 和 I. S. Kweon，“CBAM：卷积块注意力模块”，载于《欧洲计算机视觉会议论文集（ECCV）》，2018年，第3–19页。[40] R. Tibshirani，“通过套索回归进行回归收缩与变量选择”，《皇家统计学会学报：B辑（方法论）》，第58卷，第1期，第267–288页，1996年。[41] Y.-k. Choi、Y. Chi、W. Qiao、N. Samardzic 和 J. Cong，“HBM互连：面向FPGA HBM的高性能HLS互连”，载于《2021年ACM/SIGDA现场可编程门阵列国际研讨会》，2021年，第116–126页。[42] Y. Zhang、C. Wang、L. Gong、Y. Lu、F. Sun、C. Xu、X. Li 和 X. Zhou，“基于

FPGA硬件的长短期记忆网络加速器实现与优化”，载于*2017*年*IEEE*并行与分布式处理及应用国际研讨会和*2017*年*IEEE*普适计算与通信国际会议（*ISPA/IUCC*）。IEEE，2017年，第614–621页。[43] T. Geng、A. Li、R. Shi、C. Wu、T. Wang、Y. Li、P. Haghi、A. Tumeo、S. Che、S. Reinhardt 等，“Awb-gcn：一种具有运行时工作负载再平衡功能的图卷积网络加速器”，载于*2020*年第*53*届*IEEE/ACM*微体系结构国际研讨会（*MICRO*）。IEEE，2020年，第922–936页。[44] B. T. Eck 和 M. Pinedo，“在满足流时间最优性的条件下最小化最大完工时间”，《运筹学》，第41卷，第4期，第797–801页，1993年。[45] M. Fey 和 J. E. Lenssen，“使用PyTorch Geometric进行快速图表示学习”，载于*ICLR*图与流形表示学习研讨会，2019年。[46] “PowerTOP。” [在线]。网址：https://github.com/fenrus75/powertop [47] “nvidia-smi。” [在线]。网址：
htt
ps://developer.download.nvidia.com/compute/DCGM/docs/nvidia-smi-367.38.pdf