人工神经网络和加速器使用进化算法进行协同设计

Philip Colangelo英特尔 PSG 美国圣何塞 philip.colangelo@intel.com

Oren Segal霍夫斯特拉大学 美国亨普斯特德 oren.segal@hofstra.edu

Alex Speicher Hofstra 大学 美国亨普斯特德 aspeicher1@pride.hofstra.edu

马丁马加拉美国马萨诸塞州洛厄尔大学洛厄尔分校 Martin Margala@uml.edu

摘要 多层前馈人工神经网络 (ANN) 是通用函数逼近器,能够以任何所需的精度对可测量函数进行建模。

在实践中,设计实用、高效的神经网络架构需要大量的努力和专业知识。此外,为了加速的好处,设计最适合硬件的高效神经网络架构增加了另一个程度的复杂性。在本文中,我们使用进化细胞辅助设计 (ECAD),这是一个能够搜索 ANN 结构和可重构硬件的设计空间的框架,以根据一组约束和适应度函数找到解决方案。使用 OpenCL 为 Arria 10 GX 1150 FPGA 设备提供基于模块化和可扩展二维脉动阵列的机器学习加速器设计,可以在真实硬件中测试和部署结果。除了硬件之外,还开发了架构的软件模型以加速演化过程。我们展示了 ECAD 框架的结果,显示了各种优化(包括准确性、每秒图像数、每秒有效千兆操作数和延迟)对 ANN 和硬件配置的影响。

通过这项工作,我们证明了每个优化都可以存在独特的解决方案,从而获得最佳性能。这项工作为为具有不同系统约束的广泛应用寻找基于机器学习的解决方案奠定了基础。

索引词进化算法,机器学习,FPGA

一、引言

设计高性能 NNA 的困难最近引起了人们对 NNA 自动设计的兴趣激增。现有研究机构的重点一直是优化 NNA 设计以提高准确性 [1][2][3]。优化 NNA 通常是一个困难的过程,部分原因是存在大量的超参数组合,并且在组合不是最优的情况下,性能会受到影响。事实上,TensorFlow [4] 和 Keras [5] 等许多深度学习框架都提供对超参数调整的支持,但这些框架通常是将 ANN 视为黑盒并用于神经网络训练过程的贝叶斯优化。

研究表明,网络参数可以直接影响该模型在部署中的准确性、吞吐量和能耗 [6]。

一旦找到准确的 NNA,下一步就是尝试将其装入现有硬件,即 CPU、GPU 或定制但通用的神经网络硬件设备,例如 TPU [7]。这些硬件解决方案都不提供网络具体专业。两种优化之间的差距是 ECAD 的用武之地,它允许通过探索 NNA 和硬件端的设计空间来搜索最佳硬件/NNA 协同设计,并允许为特定的 NNA 模型实施定制硬件解决方案使用可重构硬件。

在本文中,我们提供了以下贡献:1) ECAD,一种使用进化算法进行人工神经网络协同设计和可重构硬件设计的框架。

2) 针对各种健身目标进行优化的示例和结果,包括最高准确度、每秒图像数、延迟和每秒有效千兆操作。

二。相关工作

在过去的几年中,我们已经看到使用深度神经网络 [8][9] 的 ML 算法在复杂任务上的性能取得了长足的进步。为 ML 手动设计最先进的深度神经网络架构需要大量的时间和劳动力 [2][10]。在过去的几十年里,自动化 NNA 搜索一直是一项持续的努力,但由于设计复杂性不断增加的深度网络存在困难,因此正在成为 NNA 研究界的焦点 [2][3][10]。自动人工神经网络架构搜索 (NAS) 可以使用不同的策略进行,例如随机搜索、进化算法、强化学习 (RL)、贝叶斯优化和基于梯度的方法 [10]。多年来,使用进化算法 (EA) 搜索高性能架构已得到广泛研究 [11][12]。最近的一些结果表明,进化算法提供了比随机搜索和强化学习更好的结果 [3] 最近,人们对专门从事图像识别的深度神经网络的 NAS 越来越感兴趣 [1][2][13]。

随着深度和复杂的神经网络变得越来越流行,并且意识到现有的硬件架构并未针对此类计算进行专门优化,因此提出并设计了新形式的专用架构 [7] 以帮助提高性能和能效。由于神经计算领域发展如此迅速,设计这种静态的新架构可能会非常昂贵和冒险。为神经网络优化硬件是一个不断发展的研究课题

978-1-7281-5020-8/19/$31.00 ©2019 IEEE授权许可

并与正在开发的不断变化的网络结构相关联。最近的出版物表明,新架构通过提供加速神经网络应用程序工作负载的独特方法,在深度学习中占据一席之地 [14][15][16]。具体来说,这些可重构架构是一个流行的研究和部署平台,因为它们能够通过低数值精度利用神经网络的弹性来更改其结构路由和资源结构以适应各种工作负载和优化 [17][18]] 和稀疏性 [19][20]。其他加速器设计使用可重构结构来更改逻辑路由以预处理数据并卸载到专用 ASIC [21]。

存在多个工具流程来优化可重构硬件 (FPGA) 的固定 NNA 设计。大多数可用的工具流都针对图像识别任务。此处提供了最近关于可用工具流的调查 [22]。

尽管针对 NAS 的优化可以导致更简化的 NNA,进而可以简化和优化硬件设计 [3] [10],但 NAS 的主要工作集中在准确性作为性能的主要衡量标准。另一方面,硬件性能参数（延迟/吞吐量/功率)的优化通常是在现有的 NNA 设计上完成的,并且没有尝试修改 NNA（层/神经元等)[22]。

结合 NAS 和硬件优化可能会关闭 NNA 的设计和实现之间的循环[22]。

据我们所知,ECAD 是第一个能够进行 NAS 和硬件协同优化的框架。它能够同时在 NNA 级别（神经元/层等)和硬件级别（LUTS/DSP 等)上工作,即给定一个通用的 NNA 结构,它将进化并搜索两个空间（NNA/硬件)串联。

三、 ECAD软件

ECAD 旨在创建针对特定设计目标进行优化的 NNA。软件方面的核心是进化算法和适应度函数向量。

健身功能目前包括准确性、速度、能源效率和吞吐量的测量。 ECAD 允许选择赋予每个适应度函数的重要性（权重）,并通过这样做引导进化过程朝着所需的适应度目标发展。结果是针对适应度函数中指定的目标优化的神经网络设计。图 1 显示了 ECAD 流程的概览。接下来的部分提供了流程中每个阶段的详细信息。

种群生成最初,系统将使用配置文件中指定的基本设计来创建神经网络种群。种群初始大小、最大大小和变化率都使用配置参数进行控制。每个自动生成的网络实例都会发生变化,并且与原始基础设计不同。

随着进化过程的进行,一旦足够数量的网络根据所有适应度进行评估参数,种群生成过程将不断重复,除了突变将基于种群中最适合的个体,根据他们的适应度分数选择（参见 [23] 中的稳态模型）。

测试种群适应度 在下一阶段,每个 NN 实例将被发送到 ECAD 术语中的一个或多个适应度评估器或工作人员。工作人员被设计为独立的进程,可以分布在计算机集群中。它们使用在 MPI [25] 之上运行的主/工作并行计算模型 [24] 进行编排。

每个 Worker 都位于一个单独的进程中,继承自一个通用的 C++ Worker 类,并实现一个通用的软件接口。该系统的构建非常灵活,可以轻松添加不同类型的工人。每个 worker 的实现细节可以完全不同。

我们目前实施了三种类型的工人：

1) 能够模拟 NN 设计的 Simulation Worker
   并返回精度和计时结果
2) 能够综合 NN 设计的 Physical Worker
   并返回硬件综合结果
3) HWDB Worker能够准确估计NN综合结果并返回估计的硬件综合结果

一旦工人的适应性评估完成,它将把结果返回给主/服务器进程。主进程将收集评估每个 NN 的所有工作人员的结果,并将综合分数应用于每个 NN。使用在最初创建时分配给每个生成的 NN 实例的唯一 ID 来组合分数。

按适应度排序种群ECAD 将根据其分数对 NN 种群进行排序,排名靠前的实例将发生变异。变异的神经网络将被引入种群并被送去评估,这个过程将一直持续到满足结束条件为止。结束条件可以是要求模拟流程运行的代数或满足所需的适应度。

神经网络模拟器神经网络模拟器 (NeuralNetSim) 负责测试和验证 ECAD 神经网络架构。我们选择 TensorFlow[4] 作为机器学习仿真框架。 Neural NetSim 由一个 ECAD 阅读器组成,该阅读器充当进化框架的接口。它的主要职责是接受文件和训练参数等输入,以及在训练完成后返回数据和报告。它从 ECAD 文件中提取网络信息并将其传递给 TF Model Builder。 TF Model Builder 保存实际的 TensorFlow 图和所有其他 TensorFlow 变量。它负责根据读者传递的信息动态创建图表。最后,它还包含从 ECAD 阅读器调用的训练函数。 TF 函数是 TensorFlow API 函数的集合,在构建图形时由 TF 模型构建器调用。

978-1-7281-5020-8/19/$31.00 ©2019 IEEE授权许可

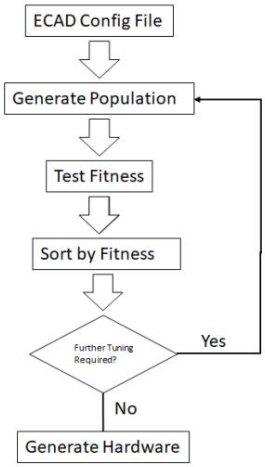

图 1. ECAD 流程

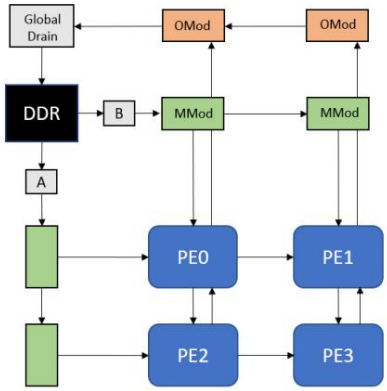

图 2. 用于设计空间探索的 2D 脉动阵列硬件架构

四、 ECAD硬件:二维收缩阵列执行

脉动阵列 (SA) [26] 非常适合 FPGA,因为它们具有流水线数据流和内存带宽调整功能。图 2 中显示的高级设计概述了 ECAD HWDB Worker 使用的架构。以下部分详细描述了该体系结构。

A. 矩阵分块和内存注意事项

如图 3 所示,矩阵针对 SA 的空间配置进行分块,并通过图 2 中描绘为 A 和 B 的加载器内核进行读取。矩阵 分块高度由 SA 中的行数乘以交错因子来定义。矩阵 B 块的高度与矩阵 A 块的宽度相同,由向量宽度和比例因子的乘积定义。矩阵 B 的宽度定义为 SA 中的列数乘以交织因子。交错是为平衡数据重用而调整以缓解带宽限制的参数。这交织因子越大,数据重用越多,传送 PE 所需的带宽越少,但随着块大小的增长,映射到较小矩阵大小的效率会降低。比例因子是用于启用更高效的全局内存访问的参数。我们将全局内存视为 512 位缓存行。每次访问全局内存都会读取一个矢量宽度的数据量,如果矢量宽度小于 512 位,那么这可能会导致带宽利用率不理想。 Scale 将调整块宽度,以便每次读取到全局内存时更接近 512 位并且效率更高。块连续存储在 DDR 内存中,以简化全局内存访问模式。因此,我们在主机上转置矩阵 B,以便顺序内存访问遍历它

行。

B. 内存模块

内存模块 (MMod) 只不过是一个菊花链智能双缓冲区,在将当前数据块写入 PE 的同时,将下一个数据块从加载器模块读入本地缓存。 MMod 沿行和列维度链接,最外面的模块连接到加载程序。按照图 4,MMod 由输入路由器组成,其工作是首先将输入块定向到写选择多路分解器,然后切换到将数据向下发送到其邻居 MMod。一旦诸如 Mem0 之类的缓存已满,buff sel select 将更新,以便将新块写入一个内存,同时读取多路复用器从另一个内存中获取数据。两个存储器都以不存在读取或写入争用的方式进行仲裁。图 4 中描绘的所有非选择线都具有支持数据向量值的宽度。

C. 处理元素

每个 PE 负责计算一个点积。外围 PE（PE0、PE1、PE2,如图 2 所示）从内存模块 (MMod) 获得一个输入,另一个从邻居获得,除了第一个 PE (PE0) 从 MMod 接收两个输入。内部 PE (PE3) 从两个相邻 PE 接收它们的输入。每个到 PE 的连接（除了即将介绍的 OMod 连接)都携带几个数据元素,这些数据元素等于数组的矢量化参数,或者换句话说,点积的宽度。图 5 显示了 PE 的内部工作原理。我们设计中的每个乘法器和加法器都对 32 位单精度浮点数据进行计算。点积的宽度在图中用 n 表示。我们选择了缩减树策略来有效利用 DSP 模块并允许对设计进行深度流水线化。 PE 还包括一个小型缓存,如图 5 中的移位寄存器 (SR) 所示。移位寄存器的大小基于交错因子（显示为 I） 每个周期,一个新向量进入树并与前一个向量一起累积存储在其中一个移位寄存器中的值。然后将此累加的输出路由到输出或移位寄存器的后面。多路分解选择器基于一个计数器,该计数器跟踪计算了多少部分和,当结果准备就绪时发出信号。一旦计数器滚动

978-1-7281-5020-8/19/$31.00 ©2019 IEEE

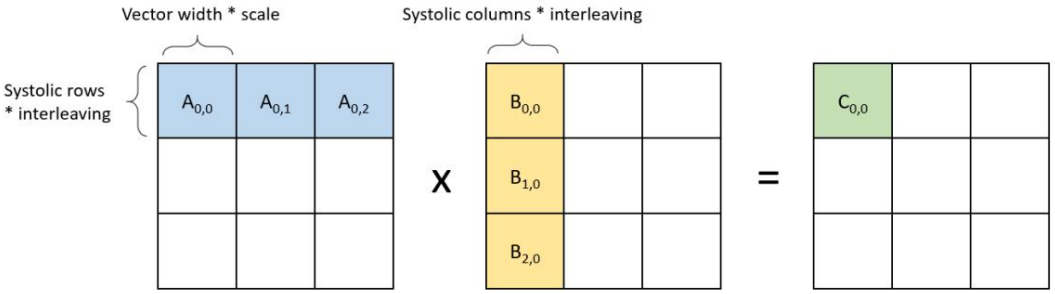

图 3. 矩阵分块示例。

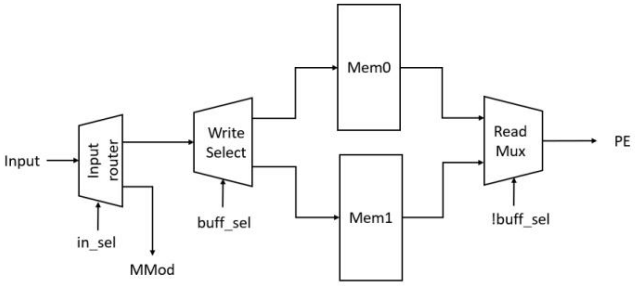

图 4. 内存模块内部结构。

结束时,漏极序列开始于将累积的结果路由出去并开始一个新的输出块序列。

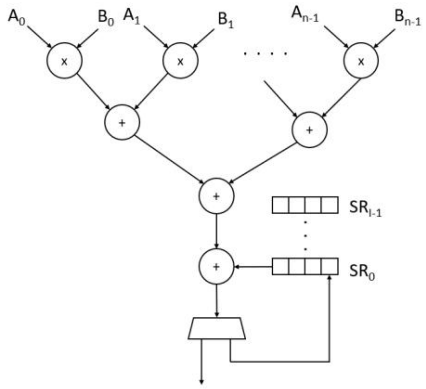

图 5. PE 内部结构。

D. 输出模块和全局漏极

一旦一个数据块准备好保存回全局内存,PE 就会开始耗尽过程,该过程首先将其缓存的内容写入其邻居。结果按行排出,因此每个 PE 沿其列排出。第一行 PE 连接到输出模块 (OMod),它们以菊花链方式相互连接（参见图 2）。 OMod 通过将结果传播到全局排水管来继续排水过程

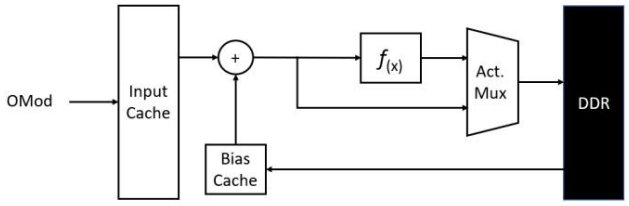

图 6. 全局排水内部结构。

职责是准备要写回全局内存的数据。被排出的数据以非连续的方式到达输出模块,因此全局排出有它自己的本地缓存,用于将数据缓冲回 DDR,请参见图 6。

虽然这是用于我们所有实验的基础设计,但全局消耗确实需要额外的内存资源,因此在扩展设计时,数据的重新排序可能需要在主机处理器上完成。通过全局内存寻址重新排序数据效率不高,因此我们总是以连续的方式写回数据。 global drain 支持传统 MLP 计算风格独有的一些附加功能。偏差和激活函数支持都包括在内,如果需要可以选择绕过。在偏置跳过的情况下,我们用全零预加载偏置缓存,并且从不从全局内存中读取。当使用偏差时,足够的偏差数据被预取到一个小的本地缓存中,以用于下一个排水序列。对于激活函数块,我们简单地使用激活多路复用器绕过,如图 6 所示。

E. 二维收缩阵列的 OpenCL 实现

前面部分中描述的每个模块都在 OpenCL 中作为单独的内核进行编码,并使用英特尔的 OpenCL 通道扩展进行连接。通过以下 C99 预处理器宏可以挂钩模块化设计:SYS ROWS、SYS COLS、SYS VEC、INTER LEAVE 和 SCALE。每个宏都会影响生成的硬件,因此,可以组合这些宏的各种排列来生成独特的硬件。

F. 硬件模型

ECADs HardwareDB Worker (HWDB) 被赋予前面部分描述的 SA 的排列。从...开始

由存储器类型及其功能（例如数据速率和模块数量)和具有目标Fmax的 FPGA 设备组成的加速器描述， EA 将向 HWDB 发送请求,其中包含 ANN 描述和返回的硬件配置所有支持指标的预测。在我们所有实验中返回的指标包括每秒图像数 (img/s)、延迟和有效千兆操作 (EGOP/s)。

五、评价

我们提出了一系列实验,展示了如何塑造 ANN 和硬件配置以适应不同的优化,并且最终一个解决方案无法满足所有设计搜索空间。首先,我们通过测量硬件并比较用投影对 MNIST 图像进行分类所需的总时间来验证硬件模型。然后,我们使用 ECAD 通过修改各种硬件参数来搜索优化,以了解它们对搜索的影响。我们考虑了四种优化:准确性、延迟、img/s 和每秒有效千兆操作 (EGOP/s)。我们用于所有硬件结果的加速器是英特尔的 Arria 10 GX 1150 FPGA 开发套件。这是一款具有 1 组 DDR4 内存的中档设备,提供 19.2 GB/s 的峰值带宽和 1.5 TFLOP/s 的峰值性能 [27]。在搜索期间,建模结果基于 250 MHz 的Fmax ,但为了进行测量比较,结果被归一化为硬件Fmax。

A. 仅硬件搜索

在没有准确性压力的情况下,EA 完全偏向于硬件,并且只会使用 ANN 参数来帮助调整加速器结果。运行此类实验的动机是深入了解 ANN 结构在某些性能优化下的外观,并了解硬件配置如何收敛。尽管搜索不准确,但每层可以使用的最大神经元数量仍然存在限制。本节中的所有结果都可以在表 I 中找到。我们搜索了一个 3 层多感知器网络,其中两个隐藏层分别由它们的神经元数量描述,如表中所示。在某些情况下,如使用 1 个 DDR 组优化 img/s 中所列,结果可能是带宽受限,这意味着性能会因每个周期可用数据与所需数据之间的比率而下降。正如我们将在下一节中看到的那样,尽管由于带宽限制而降级,但由于优化的适应性成功,一些配置仍将在进化过程中取得成功。对于这些情况,我们的性能建模仍然被发现是准确的。

Images per second img/s 的目标是找到一种加速器配置,该配置可以最快地处理单批图像,并且该配置将在一秒钟内处理最多的图像。当需要 ANN 的最高性能硬件时,此优化最有用。看资料,硬件配置

img/s 几乎相同,但 2 DDR bank 解决方案列出了更好的性能。带宽在这里发挥作用,因为单个库的 img/s 硬件配置受带宽限制。在这种情况下,通过为加速器提供更多的内存带宽,管道从潜在的停顿中得到缓解,从而获得更好的性能。

通常,从报告的数据可以看出,具有更多可用带宽的配置往往性能更好,因为更极端的配置不会像我们将在下一节延迟中看到的那样停止。

延迟在进化过程中,延迟返回为处理单个图像所需的时间量（以毫秒为单位）。当然,目标是尽量减少这个数字。与 img/s 非常相似,最佳延迟是通过更多可用带宽实现的。对于每个排列,EA 旨在通过减少配置参数来减小硬件的大小。 SA 设计中固有的某些参数有助于平衡带宽和计算,一旦这些参数降低到某个点,计算需求就会变得过多,带宽无法在不停止的情况下处理。最值得注意的是,对于较高带宽配置,交织因子能够减少到 2,而较低带宽配置的交织因子为 4。交错,处理元件在需要新数据之前保持忙碌的计算周期越多,但代价是这也会增加延迟。

有效的 GOP/s加速器的高效使用始终是人们所期望的,因为任何未使用的可用逻辑都可能被视为性能或成本的浪费。我们在搜索中包括 EGOP/s,以承受寻找更好利用加速器逻辑的设计的压力。每个设计都有潜在的千兆操作,直接基于硬件配置和 EGOP/s,基于映射到加速器的问题大小。

img/s 的目标是找到最佳的整体性能,而 EGOP/s 将在性能与逻辑区域的有效使用之间取得平衡。
如果功耗是一个问题,或者如果硬件只能为 NNA 加速分配这么多空间,则这种类型的搜索会很有用。使这些结果与众不同的主要是在神经元中可以看出,与 img/s 和延迟不同,如果关注准确性,它似乎更现实。然而,正如我们将看到的对准确性的压力,这种影响可能不会在准确性和神经元数量之间产生最佳权衡。为 EGOP/s 聚合的硬件配置在列、交错和规模方面也大得多,导致更高的延迟,这可能暗示两种优化之间的反比关系。

B. 具有模拟器精度的硬件搜索将模拟器带回进化过程增加了

ANN 精度的权重,因此结果成为 ANN 和硬件性能之间的组合。我们搜索了一个针对精度和 img/s 进行优化的 2 层多感知器网络。跟踪每一代可以跟踪、分类和分类进化过程

978-1-7281-5020-8/19/$31.00 ©2019 IEEE授权许可

表一

使用单组和双组 DDR内存进行纯硬件优化的EA结果

优化 # Banks Batch Size Neurons BW bound Latency (ms)						图像/秒	EGOP/s 行 Cols Vec 交错标度					
图像/秒		992	48; 180	是的	0.068352	1,243,182.00	119			16		
潜伏			18; 142	不	0.01792	111,607.00	4.0375					
EGOP/s		958	992； 1010	不	3.6495	47,508.98	170		16		16	30
图像/秒		992	62； 190 2;	不	0.06348	1,532,832.00	191			16		
潜伏		158	64 998；	不	0.008096	505,425.00	2.36					
EGOP/秒		960	1014	不	2.765	49,792.86	179		16		16	32

表二

优化精度和 IMG/S 的最佳排列

Top Permutation Fitness Sum Accuracy Batch size 神经元硬件配置。延迟（毫秒） 1.065							图像/秒 EGOP/秒129,056	
健身总和		0.936	508	852	2,8,32,16,2	0.352		174.61
准确性	1.006	0.942	484	1018	2,8,16,16,2	0.583	57,296人	92.62
延迟（毫秒）	1.065	0.936	508	852	2,8,32,16,2	0.352	129,056	174.61
图像/秒	1.064	0.935	572	970	2,16,32,32,2	1.283	129,144	198.93
EGOP/秒	1.062	0.933	572	980	2,16,32,32,2	1.283	129,144	200.98

评估。表 II 中列出了在该搜索过程中找到的最重要排列的结果。我们为其他指标添加了最佳排列（尽管没有明确优化)以提供与最佳表现者的对比。第 1 行显示了基于适应度和的最佳性能排列,该排列考虑了精度和 img/s 的加权累积。将此结果与准确性的最高排列进行对比,我们可以看到 0.6% 的准确性与 img/s 加速约 2.25 倍的轻微权衡。在上一节中,我们展示了延迟的结果以及 EA 如何减少 ANN 和硬件配置,但是,由于保持准确性的压力,这只能在有限的范围内完成,它恰好表明延迟表现最好与整体最佳表现者相同。

如果延迟是目标优化那么这可能不会有相同的结果。我们假设在精度压力和良好延迟下获得的结果将非常有效,因为目标是找到最小和最准确的设计。同样,比较 img/s 和 EGOP/s 的结果得出几乎相同的设计,不同之处在于 ANN。 EGOP/s 具有较低的适应度和,因为需要更多的神经元来更好地适应硬件,但事实证明,这是必要的,因为 ANN 训练过程可以用更少的 10 个神经元找到更好的准确性。

作为该实验的一部分,我们将几个表现最好的程序编译到硬件中并测量结果以验证硬件模型。表三包含测量结果

表 IV 包含用于比较的建模结果。测量的时间包括运行单个批次（大小见表 II)MNIST 图像的总执行时间。表中未列出但已验证准确性,即我们的加速器不会降低准确性。此外,以百分比形式提供了 ALM、SRAM 和 DSP 使用的资源细分。在进化过程中,HWDB（可选)不考虑 ALM,但它确实尝试淘汰无法适应必要的 SRAM 缓存的设计

并且能够确定设计所需的 DSP 块的确切数量。作为后处理和数据分析步骤的一部分,我们通过首先进行部分编译来确定哪些结果是可合成的,这些结果是表 IV 中的资源利用率。表现最好的产品都具有相似的配置,因此我们精心挑选了 4、8、8、16、18 配置作为对硬件模型的额外验证。表 IV 中的Fmax提醒我们,为了比较， Fmax已标准化为硬件。

表三

测量的硬件性能结果

硬件配置。时间 (ms)	ALM	SRAM	DSP	Fmax (MHz)		
2,8,16,16,2	9.64	37%	36% 26%		220	
2,8,32,16,2	5.53	50%	51% 43%		212	
4,8,8,16,18	4.65	38%	34% 26%		228	

表IV

建模的硬件性能结果

硬件配置。时间 (ms)	ALM	SRAM	DSP	Fmax (MHz)		
2,8,16,16,2	9.59	47%	38% 26%		220	
2,8,32,16,2	4.64	61%	55% 43%		212	
4,8,8,16,18	4.66	45%	37% 26%		228	

六。结论

我们提出了一种新的框架 (ECAD),用于基于进化算法探索和设计神经网络架构 (NNA)。我们使用可重构的硬件/软件协同设计方法在软件和硬件方面优化 NNA 设计。我们讨论并论证了优化 NNA 设计的可能性

978-1-7281-5020-8/19/$31.00 ©2019 IEEE授权许可

对于不同的和多个优化目标,而不是仅关注精度优化的主要方法。针对英特尔 FPGA Arria 10 1150 GX 设备的完整端到端 ECAD 流程提供了针对不同优化的顶级排列的性能数据,包括精度、img/s 以及两者同时进行。最后,将顶级排列的硬件性能与我们的模型进行比较,以验证进化过程的结果。

参考

[1]刘寒晓等。 “高效架构搜索的层次表示”。在:arXiv 预印本 arXiv:1711.00436 (2017)。

[2] Esteban Real 等人。 “图像分类器的大规模进化”。在:第 34 届机器学习国际会议论文集 - 第 70 卷。JMLR。组织。 2017 年,第 2902–2911 页。

[3] Esteban Real 等人。 “图像分类器架构搜索的正则化演化”。在:arXiv 预印本 arXiv:1802.01548 (2018)。

[4] 马丁阿巴迪等。 “Tensorflow:大规模机器学习系统”。在:第 12 届操作系统设计与实现研讨会 16。2016 年,第 265-283 页。

[5] 安东尼奥·古利和苏吉特·帕尔。使用 Keras 进行深度学习。 Packt 出版有限公司,2017 年。

[6] Alfredo Canziani、Adam Paszke 和 Eugenio Cu lurciello。 “用于实际应用的深度神经网络模型分析”。在:arXiv 预印本 arXiv:1605.07678 (2016)。

[7] Norman Jouppi 等人。 “第一个张量处理单元的动机和评估”。在:IEEE Micro 38.3 (2018),第 10-19 页。

[8] Alex Krizhevsky、Ilya Sutskever 和 Geoffrey E Hin ton。 “使用深度卷积神经网络的 Imagenet 分类”。在:神经信息处理系统的进展。 2012 年,第 1097–1105 页。

[9] Alex Krizhevsky 和 Geoffrey Hinton。从微小图像中学习多层特征。技术。代表 城市观察者,2009 年。

[10] Thomas Elsken、Jan Hendrik Metzen 和 Frank Hut ter。 “神经结构搜索:一项调查”。在:arXiv 预印本 arXiv:1808.05377 (2018)。

[11] Geoffrey F Miller、Peter M Todd 和 Shailesh U Hegde。 “使用遗传算法设计神经网络。”在:ICGA。卷。 89. 1989,第 379-384 页。

[12] 肯尼斯 O 斯坦利和里斯托 Miikkulainen。 “通过增强拓扑进化神经网络”。在:进化计算 10.2 (2002),第 99-127 页。

[13] Barret Zoph 等人。 “学习可扩展图像识别的可迁移架构”。在:2018 年 IEEE/CVF 计算机视觉和模式识别会议 (2018),第 8697–8710 页。

[14] Utku Aydonat 等人。 “Arria 10 上的 OpenCL(TM) 深度学习加速器”。在:CoRR abs/1701.03534 (2017)。 arXiv:1701.03534。网址: http://arxiv.org/abs/1701.03534。

[15] Naveen Suda 等人。 “用于大规模卷积神经网络的基于 OpenCL 的吞吐量优化 FPGA 加速器”。在:2016 年 ACM/SIGDA 现场可编程门阵列国际研讨会论文集。 FPGA 16。美国加利福尼亚州蒙特雷:ACM,2016 年,第 16-25 页。国际标准书号: 978-1-4503-3856-1。 DOI: 10。 1145/ 2847263。2847276。网址: http://doi.acm.org/10.1145/2847263.2847276。

[16] Jinhwan Park 和 Wonyong Sung. “仅使用片上存储器的基于 FPGA 的深度神经网络实现”。在:2016 年 IEEE 声学、语音和信号处理国际会议,ICASSP 2016,中国上海,2016 年 3 月 20 日至 25 日。2016 年,第 1011-1015 页。 DOI: 10.1109/ICASSP.2016.7471828。 网址: https://doi.org/10.1109/ICASSP.2016.7471828。

[17] Philip Colangelo 等人。 “使用英特尔 FPGA 探索低数值精度深度学习推理”。 在:CoRR abs/1806.11547 (2018)。 arXiv:1806.11547。 网址: http://arxiv.org/abs/1806.11547。

[18] Yaman Umuroglu 等人。 “FINN:快速、可扩展的二值化神经网络推理框架”。在:CoRR abs/1612.07119 (2016)。 arXiv:1612.07119。网址: http://arxiv.org/abs/1612.07119。

[19] Ganesh Venkatesh、Eriko Nurvitadhi 和 Debbie Marr。 “使用低精度和稀疏性加速深度卷积网络”。在:2017 年 IEEE 声学、语音和信号处理国际会议,ICASSP 2017,美国路易斯安那州新奥尔良,2017 年 3 月 5 日至 9 日。2017 年,第 2861-2865 页。 DOI: 10。 1109/ICASSP。 2017.7952679.网址: https://doi.org/10.1109/ICASSP。 2017.7952679.

[20] J.英尔等。 “用于深度神经网络的可定制 FPGA OpenCL 矩阵乘法设计模板”。在:2017 年现场可编程技术国际会议 (ICFPT)。 2017 年 12 月,第 259-262 页。 DOI: 10。 1109/FPT.2017.8280155。

[21] Eriko Nurvitadhi 等人。 “用于英特尔®Stratix®10 FPGA 的封装内领域特定 ASIC:使用 TensorTile ASIC 加速深度学习的案例研究(仅摘要)”。在:2018 年 ACM/SIGDA 现场可编程门阵列国际研讨会论文集。 FPGA 18。美国加利福尼亚州蒙特雷:ACM,2018 年,第 287–287 页。国际标准书号: 978-1-4503-5614-5。 DOI: 10。1145 / 3174243。3174966。 网址: http://doi.acm.org/10.1145/3174243.3174966。

[22] Stylianos I Venieris、Alexandros Kouris 和 Christos Savvas Bouganis。 “用于在 FPGA 上映射卷积神经网络的工具流:调查和未来方向”。在:ACM 计算调查 (CSUR) 51.3 (2018),p. 56.

[23] 大卫 E 戈德堡和 Kalyanmoy Deb。 “遗传算法中使用的选择方案的比较分析

算术”。在:遗传算法的基础。卷。 1.
爱思唯尔,1991 年,第 69-93 页。
[24] 托马斯·劳伯和古杜拉·朗格。 ¨ 并行编程:用于多核和集群系统。施普林
格科学与商业媒体,2013 年。

[25] 埃德加·加布里埃尔等人。 “开放 MPI:下一代 MPI 实现的目标、概念和
设计”。在:欧洲并行虚拟机/消息传递接口用户组会议。施普林格。
2004 年,第 97-104 页。

[26] HT Kung 和 Charles E Leiserson。 “收缩阵列(用于 VLSI)”。在:
稀疏矩阵会议记录 1978 年。卷。 1.
工业和应用数学学会。 1979 年,第 256-282 页。

[27] 英特尔 Arria10 设备概述。 2018.网址: https://intelcom/content/
www/us/en/products/www。
可编程/fpga/arria-10/features.html。

978-1-7281-5020-8/19/$31.00 ©2019 IEEE授权许可