准确性与效率:通过 FPGA 实现感知神经架构搜索实现两者

Weiwen Jiang1,2,3 Xinyi Zhang2 Edwin H.-M. Sha1雷洋3,4 诸葛清风1

Yiyu Shi5 Jingtong Hu2

[1] 华东师范大学 [2] 匹兹堡大学 [3] 重庆大学

[4] 加州大学尔湾分校 [5] 圣母大学

抽象的

几乎所有深度神经网络的应用都存在一个基本问题:给定特定数据集的最佳神经架构是什么?最近,已经开发了几个神经架构搜索 (NAS) 框架,它们使用强化学习和进化算法来搜索解决方案。然而,由于巨大的搜索空间和评估每个候选人所需的漫长训练过程,他们中的大多数人需要很长时间才能找到最佳架构。此外,它们中的大多数仅以准确性为目标,而没有考虑将用于实现该体系结构的硬件。这可能会导致超出规范的过度延迟,从而使生成的体系结构变得无用。为了解决这两个问题,在本文中,我们使用现场可编程门阵列 (FPGA) 作为载体来展示一种新颖的硬件感知 NAS 框架,即 FNAS,它将提供最佳的神经架构,并保证延迟满足规范。此外,通过性能抽象模型在无需训练的情况下分析神经架构的延迟,我们的框架可以快速修剪不满足规范的架构,从而提高效率。在 ImageNet 等常见数据集上的实验结果表明,在最先进的生成架构的延迟比规范长 7.81 倍的情况下,来自 FNAS 的架构可以满足规范,精度损失小于 1%。

此外,FNAS 还为搜索过程实现了高达 11.13 倍的加速。据作者所知,这是第一款支持硬件的 NAS。

ACM 参考格式: Weiwen

Jiang, Xinyi Zhang, Edwin H.-M.沙, 雷洋, 诸葛清风, 史一宇, 胡靖彤。 2019. 准确性与效率:通过 FPGA 实现感知神经架构搜索实现两者。在 2019 年第 56 届年度设计自动化大会 (DAC 19) 中,2019 年 6 月 2-6 日,美国内华达州 Las Ve gas。 ACM,6 页。 https://doi.org/10.1145/3316781.3317757

DAC 19,2019 年 6 月 2-6 日,美国内华达州拉斯维加斯 © 2019 计算机协会。
ACM 国际标准书号 978-1-4503-6725-7/19/06。 . . 15.00
美元 https://doi.org/10.1145/3316781.3317757

1 简介

深度神经网络 (DNN) 的性能主要由其架构决定。然而,DNN 架构的设计在很大程度上依赖于人类的专业知识和劳动,直到最近开发出可以自动探索特定应用程序的最佳架构的神经架构搜索 (NAS)。现有的研究工作 [6, 16] 已经证明,NAS 可以生成与人类发明的 DNN(例如,AlexNet、VGGNet、GoogleNet 和 ResNet)相比具有竞争力甚至更高准确性的 DNN。然而,NAS 的普及受到其效率的阻碍。正如 [16] 中所报告的那样,即使使用数百个 GPU,搜索过程也可能需要几天时间。问题主要在于搜索空间可能很大,并且对于每个候选架构,都需要漫长的训练过程来对其进行评估。

此外,任何现有 NAS 框架的支柱是准确性是指导搜索的单一目标 [16]。如果生成的架构要部署在云中或者延迟不是关键因素,它们仍然可以工作。但是,如果要在具有延迟规范的硬件上实施该体系结构,则无法保证满足该规范。在这些场景中,NAS 找到的最佳架构根本没有用。

在本文中,我们提出了一种新颖的硬件感知 NAS 框架来解决上述问题。为了说明我们的框架,我们选择使用现场可编程门阵列 (FPGA) 作为载体,因为它凭借其高性能和能效逐渐成为实现 DNN 的最受欢迎的平台之一,特别是对于低批量实时应用[2]。为了引入硬件意识,在现有 NAS 框架中简单地包含一个额外的指标似乎很简单,该指标描述了 FPGA 上神经架构的延迟。然而,度量的评估可能具有挑战性。首先,与大多数人类发明的 DNN 中基于路径的规则结构不同,NAS 获得的架构可以是不规则的。许多现有的专用于人类发明的 DNN 的设计流程并不适合这种复杂的结构 [2-4、8、9、13-15]。其次,NAS 的架构通常体积较大,可能需要多个加速器进行协作。因此,应该考虑多个 FPGA 上的任务调度。因此,需要一种更优雅的方法来评估指标。

为此,我们提出了一个抽象模型,在软件(神经架构)和硬件(FPGA 设计)之间架起一座桥梁,以实现高效的延迟估计。具体来说,提出了一个基于图块的图模型来描述 FPGA 设计下的给定 DNN。在模型中,我们确定粒度

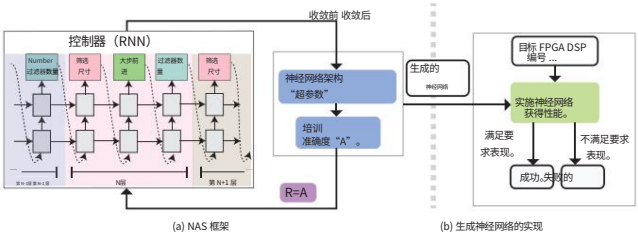


图 1:NAS 框架 [16] 及其实现。

根据 DNN 架构和设计中的平铺参数,任务、任务依赖关系和数据访问。然后,可以通过在任务之间添加额外的边来捕获 DNN 层之间复杂的依赖关系。

至于在多个加速器上的调度,文献[4,8,14]中的工作数量有限,仍然遵循单个FPGA[13]的调度器设计（见图5（a））。然而,这种调度范例无法充分利用 FPGA 之间的并行性。在这项工作中,我们提出了一种更灵活的调度机制（见图 5（b））。我们首先研究了调度器的设计原则,在此基础上我们提出了在抽象模型中调度任务的机制。此外,我们从理论上分析了执行和流水线停顿的延迟,以估计整体延迟。请注意,所提出的调度范例也可以广泛应用于超出本工作范围的多 FPGA 系统的设计。

本文的主要贡献如下。

- 框架。我们构建了一个 FPGA 实现感知神经架构搜索框架,即 FNAS,它可以在目标 FPGA 上生成具有保证延迟的最佳 DNN 架构。
- 抽象模型。我们提出了一个图模型来描述基于 FPGA 实现的神经架构,它为延迟分析提供了基础支持。此外,它还可以对不同类型的架构进行建模。
- 计划范例。我们提出了一种新颖的调度范例,以充分利用神经体系结构中的并行性。

在 ImageNet 等常见数据集上的实验结果表明,在最先进的 [16] 生成延迟比规格长 7.81 倍的架构的情况下,FNAS 的架构可以满足它们,准确率不到 1% 损失;意味着同时 FNAS 也实现了 11.13× 搜索过程的加速。

在本文的其余部分安排如下。第 2 节回顾相关背景,第 3 节展示我们的动机和问题制定。详细的 FNAS 算法在第 4 节中介绍。实验结果在第 5 节中显示,结论在第 6 节中给出。

2 背景

在本节中,我们将介绍神经架构搜索和基于 FPGA 的 DNN 实现的背景。

搜索神经网络架构。尽管关于自动预测神经网络架构的研究可以追溯到 1980 年代 [7],但在深度神经网络出现之后

表 1:FNAS 使用更少的时间生成在 PYNQ 上具有更低延迟且精度下降较小的架构。

方法		松弛。	纬度。	累积
	小姐	（小姐） 小鬼	(毫 小鬼	(%) 度。
网络存储 [17]	-	190m33s 10 74m29s	秒) -	99.42% -
FNAS	2.55×	59m19s 3.21×	19.70 8.67 2.27× 99.34% -0.08% 4.77 4.13× 99.18%-0.24%	
		17m07s 11.13×	1.80 10.94× 98.61% -0.81%	

在 AI 领域取得了巨大成功,最近人们对生成良好的神经架构的兴趣越来越大。随着架构越来越深,搜索空间呈指数增长,这使得搜索过程变得困难。在现有工作中,搜索架构有两个主要方向:(1) 采用强化学习 [1, 16, 17],以及 (2) 应用进化算法 [6, 10]。图 1 显示了 [16] 中提出的 NAS 框架。在 NAS 中,它迭代生成一个新的子网络,并通过在保留的数据集上训练它来获得它的精度 A。然后,精度 A 将用作下一次迭代的奖励信号。如果控制器收敛到最大精度,或者子网络的精度满足所需的精度 rA,则搜索过程将停止。然后将生成的最终设计实施到 FPGA 中。现有工作表明,自动搜索的网络架构可以达到与人类发明的最佳架构 [16、17] 非常接近的准确性。然而,有两个重要的挑战需要解决。首先,搜索过程效率低下。 [16] 报告说,在 4 天内通过 500 个 P100 GPU 训练了 20,000 个网络,以找到所需的网络。其次,生成的神经结构在牺牲推理速度的情况下实现了高精度。由此产生的网络通常很复杂,经常无法满足实时 AI 应用可用计算资源的时序规范。

表 1 报告了 NAS [16] 在 PYNQ 板 [5] 上使用 MNIST 数据集进行图像分类的结果。我们观察到完成 NAS 搜索需要 190 分钟 33 秒,延迟为 19.70 毫秒,生成网络的准确率为 99.42%。

FPGA 实施。 FPGA 已经展示了其出色的能力,可以为低批量实时推理实现高性能和高能效。带着这样的愿景,一系列在 FPGA 上实现神经网络的工作已经开展 [3, 4, 8, 9, 11, 13, 15]。现有的 NAS 没有考虑实现。为了使 NAS 进程实现感知,必须为每个子网络获得 FPGA 中的推理延迟,这通常通过生成 HLS 或 RTL 级代码 [8、13、15] 来分析,涉及人工干预和大量的时间。因此,如果我们简单地使用现有技术获得推理延迟并将其天真地集成到架构选择的奖励中,NAS 搜索过程将花费更长的时间。

在这项工作中,主要贡献之一是准确快速地估计 FPGA 推理延迟,以便搜索过程和网络生成都是高效的。表 1 还报告了拟议的 FNAS 的结果。从这张表中我们可以看出,对于相同的数据集,通过将推理时序规范（TS）设置为2ms,FNAS可以将搜索时间从190分钟减少到17分钟,达到11.13倍的减少。此外,PYNQ 上的推理延迟缩短了 10.94 倍。同时,准确率

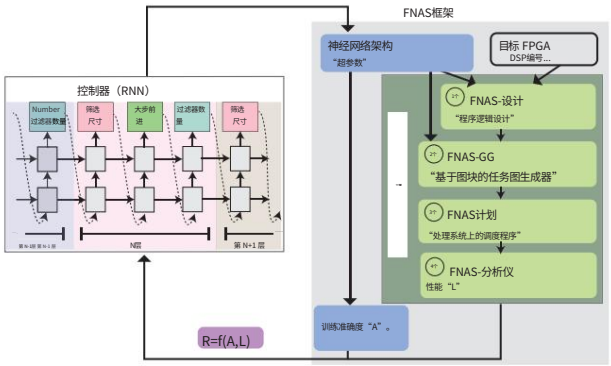


图 2:拟议的 FNAS 框架概述。

降解在 1% 以内。对于其他两种情况,FNAS 显着实现了超过 2 倍的加速,而准确率仅下降 0.08% 和 0.24%。这些验证了实现感知的 FNAS 可以显着减少搜索时间,并保证目标 FPGA 上生成的架构在推理阶段满足时序规范,同时保持准确性。

3 FNAS 框架

3.1 问题定义和 FNAS 概述

在本文中,我们的目标是开发一种 FPGA 实现感知神经架构搜索。该问题正式定义如下:给定特定数据集、目标 FPGA 平台和所需的推理延迟 rL,我们的目标是自动生成神经网络,使其在给定 FPGA 平台上的推理延迟小于rL,同时在给定数据集上实现机器学习任务的最大准确度。

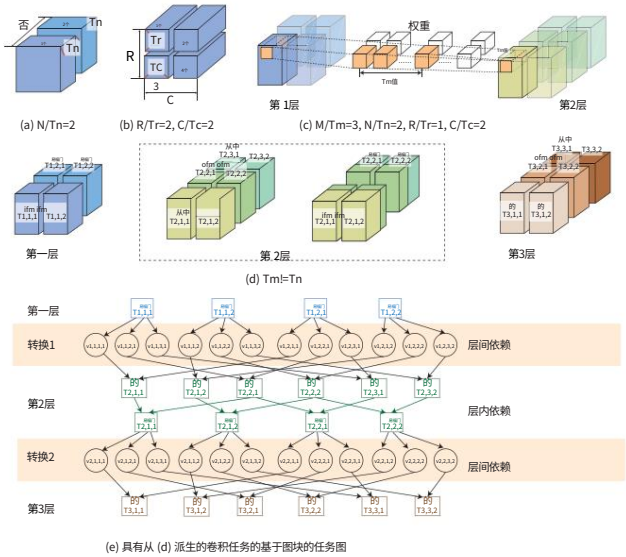
图 2 显示了 FPGA 实现感知神经架构搜索 (FNAS) 框架的概览。在 FNAS 中,它在子网络搜索过程中考虑了基于 FPGA 的推理性能。具体来说,FNAS 不是直接将准确度 A 作为奖励,而是使用奖励函数 f 来根据准确度 A 和性能/延迟 L 来计算奖励。为了高效准确地估计给定神经网络架构的推理延迟 L在目标 FPGA 上,我们开发了“FNAS 工具”。 FNAS 工具中有 4 个组件,包括 FNAS-Design、 FNAS-GG、 FNAS-Sched和 FNAS Analyzer。在下文中,我们将首先介绍奖励函数,然后一一介绍这些组件。

3.2 奖励功能

奖励函数采用准确度 A、延迟 L 和所需延迟 rL 来计算奖励信号。计算奖励 R 的函数定义如下。

$$R = \begin{cases} \frac{rL-L}{} - 1 & L > rL \\ (A - b) + \frac{大}{号} & L \leqslant rL \end{cases} \quad (1)$$

在上面的函数中,有两种情况。首先,如果 L > rL,表明最终系统的性能不能满足时序规范。在这种情况下,我们不训练子网络并直接向控制器返回负奖励。在第二种情况下,我们将性能和准确性的奖励L相加,其中性能奖励设置为表示解决方案L ,如果其延迟接近



(e) 具有从 (d) 派生的卷积任务的基于图块的任务图

图 3:FNAS-Design 和 FNAS-GG:(a)通道块; (b) 行/列图块; (c) 基于图块的卷积; (d) 区块编码; (e) 基于图块的任务图。

要求的水平。这里,b 是一个基线函数,它是先前架构精度的指数移动平均值 [16]。

3.3 Tiling Parameters: FNAS-Design由于

FPGA 上的资源有限,可能很难将整个卷积层放在 FPGA 上。因此,通常应用平铺技术将卷积运算拆分为多个小任务 [8、12、13、15]。 FNAS-Design 用于确定目标 FPGA 上给定 NN 架构的平铺参数。

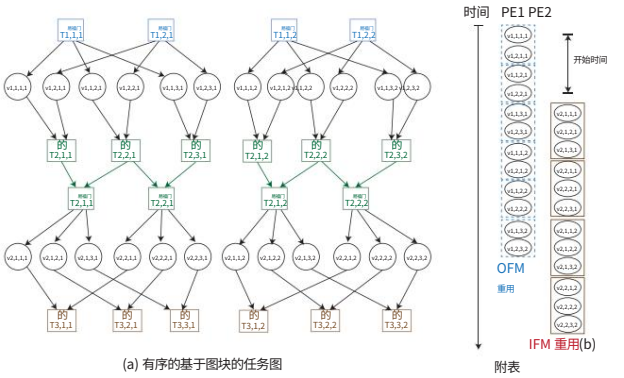
以一个卷积运算为例,它涉及四个参数Tm,Tn,Tr,Tc ,与输入/输出特征图（IFM/OFM)相关。这里,IFM通道的数量[illegible]被划分为图块,如图 3(a) 所示。类似地,具有 M 个Tn通道的OFM被划分为图块。另外,OFM的行数/列数分别为R和C。它们根据Tr和Tc平铺,如图 3(b) 所示。

否

米
Tm值

在拼接 IFM/OFM/row/col 之后,一个卷积运算被划分为更小的任务,如图 3(c) 所示。每个任务对应一对 IFM/OFM 块。一层中的任务将不断加载到 FPGA 上的处理单元 (PE) 中执行（加载顺序由 FNAS-Sched 确定）。对于一个任务,它涉及到Kh ×Kw ×Tr ×Tc ×Tm ×Tn乘法累加 (MAC) 运算,其中Kh和Kw是由控制器确定的过滤器的高度和重量。由 Tm ×Tn DSP组成的 PE可以并行执行Tm × Tn MAC 运算（16 位定点)[13]。则任务的延迟为 Kh × Kw ×Tr ×Tc 。

在 FNAS 中,每一层都分配给一个专用的 PE,PE 以流水线方式执行。这种架构可以在 [8、15] 中的一个 FPGA 或 [4] 中的多个 FPGA 上实现。可以通过考虑负载平衡来获得每一层的资源（例如,DSP和内存带宽）。然后可以根据 [8, 13]获得最佳参数Tm,Tn,Tr,Tc 。



(a) 有序的基于图块的任务图

图 4:FNAS-Sched:(a) 从 3(e) 重新排序的图； (b) 生成的时间表图。

3.4 基于图块的任务图生成器： FNAS-GG

FNAS-GG 是一种图生成器,它采用设计参数和神经网络架构来生成数据图块和任务之间的依赖图,称为基于图块的任务图。要生成图形,生成器首先需要为给定设计定义每个图块。然后,它可以生成基于图块的任务图。

根据 FNAS-Design,有两种瓦片:通道瓦片和行/列瓦片。对于channel tile,让 CHifm = {1, 2, · · · } tiling参数Tn下的层（con层的IFM）;同样CHof m = {1, 2, · · · } 是 i 中的一组索引 th 侧边我 日 在平铺参数Tm下（考虑第 i 层的 OFM）。对于行/列瓦片,让 RCi = {1, 2, · · · } 是第 i 层中在瓦片参数Tr和Tc下的一组索引。 ， 日 基于这些参数,我们可以如下定义瓦片。 我们在 IFM 中定义一个 tile 为 Tifm 指示图块在 i i,j,m, 日 层,j 日 CHifm中的通道图块和 RCi 中的第m行/列图块。 类似地,OFM 中的瓦片定义为 其中 k 是CHof m中通道瓦片的索引,k, m。然后,我们可以建立数据瓦片和任务之间的依赖关系。 .

有两种依赖关系。首先,层间依赖关系描述了两个连续层中任务和数据块之间的依赖关系。我们定义一个任务节点为bevi,j,k,m,它处理 tile Tifm

我, j, 米 并生成图块Tof m接下来,层内依赖 i,k,m。

性描述一层中两个数据图块之间的依赖性。如果一层的平铺参数Tm和Tn不同,如图 3(d) 中的第 2 层所示,则依赖将不是简单的一对一映射,而是可以

表示如下。 For the tile Tifm on tile Tof m if (j − 1) 它的数据取决于 i,k,T · i,k,m按照以上规则,可以生成图 3(d) 中的三 图 3(e) 所示。

3.5 调度器设计： FNAS-Sched

FNAS-Sched 是一个调度器,用于确定要在多个 PE 上执行的任务的顺序,从而使调度长度（延迟)可以最小化。 FNAS-Sched 试图最大限度地利用基于tile-based任务图的不同卷积运算之间的并行性。设计遵循三个原则：

- P1。尽量减少每个PE的启动时间,尽早执行任务。
- P2。最大限度地重用 FPGA 上的数据,以减少 通信带宽要求。
- P3。最小化由于缺乏输入而导致的流水线停顿 在一个 PE 上执行的数据。

FNAS-Sched 分三步进行。

第 1 步:确定每一层中 IFM 瓦片的顺序 - P1。 IFM 的执行顺序会影响下一层的开始时间。

有两种可能的策略:i)首先增加通道块的索引； orii) 首先增加行/列图块的索引。因为 OFM tile 与所有 IFM 通道相关,所以策略 i) 比 ii) 更受欢迎,可以使下一层更早开始。

第 2 步:确定每一层中 OFM 瓦片的顺序 - P1。

OFM 瓦片的顺序将决定同一层中 IFM 瓦片的就绪时间。在步骤 1 之后,IFM 块的顺序已经确定。因此,我们依次访问一层中的 IFM 瓦片 ,并排列依赖于它的 OFM 瓦片。

第三步:确定任务顺序 P2,P3。任务顺序将决定数据重用率。可以利用两种数据重用策略:i) OFM 重用 ,以及 ii) IFM 重用。 OFM（或 IFM)重用表示连续执行的任务具有相同的 OFM（或 IFM)块和相同的行/列块。我们观察到所有层的统一重用策略将导致由于缺少输入数据而导致管道停顿。在 FNAS 中 ,我们将对连续层交替应用上述两种策略。

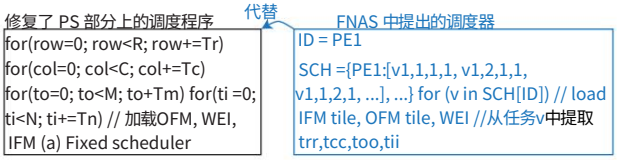
按照以上三步 ,我们就可以得到一个任务的时间表。对于图 3(e) 中基于图块的任务图,图 4(a) 给出了具有重新排序的 IFM/OFM 图块的图。我们还在图 4(b) 中给出了该图的时间表。如图4（b)所示,layer1（PE1)中的任务可以实现OFM重用,而IFM重用可以在layer2（PE2)中实现。此外,开始时间只有 4 个时间单位,两层的执行都没有停顿。

在FPGA实现中,调度器通常在处理系统（PS)端实现,加速器在编程逻辑（PL)端实现。图 5(a)、(c) 说明了 [13] 提出的常用 PS/PL 设计。在他们的设计中,不同的数据块以固定的顺序发送到 PL 部分,称为“固定调度”。为了实现所提出的调度器,我们修改了 PS 部分的实现,如图 5(b) 所示,我们可以在其中将任务指定给 PE,并按照我们的调度器确定的顺序启动任务。

3.6 延迟分析： FNAS-Analyzer

FNAS-Analyzer 旨在以确定的时间表高效准确地计算目标 FPGA 上神经架构的延迟 L。在调度中,PE 的延迟是三部分的总和:(1) 处理时间,(2) 开始时间,(3) 停顿时间。我们将在以下部分中分析它们中的每一个。

处理时间。我们首先确定基于图块的图中任务的执行时间。由于第 i 层中的所有任务都使用相同的加速器执行,因此它们具有相同的执行时间,表示为 ETi等于Kh,i × Kw,i ×Tr,i ×Tc,i （参见 FNAS 设计）。一个PE的处理时间PTi是对应第i层所有任务节点执行时间的总和,即



(b) 拟议的调度程序

```
PL 部分 (PE1) 上的加速器for(i=0;
i<K; i++) for(j=0; j<K; j++)
for(trr=row; trr<min(row+Tr,
R); trr++) for(tcc=col; tcc<min(col+Tc,C); tcc+
+)
// 展开后续 MAC 以并行处理for(too=to;
too<min(to+Tm,M); too++) for(tii=ti; tii<min(ti+Tn,N);
tii++)
        OFM[too][trr][tcc] += WEI[too][tii][i][j]*
                              IFM[tii][trr+i][tcc+j]
// 存储 OFM
```

(c) 加速器实现图 5:基于 FPGA 的 CNN 实现。

可以直接计算如下。

$$PTi = ETi \times |CHifm| \times |CHof\ m_{我+1}| \quad (2)$$

$CHifm| \times |CHof\ m_{我+1}|$是第 i 层的任务编号（见 ）。

开始时间。一层的开始时间取决于其上一层的开始时间和数据重用策略。令Δti,of m 为层 si -1 和 i 的开始时间之间的差距。我们类似地定义Δtj,ifm 。

首先,让我们考虑第 i − 1 层应用 OFM 重用,表示计算任务后 OFM 中的一个瓦片 $CHi-1$已准备就绪。对于第 i 层中的 Tn,i−1 Tn,i一个 IFM,它需要 OFM 块。因此， Tm,i可以计算如 Δti, of m

$$\Delta ti,\ of\ m = \frac{CHi\text{-}1}{Tn,i-1} \times \frac{,我}{,我} \times ETi-1 \quad (3)$$

接下来,考虑第 i − 1 层应用 IFM 重用来计算Δtj,ifm。
在这种情况下,第 i-1 层中的每个 IFM 将被重新用于计算所有相关 OFM 的部分和。在同一行/列瓦片中除最后一个之外的所有 IFM 完成后,它将在第 i 层连续生成 OFM。因此， Δtj,ifm可以计算如下。

$$\Delta tj,ifm = \frac{CHi\text{-}1}{Tn,i-1} - 1 \times \frac{CHj}{,j} + \frac{Tn,i}{Tm,i} \times ETj-1 \quad (4)$$

其中乘法的第一项表示在生成第 i 层中的第一个 OFM 之前完成的任务数。是第 i 层中一个 IFM 所需的 OFM 数量。 $\frac{,我}{,我}$

失速时间。 PE 启动后,可能会因为下一个任务的数据没有准备好而停止。但是,可能存在另一个已经准备好运行的任务。在我们的调度器中,我们可以维护一个准备运行的队列。如果发生停顿,我们将在准备运行队列中选择一个任务以避免流水线停顿。

潜伏。然后,我们可以通过总结处理时间和开始时间来得出延迟Latsys的严格下限。对于总共 N 个处理单元（PE）,假设第一个和最后一个 PE 应用 OFM 重用,我们可以计算Latsys如下。

$$拉特西斯= \sum_{i=2,4,\cdots,N-1} \Delta ti,\ of\ m + \sum_{j=3,5,\cdots,N} \Delta tj,ifm + PTN \quad (5)$$

表 2:FNAS 中的数据集和参数设置。

数据集	训练段			控制器参数			段。
	火车瓦尔。	EL FS				前线	T [TS4,TS3,TS2,TS1]
MNIST	60,000	10,000	25 4 [5,7,14]			[2,5,10,20] [1,4,10,20]	TS-高 TS-低 [9,18,36] 60
CIFAR-10	45,000	5,000	25 10 [1,3,5,7]	[24,36,48,64] 60			[1.5,2,2.5,10]
ImageNet	4,500	500	25 15 [1,3,5,7]	[16,32,64,128] 60	· L:层数； FS:滤波		[2.5,5,7.5,10]

器尺寸； FN:滤波器数量； T:小径； E:纪元

获得Latsys后,我们有延迟 L = Latsys 。

概括。 FNAS 框架在神经架构搜索过程中考虑了子网络在目标 FPGA 上的性能。如公式1所示,如果latency不能满足时序规范,则无需训练生成的子网络。此外,控制器将被引导避免搜索性能不足的架构。因此,可以显着加快搜索过程,并保证生成的子网络在目标 FPGA 上的性能。

4 实验结果

本节将报告拟议的 FNAS 框架的评估结果。在 MNIST、CIFAR-10 和 ImageNet 数据集上的结果表明,FNAS 在搜索过程中可以实现高达 11.13×、10.89× 和 10.38× 的加速。此外,对于 NAS [16] 生成的架构的延迟比规范长 9.85 倍的情况,FNAS 可以满足规范,精度损失小于 1%。

4.1 实验装置

数据集。 FNAS 将搜索卷积神经网络结构
三种数据集的结果,包括 MNIST、CIFAR-10 和 ImageNet。每个数据集由训练集和验证集组成,如表2所示。例如,训练集和验证集分别有60,000和10,000个示例。请注意 ,对于 ImageNet,我们使用较小的数据集来减少计算时间。在子网络的训练中,epochs 的数量设置为 25,最后 5 个 epochs 中的最大验证精度将用于计算更新控制器的奖励。

控制器。我们基于 [16] 实现基于强化学习的 RNN 控制器来生成子网络。对于不同的数据集,控制器有不同的配置,如表2所示。例如,我们解释MNIST的配置：(1)它的子网络有4层，(2)可能的过滤器尺寸（高度和宽度)是5、7 或 14,(3) 可能的频道号是 9、18 或 36,以及 (4) 它会找到 60 个子网络。

FNAS 工具。我们实现第 3 节中描述的所有组件并将它们集成到控制器中以实现 FNAS 框架。为了将 FNAS 与 NAS 进行比较,我们使用低端和高端 FPGA 来实现使用 MNIST 数据集的结果架构。选用的低端和高端FPGA分别是Xilinx 7A50T和7Z020。为了了解我们的结论的普遍性,我们探索了 CIFAR-10 和 ImageNet 作为附加数据集,其中使用了 Xilinx ZU9ED。

4.2 FNAS 的效率和准确性评估图 6(a)、(b)、(c) 报告了结果 DNN 在 MINST 数据集上的搜索时间、延迟和准确性的比较结果。在这些图中,x 轴代表不同的 FPGA。 FNAS-loose (TS2)、FNAS-med (TS3) 和 FNAS-tight (TS4) 对应三个时序规范（见表 2）。

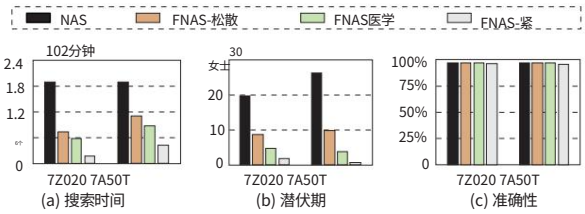


图 6:NAS 和 FNAS 的搜索时间、延迟和准确性的比较。

从图 6(a) 中,我们可以看出 FNAS 可以显着减少搜索时间。以7Z020为目标FPGA,对于松、中、紧时序规范（表2实际值），搜索时间从190分钟减少到74、59、17分钟,分别减少2.56×、3.22×、11.13×分别与NAS相比。改进的原因有两个：(1)通过早期修剪,我们不会训练推理延迟违反规范的架构； (2) 有效的 DNN 的结构通常比有违规的 DNN 更简单,因此大多数复杂的架构无需训练即可自然修剪。从图中我们还可以看出,对于 FNAS,搜索时间随着时序规范变得更严格而减少,这也是预期的,因为更严格的规范会修剪更多潜在的架构。

接下来,如图 6(b) 所示,对于不同的时序规范,FNAS 可以生成特定的架构来满足规范,即架构的延迟随着规范变得更严格而降低。相反,NAS只能生成单一架构,其延迟比规范长2.54×、4.19×和7.81×。 FNAS 的灵活性为设计人员提供了更多选择,也有助于在非常早期的设计阶段修剪无用的设计。

图 6(c) 报告了生成架构的准确性。
我们可以观察到,即使与由 NAS 生成的具有时序违规的架构相比,由 FNAS 生成的架构对于松散、中等和紧时序时序规范的准确度下降分别仅为 0.08%、0.24% 和 0.81%。上述结果清楚地表明,FNAS 在探索神经结构方面可以同时实现高效率和高精度。

最后,我们探讨了 FNAS 生成的架构的准确性以及相应的搜索时间如何与时序规范相关联。结果如图 7 所示,其中使用了三个数据集 MNIST、CIFAR-10 和 ImageNet。

对于 MNIST,使用的是高端 FPGA。 x 轴表示不同的时序规范,其中 TC1 是最宽松的,而 TC4 是最严格的,相应的值总结在表 2 中。在报告精度和搜索时间时,我们使用 NAS 的架构作为参考,显示精度损失以及搜索时间减少。从图 7(a) 中我们可以观察到,与有违规的 NAS 生成的架构相比,没有时序违规的 FNAS 生成的架构通常只有不到 1% 的精度损失。随着时序约束变得更严格,精度损失变得更高。

同时,从图 7（b)可以看出,FNAS 的搜索时间可以显着减少,与 NAS 相比,CIFAR-10 的搜索时间减少了 11.18 倍。

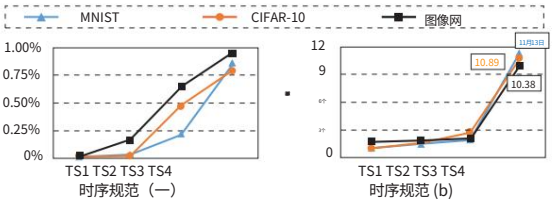


图 7:(a) 精度损失,和 (b) 三个数据集的搜索时间减少。 NAS 用作基线方法。

5 结论

在这项工作中,我们使用 FPGA 作为工具来探索硬件感知神经架构搜索 (NAS)。我们的目标是以最佳精度自动搜索神经架构,同时满足目标 FPGA 的时序规范。为了实现这一目标,提出了一种新型的 FPGA 实现感知 NAS 框架。在该框架中,我们构建了一个性能抽象模型和一个新的调度范例,以充分利用多个 FPGA 之间的并行性。评估结果表明,对于最先进的 NAS 生成推理延迟比规范长 7.81 倍的架构的情况,所提出的框架可以满足规范,精度损失小于 1%,以及高达 11.13 × 搜索过程中的加速。

致谢

这项工作得到了中国国家自然科学基金 61472052 号拨款、国家科学基金 CCF-1820537 号拨款以及中国国家留学基金委 201706050116 号和 201706050117 号拨款的部分支持。

参考

[1] 鲍文贝克等。 2016. 使用强化学习设计神经网络架构。 arXiv 预印本 arXiv:1611.02167 (2016)。
[2] 埃里克·钟等人。 2018. 使用 Project Brainwave 在数据中心规模实时提供 DNN。 IEEE 微型 38, 2 (2018), 8–20。
[3] 杰里米福尔斯等人。 2018. 可配置的云级 DNN 处理器,用于真实时间人工智能。在过程中。伊斯卡。 IEEE 出版社,1-14。
[4] 蒋伟文等。 2018. 基于异构 FPGA 的成本优化设计时序约束 CNN。 IEEE TCAD (2018)。
[5] 平克。 2018. PYNQ:ZYNQ 的 Python 生产力。 http://www.pynq.io/（2018 年）。
[6] Esteban Real 等人。 2017. 图像分类器的大规模进化。 arXiv 预印本 arXiv:1703.01041 (2017)。
[7] J David Schaffer 等人。 1992. 遗传算法和神经网络的结合:对现有技术的调查。在过程中。 COGANN-92。 IEEE,1-37。
[8] 沉永明等。 2017. 通过资源分区最大化 CNN 加速器效率。在过程中。伊斯卡。 535–547。
[9]魏学超等。 2018. TGPA:用于低延迟的平铺粒度管道架构 CNN 推断。在过程中。国际计算机辅助设计协会。美国计算机学会,58。
[10] 谢令喜等。 2017. 遗传 CNN .. 在过程中。的ICCV。 1388–1397。
[11] 许晓薇等。 2018. 使用 FPGA 进行实时障碍物检测的资源受限细胞神经网络。在过程中。的 ISQED。 IEEE,437–440。
[12] 雷洋等。 2018. 在基于 STT-RAM 的路由器中计算片上网络的最佳应用映射和调度。 IEEE 技术委员会 (2018)。
[13] 陈章等。 2015. 为深度卷积神经网络优化基于 fpga 的加速器设计。在过程中。的FPGA。美国计算机学会,161–170。
[14] 陈章等。 2016. 深度节能 CNN 实现流水线 FPGA 集群。在过程中。的 ISLPED。 326–331。
[15] 张晓凡等。 2018. DNNBuilder:用于为 FPGA 构建高性能 DNN 硬件加速器的自动化工具。在过程中。国际计算机辅助设计协会。美国计算机学会,56。
[16] Barret Zoph 等人。 2016. 强化学习的神经结构搜索。 arXiv 预印本 arXiv:1611.01578 (2016)。
[17] Barret Zoph 等人。 2017. 学习可缩放图像的可迁移架构认出。 arXiv 预印本 arXiv:1707.07012 2, 6 (2017)。