基于 FPGA 的机器学习硬件加速研究进展

王 超 王 腾 马 翔 周学海
(中国科学技术大学计算机学院 合肥 230027)

摘 要 随着日益剧增的海量数据信息的产生以及数据挖掘算法的广泛应用,人们已经进入了大数据时代.在数据规模飞速增长的前提下,如何高效稳定的存取数据信息以及加快数据挖掘算法的执行已经成为学术界和工业界急需解决的关键问题.机器学习算法作为数据挖掘应用的核心组成部分,吸引了越来越多研究者的关注,而利用新型的软硬件手段来加速机器学习算法已经成为了目前的研究热点之一.本文主要针对基于 ASIC 和 FPGA 等硬件平台设计的机器学习加速器进行了归纳与总结.首先,本文先介绍了机器学习算法,对代表性的算法进行了分析和归纳.接下来对加速器可能的着眼点进行了列举综述,以各种机器学习硬件加速器为主要实例介绍了目前主流的加速器设计和实现,并围绕加速器结构进行简单分类和总结.最后本文对机器学习算法硬件加速这个领域进行了分析,并对目前的发展趋势做出了展望.

关键词 机器学习;FPGA;加速器;大数据;神经网络加速器
中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2020.01161

Research Progress on FPGA-based Machine Learning Hardware Acceleration

WANG Chao WANG Teng MA Xiang ZHOU Xue-Hai
(*School of Computer Science and Technology, University of Science and Technology of China, Hefei* 230027)

Abstract With the increasing production of massive data information and the widespread use of data mining applications in the fields of voice, languages, image, video, etc., people have entered the era of big data. In such an era, how to access data information efficiently and steadily and how to speed up the implementation of data mining applications have become the key issues that need to be solved urgently in academia and industry. And the machine learning algorithms, as the core component of data mining applications, have attracted more and more researchers' attention to applicate in various fields. Therefore, using existing hardware and software means to accelerate machine learning algorithms has become a research hotspot. In this study stocks boom, the current acceleration platforms can be summarized into four categories respectively: the custom logic circuits (such as FPGA/ASIC), the general graphics processing unit (GPGPU), the cloud computing platform and the heterogeneous computing platform. These acceleration platforms often show different parallel granularity and are suitable for different application scenarios. However, this is also a choice that they are combined to form heterogeneous systems to give full play to the processing capabilities of different acceleration devices. But, due to the customizable and high energy-efficient, the FPGA-based hardware acceleration is becoming a hot choice as the machine learning acceleration. Therefore, this paper mainly focuses on the field of machine learning algorithm accelerator based on FPGA. First of all, the paper introduces the machine learning algorithms and background knowledge in chapter 1&2. And then, we give the current

收稿日期:2019-02-27;在线出版日期:2020-04-20.本课题得到国家重点研发计划(2017YFA0700900)、中国科学院青年创新促进会(2017497)资助.**王 超**,博士,副教授,中国计算机学会(CCF)高级会员,主要研究方向为深度学习处理器.E-mail:cswang@ustc.edu.cn.**王 腾**,硕士,主要研究方向为神经网络加速器.**马 翔**,硕士,主要研究方向为神经网络加速器.**周学海**,博士,教授,主要研究领域为嵌入式系统、高性能计算.

development in this field, which composed of three parts (the methods for acceleration, the hardware platforms for acceleration and evaluations for accelerators) in chapter 3. And we propose an overview of the possible points of the accelerator with four sections. There are accelerating the kernel in the algorithm, abstracting the common feature in the algorithm, parallelizing algorithm and optimizing the data communication. After that, in chapter 4, the design and implementation of the current mainstream accelerators are introduced with four kinds of examples of various hardware machine learning accelerators, which is in order of specific issues, specific algorithm, common feature, and hardware template. Meanwhile, the structure of the design of accelerators is also simply classified and summarized in this chapter. The current hardware accelerator design is divided into two types, Stream and Single Engine. The characteristic of the Stream model is optimized for each calculation process in various hardware blocks and paralleled in the pipeline to achieve high performance. And the characteristic of the Single Engine model is focused on the common feature between all calculation process. Therefore, the Single Engine model could get larger and more hardware blocks than the Stream model to get high computation performance and compatibility. Finally, this paper summarizes the field of hardware-accelerated machine learning algorithms and puts forward the research direction and development trend in six points. In summary, this paper summarizes the current status of the development of machine learning accelerators and pointed out the future direction of development in the analysis of the current accelerator method.

Keywords machine learning; Field-Programmable Gate Array; accelerator; big data; neural network accelerators

1 引 言

随着日益剧增的海量数据信息的快速增长以及数据挖掘领域应用的广泛,人们已经进入了大数据时代.而在这样快速发展的大数据时代浪潮下,不少问题同样困扰着计算机设计人员.如何高效稳定的存取数据信息以及加快数据挖掘应用的执行已经成为学术界和工业界急需解决的关键问题.其中,机器学习算法作为数据挖掘应用的核心组成部分,吸引了越来越多的研究者的关注,而利用现有的软硬件手段来加速机器学习算法已经成为当下的研究热点.

然而,在大数据时代下加速机器学习算法面临许多新的挑战.在这样的环境下,有很多因素使得越来越多的用户放弃了原有的基于 CPU 的单节点处理平台而转向利用其它平台和手段来加速数据挖掘/机器学习应用的执行.图 1 展示了整个加速硬件的发展汇总.而目前发展的契机[1]主要有以下几点:

(1)海量数据.很多应用领域的潜在的数据规模极其庞大,这使单机处理数据较为困难;

(2)数据高维度.某些数据挖掘应用中,实例数据的特征数量繁多,机器学习算法为了处理这些数据可能需要对数据特征进行分割;

(3)模型和算法复杂.一些高精度的机器学习

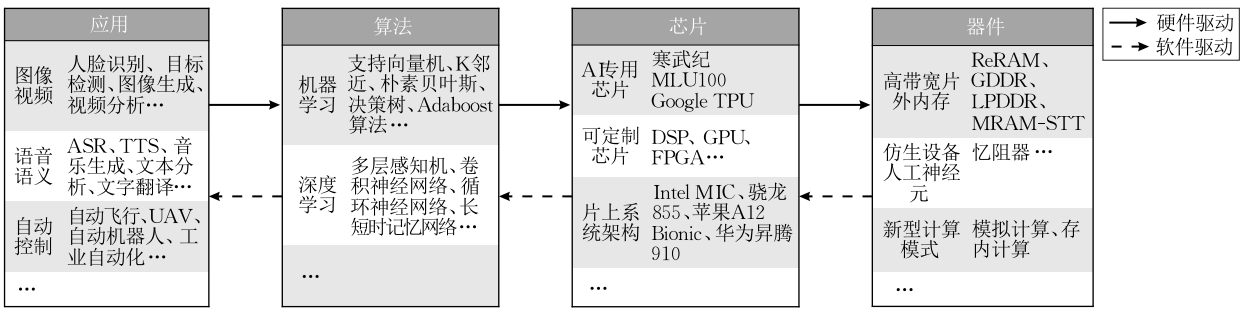


图 1 面向领域的计算加速器要素间关系

算法通常有着较为复杂的模型表示，并往往需要大量的数据计算；

(4) 推理时间约束. 某些数据挖掘应用如语音识别、视觉物体探测等有着实时性的要求，使得单机处理无法满足特定应用的需求；

(5) 多级预测. 某些机器学习算法能够表示成多级管道的形式，管道中的多级分类器需要并行工作，而单节点 CPU 处理平台往往无法满足这一需求.

现今，学术界和工业界存在着多种较为成熟的加速平台，研究者可以利用这些平台能够较好处理海量数据以及实现较高效率的机器学习算法. 而这些加速平台可以概括为四类，它们分别是：Intel 集成众核 MIC、专用集成电路 ASIC、通用图形处理单元(GPGPU)、云计算平台、可重构逻辑电路如 FPGA 以及各种异构计算平台. 这些加速平台往往表现出不同的并行粒度，适用于不同的应用场景，并且也能够相互结合形成异构系统来充分发挥不同加速器件的处理能力.

同时，加速平台无法仅依靠硬件系统运行，同时还需要一系列配套的软件系统作为支撑. 目前也存在多种软件与中间件系统，适用于不同的加速平台中，例如适用于云计算平台的 Hadoop、Spark、DryadLINQ、Pregel 以及 PowerGraph[2] 等，适用于 GPGPU 平台的 CUDA、OpenCL 和 OpenACC 等. 这些软件系统既充分利用了加速平台的能力又方便用户编程和使用，利用这些软件系统，用户只需要按照相应的规范和利用提供的接口来编写软件应用就能够获得较为可观的加速效果.

云计算平台以及通用图形处理器(GPGPU)是目前使用比较广泛的通用加速平台；FPGA 与 ASIC 则往往则用于对特定的问题实现特定的加速器来实现硬件加速；而对于将 CPU、GPU 和 FPGA 相结合的异构计算平台，如 Axel、OptiML 和 Lime 等，虽然它们在理论上有较大的加速潜力，但由于实现难度等问题，目前多处于研究阶段，还存在着很多挑战性问题亟待解决. 就并行方式而言，云计算平台主要依赖大规模基于 CPU 节点的计算集群来实现，它主要利用粗粒度的任务级并行来加速应用执行；GPGPU 则主要利用了细粒度的数据级并行；FPGA/ASIC 主要利用了细粒度的数据级并行以及管道流水线的方式来加速应用，可以用在边缘计算等场景中[3]. 从软件系统来说，云计算平台主要包括了基于 Map-Reduce 编程模型的 Hadoop、Spark 等以及基于图计算的编程模型 Pregel、PowerGraph 等；GPGPU 的软件系统则涵盖了基于 SPMD 的 CUDA、OpenCL 和 OpenACC 等；

而对于 FPGA/ASIC 等加速器结构，目前有一些相对成熟的深度学习自动代码生成方法如 TVM[4] 等，可以为不同硬件后端的深度学习工作负载提供可移植性，并提供一系列的优化策略，比如高级算子融合，硬件原语映射以及内存延迟覆盖等. 同时也有多家企业展开了深度学习开发平台的建设，其中基于 FPGA 的包括百度昆仑 AI、深鉴科技的 DNNDK 和微软的 Brainwave Project 等. 而 ASIC 方面的硬件加速器有谷歌 TPU[5]、寒武纪思元系列[6]、海思科技昇腾 910、高通骁龙 855 和苹果的 A12 Bionic 等不仅有服务端更有移动端的芯片，足以说明了目前各厂商对于硬件加速器的重视程度.

如表 1 所示，我们对各个硬件平台的开发情况做出了对比[7]，对于云计算平台和各种通用处理器的应用情况，目前所使用的 CPU 由于机器学习算法其数据密集型与计算密集型等的特性使得它处理这些问题时的性能较低而且耗能更高. 然而由多 CPU 构成的云计算平台的数据通信开销也成为了阻碍效率提升的绊脚石；而 GPU 在处理数据关联程度比较高的数据时无法获得较好的计算效率，并伴随着较大的功耗；ASIC 限于其开发难度和开发周期等问题在加速器设计的早期无法快速开展. 故因此利用 FPGA 来设计用于机器学习算法的加速器体系结构并以此构建整套开发体系是一个逐渐发力的科研方向.

表 1　不同硬件的开发情况对比[7]

硬件类型	理论性能	平均能耗	软件支持	编程难度	开发周期
CPU	低	中	好	容易	短
GPU	高	高	好	中等	短
MIC	中	高	少	容易	中
FPGA	高	低	少	难	较长
ASIC	高	低	少	难	长

本文主要针对利用 FPGA 设计专用于机器学习算法加速器领域进行了分析与总结. 本文第 2 节主要介绍相关的背景知识，对相关机器学习算法做概述；第 3 节当前发展现状中则主要对目前各种加速器和加速手段进行介绍，并对可能的加速着眼点进行列举；第 4 节硬件加速器的设计则主要对相关的机器学习硬件加速器的结构设计和实现进行简单的分类和总结；最后的第 5、6 节展望和总结则是在基于全文的情况下对硬件加速机器学习算法这个领域进行归纳总结，并简要提出了潜在的研究方向及发展趋势.

2 背景介绍

2.1 机器学习概述

机器学习关注如何利用数据来构造出相应的预测模型来对未知的数据进行预测.机器学习的主要任务就是从某类函数模型中选出一种函数 f 并根据数据集进行学习使得该函数能够较为准确的将输入域 X 映射到输出域 Y 中,即 $f:X\rightarrow Y$.输入域 X 往往代表多组数据构成的集合,输出域 Y 则代表每组数据对应的标识或结果.

根据用来学习的数据类型不同,机器学习算法可以分为监督学习和非监督学习两类,例如决策树、随机森林、支持向量机等监督类学习和 K-Means 与隐马尔可夫模型等非监督学习.同时又可分为连续和离散两类,例如常见的线性回归、SVR 等连续数据学习和 KNN、朴素贝叶斯等离散数据学习.

在监督学习算法中,用于训练的数据集中的每组训练数据都有一个明确的标识或结果,算法则利用训练数据构造出一个函数 f,而该函数 f 将用来对未知标识或结果的数据来做预测.在非监督学习中,已有的输入数据集合的标识或结果往往是未知的,大多数的非监督学习算法都假定数据是服从某种联合概率分布,算法利用该假定来寻找出最贴合输入训练数据的函数 f.

监督学习主要分为分类和回归两类任务,在分类中,函数 f 的输出域 Y 由一组离散的值构成;在回归中,函数 f 的输出域 Y 则是连续的实数.而非监督学习则主要进行数据聚类,数据聚类就是把没有分类的数据集上的数据按照距离、相似度等属性来归为若干类.

监督学习和非监督学习都需要学习(Learning)和推理(Inference)这两个阶段.学习是指确定预测函数 f 的过程,推理则指根据 X 上的某一个实例 x 来计算 $f(x)$ 结果的过程.因此,对于机器学习算法,我们可以根据具体的应用场景来选择具体针对学习过程或是推理过程来进行加速.

此外,如果根据算法本身的特性来分,机器学习算法还可以分为批量学习(Batch Learning)和在线学习(Online Learning)两种形式.批量学习是指传统意义上的学习方式,即先给出一个训练集,训练出 f 后,再将 f 运用于测试数据;而在线学习则不同于传统的学习方式,它是一种边学习边对数据进行预测的过程,因此在线学习往往对实时性要求较高,对在线学习算法进行加速往往比对批量学习算法进行加速显得更加重要.

2.2 FPGA 介绍

现场可编程门阵列(Field-Programmable Gate Array)即 FPGA,是在经历了 PAL、GAL、EPLD、CPLD 等可编程硬件后发展出的硬件设备.FPGA 一开始是作为 ASIC 领域中的一种半定制电路芯片而产生的,由于其克服了定制电路无法快速修改的不足,而且也避免了以前可编程器件门电路的缺点,因此采用 FPGA 来快速搭建领域专用的计算系统成为芯片设计和验证平台的主要技术手段.

对于 FPGA 来说,可重构性是其能实现复杂逻辑的关键特性.与 ASIC 中集成的固定逻辑不同,FPGA 利用了基于 SRAM 的查找表(LUT)来实现硬件逻辑的配置.FPGA 芯片主要由 6 个部分组成,分别是可编程输入输出单元(IOB)、可配置逻辑块(CLB)、数字时钟管理模块(DCM)、嵌入式块 RAM(BRAM)、互联互通资源和底层内嵌功能单元以及专用硬核,其中查找表(LUT)就是 CLB 的一部分.而 FPGA 的工作原理是利用已经编译过的硬件程序控制内部逻辑单元和各模块间与输入输出单元中的互联形式,实现不同的逻辑功能.

由于 FPGA 具备快速定制性和可重构等特性,使其在目前越来越复杂的计算机体系结构设计,特别是面向领域的专用平台设计与实现中崭露头角.通过基于 FPGA 软硬件平台重新编译和仿真,研究人员对机器学习算法的加速器进行快速实现和验证,大大提高了加速器的设计效率,从而使得基于 FPGA 的机器学习加速器成为目前的研究热点之一.

3 当前发展现状

3.1 目前的加速手段

从研究者的角度来看,目前对机器学习算法进行加速的手段可以大致分为三大类,即软件层次上的优化、机器学习算法的并行化和硬件层次上的改进.

软件层次上的优化主要包括的是对机器学习算法本身进行优化改进、对算法运行时库环境等的优化改进等.对机器学习算法本身进行改进是指对某个算法提出新的数学模型等来提高算法的执行速度,如针对 SVM 算法的 SMO 方法的提出等;对算法运行时库环境的优化则指的是对算法运行时处于的软件环境,如运行时库、操作系统等进行进一步地

优化以提升执行机器学习算法的效率.

并行化机器学习算法则是目前最普遍的加速手段,它主要是对机器学习算法进行并行化与分布式处理使得算法本身能在特定的硬件并行平台上实现任务级并行和数据级并行等.很多的机器学习算法能够相对简单地进行并行化处理并且能够很好运行在多核多节点的硬件平台上,如云计算平台或者 GPU 等[8].

硬件层次上的改进则主要是指针对机器学习算法的特征来改进现有的处理器体系结构使其能够高效快速地执行机器学习算法.然而目前的通用 CPU 的体系结构并不适合于处理机器学习问题,其主要原因是机器学习算法自身的 3 个特征,即数据密集型与计算密集型的结合、流式数据传输与迭代计算和较低的分支指令等[9].机器学习算法自身是数据密集型与计算密集型的综合,这使得机器学习算法往往既需要频繁访存来获取大量数据又需要对数据进行高强度大规模的复杂运算,而 CPU 的访存效率以及计算能力往往无法满足大规模机器学习应用的要求;机器学习算法一般以流的方式顺序读入数据并进行处理,并且往往有着以整个数据集为单位的迭代计算,即某项数据一次处理完成后往往需要整个数据集处理完成一次后才能进行下一次计算,这些都会导致基于 LRU 策略的 CPU Cache Miss 的比率很高,引起整个算法执行效率较低;而分支指令在机器学习算法中往往所占比例较低,这反映出整个算法过程相对而言具备顺序执行的特征,同时也说明 CPU 中分支预测部件利用率不足.

3.2　目前的加速平台

目前对于机器学习算法的加速平台主要分为 4 类,它们分别是云计算平台、通用图形处理器(GPGPU)平台、FPGA/ASIC 平台以及综合了上述 3 种平台特性的异构计算平台.这些平台往往表现出了不同的并行粒度,并且适用于不同的机器学习问题.

云计算平台是目前普及最广的平台,利用云计算平台可以比较方便的对数据进行分布式处理和并行化机器学习算法.云计算平台一般都由大量同构的基于 CPU 的单节点服务器构成,多个节点间互相配合、协同工作,并且可以对问题采用任务级并行与数据级并行的手段.云计算平台编程模型大体上可以分为基于 Map-Reduce 编程模型和基于图计算编程模型 2 种,采用 Map-Reduce 编程模型的程序可以抽象成 Map 和 Reduce 两个阶段,这种模型比较适合处理依赖程度比较低的数据;采用图计算编程模型的程序可以抽象成基于一个图的计算,每个图的节点都根据相邻边和节点的信息进行计算,这种模型比较适合于数据相互依赖程度比较高的情况[10].同时还有另一种由 Map-Reduce 编程模型发展出来的计算引擎 Spark.它是由 UC Berkeley AMP lab 所开源的类 Hadoop MapReduce 的通用并行框架①.该框架拥有 Hadoop MapReduce 所具有的优点;同时不同于 MapReduce 模型的,Spark 将任务的中间输出结果保存在内存中来降低读写 HDFS 文件系统需求.因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 算法.

通用图形处理器由于其自身的特殊结构使得它能够很好的对数据进行数据级并行处理.通用处理器内部往往由多个 SM 构成,每个 SM 由多个 SP 组成;多个 SM 共享一个全局内存,同一个 SM 的 SP 共享多个寄存器和共享内存.本质上而言,GPGPU 相当于一个众核的架构,并且其不同层次的内存器件并不像 CPU 那样自动维护,而是由程序员来指定,因此 GPGPU 能够很好地对问题进行数据级并行,并且 CUDA、OpenCL 和 OpenACC 等编程规范的提出和实现使得针对 GPU 编程变得简单快捷,因此 GPU 也成为了目前也较为广泛使用的加速并行平台.

FPGA 与 ASIC 目前主要用于针对特定的算法和问题本身去设计专用的硬件加速器件.由于其定制的原因,往往能获得更好的性能和功耗.由于 FPGA 通常是一个用于对设计出的加速器体系结构进行验证仿真的中间器件,当验证完成后,即可实现专门的 ASIC 加速器芯片.而 FPGA 本身由于其灵活的可编程与可重构的特性使得它也可以充当一个专门的加速器件,对于不同的问题来进行最贴合的重构使得 FPGA 有着很大的加速潜力.但是,FPGA 与 ASIC 平台由于设计难度和编程门槛较高等因素使得其尚不能广泛普及,它们主要存在于嵌入式设备、云数据中心、各类仪器以及大型通信设备中或者特定领域的应用当中.

此外,还有一些异构计算平台[11]综合利用了 CPU、GPU 和 FPGA,并且往往也采用了由异构计算节点构成的集群的方案.不过这种异构计算平台还存在如何充分利用计算资源以及怎样为用户提供

① Apache Spark. http://spark.apache.org/docs/latest/

简洁的编程模型等的问题，目前尚不成熟，仍处于研究阶段. 现有的一些异构计算平台的原型有 Axel[12]、OptiML[13]和 Lime[14]等.

同时各个相关企业也提出了很多的商用机器学习算法加速平台，比如阿里巴巴的 PAI 3.0 等. PAI 计算平台是阿里巴巴公司于 2015 年正式对外提供服务的计算平台，通过利用 PAI 服务将用户的算法快速构建，并通过 PAI 核心引擎部分将算法描述高效地编译优化调度成为底层硬件需要执行的指令代码，最后利用硬件层中 CPU、GPU、FPGA 或人工智能芯片 ASIC 等异构资源经过软件以及调度把合适的图计算模型进行有效的分割，将不同的计算部署到比较合适的硬件结构上，从而实现高效的计算能力. 最新的 3.0 版本增加了深度学习编译器 TAO (Tensor Accelerator and Optimizer)，以通用化、平台化的方式来解决上层 Workload 与底层硬件计算单元之间高效映射的问题[15]. 其利用一种基于沿图的跨度(关键路径)的分层节点结构的新型深度融合算法[15]. 使用关键路径缩减作为驱动启发式，不仅考虑 producer/consumer 融合变换，而且考虑发生在同一层中的细粒度操作，显著减少硬件访存开销以及硬件、框架层面的调度执行开销，并且引入了更广阔的全局优化空间，从而完成了算法加速的目的.

3.3 衡量加速效果的指标

衡量加速平台加速效果可以基于很多不同的指标，这些指标往往反映了加速平台各个不同的方面，现列举常见的一些指标，即加速比、效率、可扩放性和资源利用率.

加速比(Speedup)是指程序的串行版本运行的时间与程序的并行版本运行的时间的比值. 只有在比值大于 1 的时候，对程序进行并行化处理才有意义，而且比值越大就说明了对程序的并行化有着较高的加速效果.

效率(Efficiency)一般指程序的加速比与处理单元数量的比值，它通常反映了多个处理单元的利用率的情况，效率越高，多个处理单元的利用率就比较高.

可扩展性(Scalability)则是描述了程序随着处理单元数量的增加效率值的波动情况. 而可扩放性通常和效率有一定关联，效率越高，程序的可扩放性越好，反之亦然.

资源利用率则主要针对利用 FPGA 或者 ASIC 的硬件平台进行加速的情况. 在利用 FPGA 或者 ASIC 等平台设计硬件加速器结构时，芯片能够提供的资源一般是很有限的，因此在设计时不能一味的通过增加硬件资源来提升性能，而是需要在资源和性能间进行综合考虑来寻求平衡.

3.4 加速算法的着眼点

加速机器学习算法的执行需要有针对性的对算法进行分析，进而确定需要加速的算法具体部分. 机器学习算法由于其兼容了数据密集型和计算密集型的特性，因此对机器学习算法的加速需要兼顾考虑加速数据通信传输以及加速算法计算执行两个方面.

根据现有的机器学习算法加速的相关文献总结，大体可以分为 4 个方面：即加速算法的计算核心、抽象算法的共性特征、并行化机器学习算法以及优化机器学习算法的数据通信传输. 加速算法的计算核心与并行化机器学习算法属于加速算法计算执行的维度；优化算法的数据通信传输则属于加速数据通信传输的范畴；而抽象算法的共性特征并对特征加速则都包含这二方面的内容. 上述 4 个着眼点之间联系比较紧密，需要在设计专用加速器时进行协同考虑. 例如优化数据通信传输也是抽象并加速机器学习算法共性特征的一个特例；而并行化机器学习算法可以首先提取算法的计算核心模块，而后有针对性地进行并行化处理. 在算法的分析和抽象过程中，提取出的算法的共性特征未必是算法的计算核心模块. 根据 Amdahl 定律，针对非计算核心的共性特征来进行加速，最终获得的系统加速比往往性能提升效果不明显. 总体来说，基于上述 4 个着眼点，可以帮助并指导研究人员展开对机器学习算法的分析和加速器的硬件设计工作.

3.4.1 加速算法计算核心

由于机器学习算法种类多样，在执行过程中算法的特征明显，其中核心计算单元对整个算法的执行时间的影响的比重也有较大的差异性，需要重点分析算法的计算核心部分.

通常来说，算法的计算核心(Kernel) 是指算法最耗时间的那部分计算过程，而加速 Kernel 则能够显著缩短整个算法的执行时间. 因此对于 Kernel，我们既可以利用如 GPGPU 的多个计算单元对不同的数据进行并行计算，也可以利用 FPGA 对算法的 Kernel 固化到多个计算单元上来加快执行. 表 2 列举了 15 种常见机器学习算法最耗时的前 3 个 Kernel[9]. 机器学习算法在执行过程中算法的不同部分对整个算法的执行时间影响的比重各不相同.

表 2　各机器学习算法计算核心 TOP3[9]

Application	Top Three Kernel/%			Sum/%
	Kernel1	Kernel2	Kernel3	
K-Means	Distance (68)	Clustering (21)	minDist (10)	99
Fuzzy *K*-Means	Clustering (58)	Distance (39)	fuzzySum (1)	98
BIRCH	Distance (54)	Variance (22)	Redistribution (10)	86
HOP	Density (39)	Search (30)	Gather (23)	92
Naïve Bayesian	ProbCal (49)	Variance (38)	dataRead (10)	97
ScalParC	Classify (37)	giniCalc (36)	Compare (24)	97
Apriori	Subset (58)	dataRead (14)	Increment (8)	80
Eclat	Intersect (39)	addClass (23)	invertClass (10)	71
SNP	CompScore (68)	updateScore (20)	familyScore (2)	90
GeneNet	CondProb (55)	updateScore (31)	familyScore (9)	95
SEMPHY	BestBrnchLen (59)	Expectation (39)	lenOpt (1)	99
Rsearch	Covariance (90)	Histogram (6)	dbRead (3)	99
SVM-RFE	quotMatrx (57)	quadGrad (38)	quotUpdate (2)	97
PLSA	pathGridAssgn (51)	fillGridCache (34)	backPathFind (14)	99
Utility	dataRead (46)	Subsequence (29)	Main (23)	98

3.4.2　抽象算法共性特征

虽然目前机器学习算法种类多样，但同时许多机器学习算法都表现出一些共性的特征，针对这些共性特征进行加速既能做到较好的加速效果又能表现出相对通用的特性[16]. 机器学习算法的共性特征可以大致概括为 5 个方面，即大规模线性代数运算、同步/异步迭代运算、算法加乘化、常用激励函数的使用和基于图模型的抽象.

大规模线性代数运算指的是大部分的机器学习算法往往都涉及了大量的大规模线性代数运算，加速这些运算的执行能提升整个算法的性能. 文献[17]设计出了一个加快矩阵相乘运算的加速器件，并在多种机器学习算法上取得了较好的加速效果. 同步/异步迭代运算是指很多机器学习算法需要在算法中反复对数据进行同步/异步迭代，对迭代算法进行优化能够显著改善算法性能. 文献[18]设计出了一个基于 FPGA 的异步迭代加速器结构，利用该加速器可以加速多种机器学习算法的执行. 而对于算法中显式需要迭代的 LSTM，也可以通过在加速器的计算核心中加入隐层单元储存隐层数据从而进行快速迭代计算[19-20]. 算法加乘化是由文献[3]提出，主要是指一部分机器学习算法在学习或推理过程中通常表示成相乘-累加的形式，每次相乘对应的数据的依赖程度一般较低，因此对于这种情况可以方便的对算法进行并行化处理，目前是各类加速器研究的优化手段之一[19-23].

激励函数是多种机器学习算法在执行至某特定步骤时均会采用的辅助函数，如 Sigmoid 函数等，针对这些常用激励函数进行加速可以取得一定的加速效果.

基于图模型的抽象由文献[24]提出，表明了图计算模型能够较好有效地处理那些数据间依赖程度较高的数据挖掘算法，因此可以对将数据抽象为图，然后进行基于图的顶点计算的这一过程进行加速. 而且还有团队利用图的跨度的分层节点结构，从而优化计算流程进行加速.

需要注意的是，正如前面所提到的，从多个算法中抽象出来的共性特征可能属于这些算法的计算核心的某部分，也可能不是. 因此，如果抽象出的特征在很多算法中都是计算核心，那么我们去加速这个特征的执行就相对收益较大；而反之，如果抽象的特征在大多数的算法中耗时较短，则针对该特征去设计加速器结构最终并不能获得可观的系统加速比.

3.4.3　机器学习算法并行化

对机器学习算法并行化是目前常用的加速手段，利用任务级并行或者数据级并行，或者将二者混合可以对绝大多数的机器学习算法进行并行处理[25].

目前并行化机器学习算法的平台主要是云计算平台、通用图形处理器以及 FPGA/ASIC 平台三大类. 利用云计算平台并行主要利用了任务级并行和数据级并行，并且并行粒度相对较粗，例如 Map-Reduce 模型中的 Map 和 Reduce 过程即可并行执行[26]，而图计算模型中没有依赖关系的顶点间也可进行并行执行[15]；利用 GPGPU 平台并行则主要利用了数据级并行的方式，并行粒度相对较细；利用 FPGA/ASIC 平台并行则主要取决于设计出的加速器体系结构的不同，既可以利用任务级并行，也可以利用数据级并行[27]. 另外，在硬件加速器中一般也都会采用了管道流水线的技术来增加吞吐率[21]. 而

在文献[28]中则根据神经网络模型的上下层参数动态选择并行方式,有效减少了训练时数据通信带来的开销成本.同时,BitFusion[29]加速器则是对神经网络参数进行了预先划分,根据输入数据的 Bit 范围动态调节计算核心的尺寸参数,充分利用了计算硬件的并行性.

3.4.4 优化数据通信传输

由于机器学习算法是兼具计算密集型和数据密集型的特性,所以单纯对计算密集型的部分进行加速是远远不够的,算法的访存等数据通信往往会成为提升性能的瓶颈.而针对机器学习算法优化数据通信传输以及访存模型等是属于针对机器学习算法数据密集型的部分的加速入手点.

目前现有的三种加速平台都不同程度上面临着数据通信等的问题.对于云计算平台来说,虽然其提供了大规模的计算和存储资源,然而在云平台上对某些机器学习算法进行并行加速,其效果往往不甚理想,通常原因在于数据通信带来的巨大开销.云计算平台利用分布式文件系统对数据进行存储,单个节点间通过以太网连接.如果算法需要的数据分布到多个节点中,或者算法需要较频繁存取数据,由此带来的数据传输通信开销就会比较可观,从而降低整体的加速性能.对于通用图形处理器(GPGPU),利用它来加速机器学习算法同样也需要考虑数据传输问题.程序所需要的数据往往存放在节点中的磁盘上,经由内存传输到了 GPGPU 的全局内存中,这个过程会占据大量的时间开销.另外,GPGPU 的内部也存在寄存器、共享内存、L1 Cache 等不同层次的内存模型,因此利用 GPGPU 并行化算法时需要着重考虑这些不同的存储部件的使用方式.而基于 FPGA 与 ASIC 设计专用加速器往往也面临着数据从 Host 内存传输到 Device 内存的过程.并且由于 FPGA 内部同时集成了不同频率的存储器件,因此设计者在设计时需要着重考虑加速器存储模块的设计,如针对迭代计算使用的中间值设计相应的缓存单元等.为了降低存储的代价,可以进行缓存的重新划分,即通过将缓存按照计算比例分为输入缓存、输出缓存和内部参数,同时引入多缓存机制来保证数据通信时间和计算时间的流水化,从而加速了整个算法的计算速度[19,21].而 PuDianNao[5]则是按照数据复用程度分为冷缓存区、热缓存区和输出缓存区.主要思路是降低高复用数据的通信代价,从而减少整个计算过程时间.

4 硬件加速器设计

围绕目前基于各种硬件的机器学习加速器设计较多,本节重点介绍 FPGA 和 ASIC 实现的相关加速器工作.基于各个加速器的特性,本文将近年来的相关文献分为四类,分别是针对特定问题的加速器、针对特定算法的加速器、针对算法的共性特征的加速器以及利用硬件模版的通用加速器框架.这四大类遵循了一个从特殊到一般的过程,并且设计难度呈递增趋势.对于前面两类问题设计加速器目前较为普遍,并且设计难度也相对较小,而对于后两类尤其是最后一类,设计难度相对较大,尚处于研究阶段,并没有得到大规模普及.

从研究的角度来看,本文的观点是体系结构设计人员应该以设计出针对机器学习算法通用的加速器体系结构为最高目标,而不仅仅局限于某个特定的应用场景或机器学习算法,以此来进一步推动该领域的发展.

4.1 针对特定问题设计加速器

针对特定问题设计加速器是目前硬件加速器应用的最广泛的领域.专门针对某一特定问题去设计加速器不仅能够很好地贴合问题的需要,并且设计难度也相对较小.针对特定问题设计加速器往往加速的是机器学习算法的推理过程而不是学习过程.

为了加速解决 Online Traffic Classification 问题,一种专用的加速器孕育而生[30].Online Traffic Classification 问题是指根据一个 TCP 连接/UDP 的建立起的传输流的若干个数据包来判断这个 TCP 连接/UDP 是由哪个应用程序发起的,该问题所使用的算法就是常见的 C4.5 决策树算法.为了完成加速目的,该团队基于问题特性设计了一种加速器结构.

整体加速器结构分两部分,一是离散化模块,二是分类模块.离散化模块对输入数据进行预处理,而分类模块则是对输入数据进行分类决策.数据的属性向量被输入至离散化模块,经过每一级离散化处理单元,数据对应的某个属性值就被离散化.然后数据被送入分类模块,经过每一级,数据就在决策树上向下走一层.一个分类单元的本地内存中保存了对应决策树中这一层的所有中间/叶子结点,下一层分类单元接收到参数(数据属性集、中间结点地址),然后找到对应中间结点继续分类.

该论文设计出的特定加速器结构还存在的一些

不足，在对于分类模块，决策树的每一层由一个 PE 负责处理，由于每层节点都不一样，势必会导致计算资源的不平衡，因此当输入数据规模较大时，该加速器件往往面临性能瓶颈. 对于计算资源的不平衡问题，可以采用负载平衡感知的剪枝方法. 以 ESE[31] 为例，通过利用该剪枝策略可以将 LSTM 模型尺寸压缩 20 倍，同时使用单独的调度器将模型编码分割到多个 PE 来完成并行计算. 以语音识别为典型应用，这种方法能够相对 CPU 和 GPU 等实现方式大幅提升性能和能效.

4.2　针对特定算法设计加速器

针对某一个机器学习算法设计的加速器是目前较为常见的应用领域. 针对特定机器学习算法设计出来的加速器在应用于某个特定问题时，往往只需要进行特定参数的配置或是一些小幅度的改动就可以较好地贴合特定的问题.

SVM 算法. SVM 算法是基于核的机器学习算法中广泛应用的算法. 目前大部分的论文主要针对 SVM 算法推理过程来设计加速器件. 在 SVM 算法的推理过程中，对于一个需要分类的数据，它需要与所有的支持向量进行相乘累加得到中间值，接着中间值会送入核函数进行处理从而得到最后的结果. 因此对于 SVM 的推理过程，我们可以针对相乘累加部分或是针对核函数执行来进行加速器的设计.

首先，对于 SVM 算法推理过程的加速，可以只对待分类向量与支持向量进行相乘累加的部分进行加速，而核函数的计算仍处于 CPU 中执行. 以文献[32]为例，在其加速器体系结构的设计中，FPGA 上集成了多个 Vector Processor Cluster，每个 Vector Processor Cluster 由多个 VPE Array 构成，每个 VPE Array 由多个 VPE 构成，每个 VPE 是一个向量处理单元，该处理单元能够处理两个向量间的点乘运算. 在加速器件执行的过程中，规模大的矩阵以数据流的形式传入，规模小的矩阵则存放在片内存储器上，每个 VPE Array 中存放的是规模小的矩阵的某一列. 此外，该加速器件还有着更细化的点乘操作，即每次向量点乘操作划分为多个 chunk 点乘，chunk 的大小需要进行合理调整从而使得 FPGA 与 CPU 间数据传输不会成为瓶颈. 但由于其设计出的加速器结构并没有同时对核函数的计算过程进行加速，所以整理的系统加速比还有待进一步提升.

另外也有团队设计了专门的计算单元来加速核函数的计算以支持加速核函数计算，并设计了新的加速器结构[33]. 以级联 SVM 加速器为例，每个 SVM 的分类模型和分类能力有较大差异. 与机器学习中的 Boost 思想类似，级联 SVM 加速器由多个弱分类器组合构成了一个强分类器. 对于某一级 SVM，如果它不能较为准确地判断出输入值的类型，则将其交给下一级功能更为强大的分类器处理. 这种方法最终呈现了一个二级分类器，其中第一级分类器能够较好地分类出离超平面较远的点. 由于其采用的核函数较简单，运行起来速度较快；而对于第二级分类器，能够分类出处于超平面边缘的点（即第一级分类器不能判断的点），采用的核函数可能较为复杂，运行起来速度相对稍慢.

SVM 算法的广泛应用使得加速 SVM 算法前景广阔. 对于 SVM 算法的推理过程来说，相关工作包括专用加速器等相对已经比较完善；而对于 SVM 算法的训练过程来说，目前加速器相对比较少. 此外，在 SVM 算法的推理过程中，数据在分类前往往需要正交化和规则化等预处理过程，该预处理过程如果在 CPU 上执行效率较低，并且占用时间比重也较高. 因此加速预处理过程的执行也是一个值得研究的 SVM 加速器改进方向.

Apriori 算法. Apriori 算法是处理关联分析的一个重要的算法. Apriori 算法主要用于发现事物之间的关联联系，它通过统计事物间相互出现的频率次数，从而获得关联度并得到各项目间关系.

通过分析 Apriori 算法特性，针对该算法前半段获取频繁项集过程的加速可以有效加速算法的运行[34]. 该工作将 Apriori 算法划分为 Candidates Generate、Candidates Pruning 与 Support Calculation 三个部分. 该加速器结构能够对于这三个阶段都能重复利用，并且也表现出优异的加速效果. 其中，Candidates Generate 部分用于生成候选的 $K+1$ 频繁项集，Candidates Pruning 部分用于对刚刚生成的候选频繁项集做预处理，去掉任意一个属性，检验剩下的 K 项集处在已经生成好的 K 频繁项集的集合中；Support Calculation 部分则用于对已经通过预校验的 $K+1$ 候选频繁项集来做统计，并计算 $K+1$ 候选频繁项集在整个数据集中出现的频率.

由于 Apriori 算法的计算原理类似一种数据统计的过程，需求数据预先应该以字典序的形式进行排列，因此对整体数据集进行字典序排序这一预处理过程也应是一个潜在的加速点.

决策树算法. 决策树算法是较为常用的机器学习树模型，分为学习和推理的两个阶段. 而决策树算法学习过程的计算核心是计算 Gini 系数或是熵增

益因子.目前对决策树算法的在学习和推理两个阶段都有着相关研究.如在 C4.6 决策树算法的学习过程加速 Gini 系数计算的加速器结构[35].每个连续属性 Gini 系数计算可以通过 FPGA 中自己定义的 Gini unit 来完成,之后通过比较单元层次连接所有的 Gini 结果,并选取出最小 Gini 系数.在这个工作的基础上,近年来对于决策树的加速器结构出现了较大进展,例如将整个决策树学习过程而并非其中的某个部分置于加速器结构中执行,从而达到减少数据间的通信延迟的目的.此外,大部分的决策树算法的输入数据通常是离散的,因此还可以针对输入数据离散化这一预处理过程进行加速.

K-Means 算法. *K*-Means 算法是一种常见的聚类算法.虽然该算法的计算核心在于求取每个点与各个质心的距离这一过程,但是将整个 *K*-Means 算法而不仅仅是计算核心固化到 FPGA 实现的加速器结构将使得加速效果进一步提升[36-37].通过将整体算法上分为了 4 个模块:Distance Kernel Block、Minimum Distance Finder Kernel Block、Accumulation Kernel Block 和 Sequential Divider Kernel Block,从而让整个 *K*-Means 算法逻辑映射到加速器上.其中 Distance Kernel Block 模块接收从片上存储器或者是片外存储器发来的数据,计算每一个点到所有类的距离.该模块 DP 单元的单元数量与类的数量对应,每个 DP 单元计算一个点到某个类的距离,则利用并行性同时计算完所有需要计算的距离数据.而 Minimum Distance Finder Kernel Block 模块接受从 Distance Kernel Block 发来的距离数据,并从中输出最小的距离对应的那个类.Accumulation Kernel Block 模块接收上一个模块产生出的最短距离对应的类,把当前数据点的特征累加到对应类的累加器中,并增加对应类计数器值,每个类都附带了一个累加器和计数器.而当所有数据点对应的类都累加后,该模块会将数据传送至下一个模块.Sequential Divider Kernel Block 模块则是由若干个除法单元以流水线化的方式组成,通过把上个模块输出的每个类数据各个特征累计的值与数据个数计数值相除,从而计算每个类在下一轮新质心的数据.通过一系列的模块化和流水线化,将整个算法进行拆分从而达到加速的效果.

贝叶斯图算法. 贝叶斯图算法主要包含了贝叶斯信念网、马尔科夫随机场等图模型,该算法利用图计算模型描述了一种变量间的相互关系.针对贝叶斯信念网模型进行求解的信念传播算法等是基于贝叶斯图的机器学习算法的常见的计算核心.由于贝叶斯图算法的特征问题,对于加速该种类型的算法可以从有向无环图的结构入手[38].本质上来说,该结构是用来解决拓扑排序问题,而并不是真正地解决贝叶斯信念网等的问题.研究人员为此设计的加速器包含多个处理单元,并且每个处理单元采用了 20 级～30 级的超深度流水线来加大吞吐率,所有处理单元通过交叉开关与多个内存模块相连.该结构依赖于对问题进行静态分析从而获得运行策略,因此避免了数据相关等问题.这样的做法虽然可以加速贝叶斯图算法,但是依然存在着缺点和不足,例如只加速了拓扑排序部分,但对算法的其它部分涉及较少.对基于贝叶斯图的机器学习算法来说,目前相关工作较少,其具体实现由于图模型的限制因素使得并行化等加速手段相对来说比较困难,仍需进一步研究.

神经网络算法. 深度学习目前是机器学习领域炙手可热的研究方向,而针对于深度学习中神经网络算法的硬件加速器也是层出不穷.其中代表性的工作是中科院计算所研发的 ASIC 硬件加速芯片 DianNao.通过将整个神经网络的计算部分分为三部,乘法单元、加法树和激活单元并映射到硬件逻辑上,从而完成对整个计算部分的硬件加速[39].对于神经网络计算流程的软件优化,可采用循环平铺[40]或循环分块[41]的方法优化其循环过程.而且也可通过分析其计算过程中各个矩阵的相互独立关系和多缓存机制优化了缓存管理[42],或者是利用模型传播路径提高数据复用率[43]或权重复用率[44],以获得性能更为优越的基于 FPGA 的神经网络硬件加速器.

除了软件的优化之外,硬件效能的提升往往也更为直观.比如新兴的金属氧化物电阻随机存取存储器(ReRAM)就带来存储方式的新变化.通过交叉阵列结构,ReRAM 可以更高效地执行矩阵向量乘法,并由此设计出了一种新的 PIM 架构,PRIME[45].并且得益于硬件的改变,通过实验可以观测到相较于之前的各种神经网络硬件加速器得到了显著的性能提升和节能效果[46].

4.3 针对算法共性设计加速器

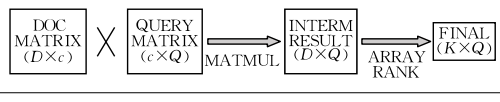
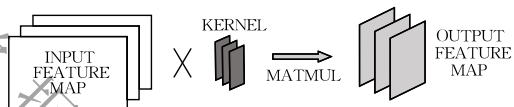
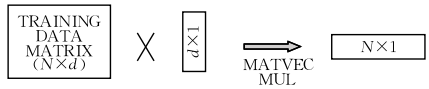
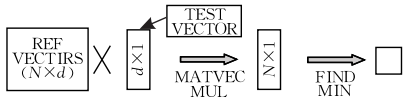
上述两种加速器的设计手段都相对专用,设计出的加速器往往只能应用与特定问题或特定算法中.为了扩大加速器的应用范文,还可以针对算法的共性特征来设计加速器,从而实现一类机器学习算法的加速执行.

通常来说，可以从两个方面来利用共性特征设计加速器，一是根据之前对机器学习算法的分类来寻找某一类机器学习算法的共性特征设计加速器；二是不局限于某类机器学习算法，而是在整个机器学习算法中寻找某些共性特征. 通过对现有文献进行总结，目前提取出的共性特征有线性代数运算、迭代计算以及简化算法访存模型这 3 种.

4.3.1　线性代数运算

大部分的机器学习算法在学习或者推理的过程中都涉及到了大量的大规模的线性代数运算，这些线性代数运算一般来说都需要占用大量的计算资源，因此往往也是算法的计算核心. 因此针对涉及到的线性代数运算进行加速能够有效地提升算法整体的性能. 其中，很多机器学习算法中间步骤都能表示为矩阵/向量乘积运算的形式，并且当中间步骤运算完成产生中间数据后，最终步骤往往是相对简单地对中间数据进行排名、寻找最大/最小值、聚合等归约操作[17]. 如表 3 所描述的 5 种算法为例，这 5 种算法均能表示为这种形式. 其中，一项代表性的工作 MAPLE 硬件加速器就是针对矩阵/向量乘积运算的过程来进行加速的[17]. MAPLE 利用矩阵/向量乘积运算的特征，处理计算中间数据并对它们进行归约操作. 对于规模大的不易变的矩阵，它往往存放在片外内存中，数据以流的形式传进给 MAPLE；对于规模小的易变的矩阵，它就被划分并存放着 MAPLE 的多个计算单元中. 每个 PE 是一个向量计算单元，能够在一个 Cycle 进行乘加运算；每个 PE 都有一个 Local Storage，存放规模小的矩阵的列；M 个 PE 构成了一条链，每个 Core 有 H 条链；对于每条链，input 从左传到右，output 从右传到左；每条链的 output 都连接着一个 Smart Memory Block，它能够对每条链的输出结果进行排名、最大/最小、聚合等归约操作，并存放符合归约条件的结果.

表 3　5 种典型机器学习算法的计算共性[17]

模型	计算描述	常用参数范围	计算关键(经过变换后)
SSI	对于每一个维度为 Q 的问询，从维度为 D 的库中找到语义结果最佳匹配的 K 个输出结果.	D：数百万 Q：32～128 K：64～128	
CNN	利用不同大小的卷积核在图片各通道做卷积运算来抽取特征.	图片尺寸：640×480 模型内参：数百个 5×5 到 10×10 的卷积核	
K-means	从给定的维度为 d 的 N 个点和常数 K 中，为每个点找到最近的分类.	N：数十万个 d：3～5 K：8～64	
SVM	将维度为 d 的 N 个训练向量重复乘以一个向量	N：1～4 百万 d：500～5000	
GLVQ	输入向量的分类等于 N 个参考向量的最近分类	N：100～1000 左右 d：100 左右	

4.3.2　迭代运算

除了线性代数运算，反复的迭代计算也是机器学习算法显著的一个共性特征. 大量的机器学习算法都需要进行反复的迭代计算，直到得到最终收敛结果，但是迭代的次数往往无从得知，因此简单地用数据流模型去设计加速器是远远不够的[47]. 此外，迭代计算还分为了同步迭代计算与异步迭代计算两种情况，同步迭代是指对一个数据的下一次迭代需要等到整体数据迭代过一轮之后才能执行；异步迭代则指对数据的下一次迭代可以立即执行，无需等待整体数据完成迭代过程.

迭代计算相较线性代数运算而言，其可加速的具体功能并不直观，因为对于不同的算法，其利用的迭代公式通常也是不同的. 但是对于迭代计算还是有一些共性特征可以进行优化加速的，比如针对迭代计算产生中间值的存储进行优化和改进或者是对迭代数据进行分配调度的方式进行改进等. 举例来说，近年来的一个代表性工作 Maestro 的加速器结构就针对异步迭代的过程进行加速[18]. 该加速器结构将所有异步迭代计算分为了两个步骤，即

Accumulate:一个节点采集与该节点有边联系的其它节点发来的消息 m,并把该消息存放在本地变量 Δv 中;Update:节点使用 Δv 和原有的权值 v 来求出新的权值 v.之后把节点权值改变量 Δv 应用一个函数得到一个值并把该值发送到相邻节点,最后把 Δv 重置为零.该系统由 1 个 CPU 充当 Master,4 块 FPGA 充当 Slave.基于 CPU 的 Master 用于进行 Slaves 的任务分发和检查停止条件等.基于 FPGA 的多个 Slaves 并行运行并通过以太网相连接.每个 Slave 除 FPGA 外还有一个 CPU/FPGA Assistant,CPU/FPGA Assistant 辅助 FPGA 从分布式文件系统中读写信息.而对于同步迭代目前常见的做法则是在加速器的计算单元中加入中间存储单元从而实现节约多次迭代的数据传输和计算时间损耗[19].

4.3.3 简化算法的访存模型

前面的两个共性特征主要是对机器学习算法计算密集型的部分进行加速,而对数据密集型的部分进行优化也能够整体提升算法的执行效率.其实无论在云计算平台还是 GPGPU,数据通信传输往往都会成为算法性能进一步提升的一个瓶颈.鉴于大量的机器学习算法的访存模型都比较类似,因此可以针对这些数据通信传输与访存的共性特征来设计加速器从而达到加速一大类的机器学习算法的目的.此种访存结构能够满足大部分机器学习算法的需求,并且能够降低开发者对加速器访存模块的开发难度[48-49].在设计时,开发者只需要在核函数的 Verilog 中声明出访存接口模块(相当于黑盒),接下来的数据读写操作都通过该模块完成;其本质是简化了开发者在设计加速器访存模块时的难度,而不是专门的对机器学习算法的数据通信传输部分进行的优化,从而获得一定的加速效果,并降低了加速器的设计难度.

另外一类加速器(如 PuDianNao)的做法是先分析各个算法的数据热度,提取出关键的计算原语并对算法循环进行展开处理来优化其算法的内存带宽需求[6].因此此类工作在加速器中使用多个单独的片上缓冲区,每个缓冲区分别存储具有类似重用距离的变量.例如 PuDianNao 使用了三个数据缓存区,分别为 hotbuf(8 kb)、coldbuf(16 kb)和 outputbuf(8 kb).其中 hotbuf 存储复用距离较短的输入数据,coldbuf 存储复用距离相对较长的输入数据.outputbuf 存储输出数据或临时结果.同时在生成程序代码时以乒乓方式利用 hotbuf 和 coldbuf,即每次加载 hotbuf 和 coldbuf 大小一半的数据,在计算这一半数据的过程中掩盖加载另一半数据的时间,从而实现节省整个计算过程的目的.

4.4 利用硬件模板设计通用加速器框架

相较之前的 3 种加速器设计方法,利用硬件模版设计加速器是一种更通用化的方式.通常情况下,这些硬件模版往往是某种编程模型的 FPGA 版本实现,在使用过程中仅需要针对特定问题设计相应的部分模块并配置好参数,当参数和模块确定下来之后,该加速器框架就能够自动运行从而加速用户要解决的问题.得益于 C to RTL 工具的发展,研究人员在设计特定模块时可以直接使用 C 语言而并非 Verilog/VHDL 语言,这极大简化了硬件的设计难度,并促进了硬件模版框架的普及利用.目前常见的三种基于硬件模版的加速器框架包括基于 Map-Reduce 模型的加速器框架、基于 LINQ 模型的加速器框架和基于图计算模型的加速器框架.而且这些加速框架通常是某一种编程模型的 FPGA 平台实现,并且能够覆盖大部分的机器学习算法.

4.4.1 基于 Map-Reduce 模型的加速器框架

Map-Reduce 模型是云计算中广泛应用的模型,在很多软件系统,如 Hadoop、Spark 等的实现中均采用的这种模型.因此,有很多研究机构都在试图将 Map-Reduce 模型应用于 FPGA 的硬件模版框架中.

云计算中的 Map-Reduce 模型需要首先完成 Map 函数和 Reduce 函数的定义,再由相应的系统完成之后的计算任务.对于基于 Map-Reduce 模型的硬件加速器框架,也都有多种实现方式.例如基于 FPGA 的 Map-Reduce 加速器框架 FPMR,在使用时也仅需要设计出相应的 Map 模块与 Reduce 模块并配置好相应的加速器参数,之后便可自动运行[26].Axel[12]提供了另外一种思路的实现.在 Axel 中,某个节点充当 Master,用于对集群的总控;对于其它的 Slave 节点,Map-Reduce 计算模型可以通过两种方法实现,一种是节点的 CPU 进行总控,GPU 负责计算 Map 过程,FPGA 负责执行 Reduce 过程,并通过 FPGA 的总线进行节点间交换;另一种则是 GPU 和 FPGA 共同来 Map 计算,CPU 负责总控和 Reduce 计算,并通过系统 I/O 来交换信息.此外,Axel 最关键的部分是运行时资源管理,其运行在系统中的每个节点的最上层,并负责处理数据分发、计算单元任务分配,节点间通信等关键任务.

4.4.2　基于图计算模型的加速器框架

Map-Reduce 模型虽然是云计算中较为常见的模型，但是在使用中也存在一些不足，如对数据关联程度较高的算法它的处理能力十分低效，难以去并行加速等. 为了应对这类问题，近些年来出现了基于图计算模型的加速器框架，图计算模型不仅能够很好的解决 Map-Reduce 模型的不足，并且也能够很好的兼容 Map-Reduce 模型，因此具有比较广泛的应用前景. 然而图计算模型在云计算中目前还并没有得到广泛普及，暂时处于试用阶段，仅有部分系统实现了图计算模型，如 Pregel、GraphLab 和 GraphX 等. 在 FPGA 平台上，实现的图计算模型加速器框架也是相对较少，并且提出的加速器框架也存在一些不足. 因此图计算模型的加速器框架的研究有着较大的理论意义与应用价值待进一步研究.

但是由于大规模的图形处理需要较高的带宽进行数据访问，而随机的数据访问、不可预测的全局数据访问、更新顶点时的冲突和不平衡的工作负载等问题给图计算加速器框架带来了严峻的挑战. 近年来有部分工作例如 GraphH[50] 等，通过引入混合存储器立方阵列的 PIM 架构来提升数据访问的速度. 此外，研究人员还可以通过基于 SRAM 的片上顶点缓存和可重构的双网连接优化了数据访问的相关问题[3]，并引入索引映射等分区和调度方法实现了工作负载平衡，也避免了冲突，同时又进一步使用一些优化方法减少了同步开销和数据复用，使其获得明显的性能提升.

在硬件加速器方面，也有部分代表性工作例如 CoRAM 访存结构[47] 通过在原有加速器工作[51] 的基础上，进行优化设计，得到新的图计算加速器框架 GraphGen[52]. 该框架在接收到图信息和设计参数后便生成出相应的 RTL 级代码. 但使用者不仅需要定义好图的结构与相应的权值信息，还需要提供出 update-function，以及构成 update-function 的一些函数操作也需要用户提供出相应的 RTL 级实现，并且用户需要指定出节点间的执行顺序. 使用者在完成这些操作后，GraphGen 的编译器负责完成一系列的子图划分，优化等操作，并最终生成出 RTL 描述的体系结构.

总的来说，基于图计算的通用加速器框架由于比较新颖，并且实现难度相对较大，目前实现的原型也存在着一些缺点，值得进行深入探索.

4.4.3　基于 LINQ 模型的加速器框架

LINQ(Language Integrated Query)是微软提出的一种类似于 SQL 的语言. 和传统 SQL 不同的是，LINQ 可以在语言集的基本操作中内嵌入用户自定义的一些函数，因此研究人员可较便捷地将现有程序改写成 LINQ 的方式.

基于 LINQ 语言也可以进行基于高层语言的硬件框架设计. 例如 LINQits[53] 便利用 SOC 设计了一个针对 C# 语言 LINQ 语言子集加速的硬件框架. 利用该框架，用户在编程时只需要把原有的程序改写成为 LINQ 的形式，然后利用 LINQits 的一系列工具既可以生成相应 RTL 级的代码，并通过运行时库来分析决定是否利用 SOC 中的可重构逻辑单元进行加速.

综上所述，由于 LINQ 语言属于专属语言，其开放程度较低，并且目前也仅仅是在微软的产品线上得到了普及. 因此，针对 LINQ 涉及加速器框架的相关研究存在普适性及适用性等问题.

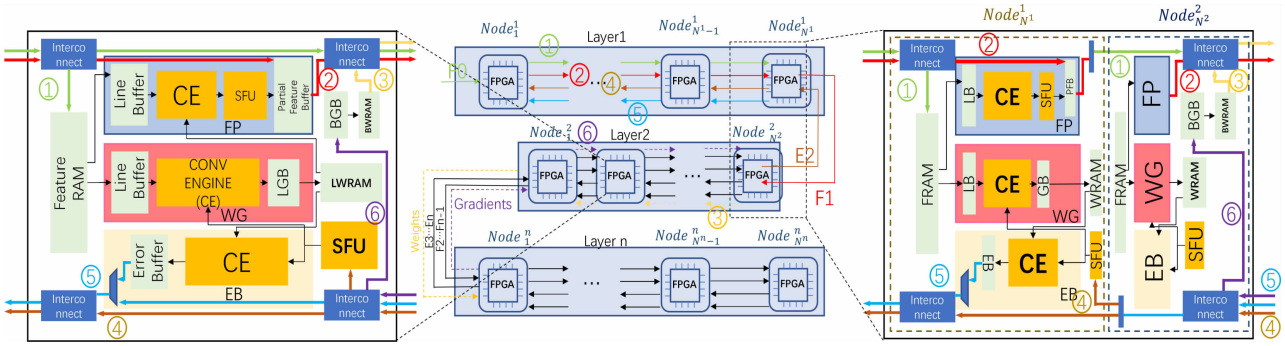
4.5　硬件框架设计模式

由于目前对于硬件加速方向的研究和需求越来越旺盛，各个研究团队和企业也开始了硬件加速器的平台建设，而目前来说的硬件加速平台的框架设计模式主要分为了 Stream 方式和 Single Engine 方式[54]. 而目前由于深度学习是机器学习的热门方向，国内外诸多研究机构也对于深度学习算法提出了加速需求，本文接下来就以深度学习算法为例介绍这两种方法和目前的研究情况.

4.5.1　Stream 方式

Stream 方式通常由用于目标神经网络算法每个层的不同硬件模块组成，其中每个块被单独优化以利用其层的并行性，并将所有异构块都被链接以形成管道. 当数据通过体系结构流传输时，数据通过神经网络的不同部分进行. 因此，这种设计方法通过流水线技术利用层之间的并行性，并使它们能够并发执行. 然而，由于必须为每个不同模型生成新的比特流，所以会导致较长的编译时间. 目前采用 Stream 方式的代表性工作包括 FPDeep[55]、AutoCodeGen[56] 和 Finn[57] 等.

FPDeep. FPDeep 是采用 Stream 方式来进行 CNN 单层和跨层的流水线结构映射[55]. 如图 2 所示，其为每一层分配一个单独的处理单元，可直接在 FPGA 集群上应用和部署. FPDeep 利用层间映射和层内划分以及权重负载平衡策略，能够合理分配各个 FPGA 的任务使得算法的实现深度流水化，如表 4 所示相较于单 FPGA 的实现，该方法的计算性能获得了约 5 倍的提升.



(a) 映射 CNN 单层部分的 FPGA 结构 (b) FPDeep 流水线结构 (c) 映射 CNN 跨层部分的 FPGA 结构

图 2 FPDeep 加速器系统框架概览[55]

表 4 集群式加速器 FPDeep 的实验结果[55]

Device	CPU	GPU	GPU		FPGA		FPDeep		
	AMD A10	Titan X	Tesla K80		XC7VX690T				
CNN Model	AlexNet	AlexNet	AlexNet	VGG-16	AlexNet	VGG-16	AlexNet	VGG-16	VGG-19
Configuration	1 CPU	1 GPU	1 GPU	1 GPU	4 FPGAs	1 FPGA	15 FPGAs	15 FPGAs	15 FPGAs
Precision	Floating	Floating	Floating	Floating	Fixed 16	Fixed 16	Fixed 16	Fixed 16	Fixed 16
Performance (GOPS)	34.23	1385	2330	2018	207(Per FPGA)	290(Per FPGA)	1157(Per FPGA)	1197(Per FPGA)	1220(Per FPGA)
Power efficiency (GOPS/J)	0.39	4.22	7.87	6.86	6.55	8.28	37.09	37.88	38.13

AutoCodeGen. AutoCodeGen 包括层级的参数化硬件块，支持 CONV，POOL，NORM 和 FC 层[56]. CONV 块由卷积单元组成，它们以完全展开的方式执行点积运算. 实例化的卷积单元进一步包含在可调数量的 Group 中，输入特征映射在所有 Group 之间共享. 每个 Group 使用不同的权重集处理输入特征图，以计算独立的输出特征图. FC 层映射到名为 FC cores 的计算单元，可以可调节地利用输入神经元并行性并且可以进行时间复用. 类似地，POOL 块利用输出特征映射的并行性到可调度. NORM 层映射到固定硬件块，该块采用分段线性逼近方案进行指数运算，单精度浮点算法则最小化精度损失. 与生成流体系结构的其余工具流的数据驱动控制机制相反，AutoCodeGen 以分布式方式执行每个硬件块的调度和控制，其中由专用的本地 FSM 协调每个块的操作.

Finn. Finn 是一个基于 BNN 的结构生成自定义流式架构[57-58]. 给定目标 BNN，每个层被映射到专用计算引擎，并且所有引擎以流水线方式连接. 通过这种设计，每个计算引擎可以配置为满足相关层的要求并匹配相邻引擎的处理速率. 而这种实现方式使得整个架构适合于特定网络. 该团队将加速重点放在 BNN 特性上，其计算引擎不同于传统的 CNN 硬件设计，而且针对二值化层的有效映射进行了优化，包括用于二值化卷积，最大池化和批量归一化的专用硬件[59]. Finn 将二进制卷积表示为矩阵向量运算，然后进行阈值处理. 为此，该体系结构的矩阵矢量阈值单元(MVTU)，经过优化后可以执行大多数核心二值化操作. 同时该框架所有二进制化权重都需要存储在片上存储，而外部存储器的传输仅限于网络模型的输入和输出数据，以此解决了硬性资源限制带来的问题.

4.5.2 Single Engine 方式

而另一种硬件架构模式是 Single Engine 方式，它则更有利于灵活性而不是定制. 其计算引擎通常以处理元件的脉动阵列或矩阵乘法单元的形式，顺序地执行计算层，而硬件控制和操作调度则是由软件执行. 因此可以根据输入模型和可用的 FPGA 资源进行扩展. 通过将此方法发挥到极致，可以仅基于目标 FPGA 的资源来配置和扩展架构，而无需针对特定的网络模型，因此，在单次编译之后，相同的比特流可以针对许多模型而无需进行比特流的重配置. 尽管灵活性增加，但由于类似于处理器的控制机制，降低了效率[60]. 此外，一刀切的方法可能导致具有不同工作负载特性的网络模型上实现最终性能并不一致. 此类代表性的工作包括 ALAMO[61]、FP-DNN[62] 和 FFTCodeGen[63] 等.

ALAMO. 该团队设计的硬件体系结构包括

了用于 POOL、ReLU 和 NORM 层的硬件块，以及在 CONV 和 FC 层之间共享的 2D 计算单元阵列[61,64-67]. 在 CONV 层中，阵列利用一个输入要素图中的并行度和多个输出要素图. 在每个时刻，阵列的每一行负责一个输出特征图，其列处理相同输入特征图的不同窗口并协同地组合它们的部分结果. 通过将 FC 层转换为 1×1 CONV 层，将 FC 层映射到同一硬件块上. 此外，ALAMO 包括批量标准化块和逐元素加法器. 这些组件用作主要块的补充，元素加法器用于实现具有不规则数据流的模型，包括残余网络[68]. ALAMO 的编译器会考虑目标 CNN 中存在的层，并仅实例化必要的硬件块. 在生成体系结构之后，以顺序方式调度层. 该方法减轻了在相同类型的不同层之间分配资源的问题，并且简化了设计空间和网络层间的调度. 而加速器的控制则在编译时被静态地确定，并在执行网络的不同部分时进行顺序加载.

FP-DNN. FP-FNN 通过将 CONV 和 FC 层以及 RNN 中的循环连接和 LSTM 中的各种门单元转换为矩阵乘法[62]，FP-DNN 生成以单个通用矩阵乘法(MM)引擎为核心的架构. 为了平衡计算资源和外部存储器带宽，在输入矩阵上应用平铺，并以流水线方式处理平铺. MM 引擎通过点积单元以矢量为基础处理各个 tiles. 点积单元由一系列乘法器组成，它们完全展开点积过程的所有乘法，再连接到加法树上. 为了维持计算资源的高利用率并隐藏片外存储器的延迟，FP-DNN 采用双缓冲来传输矩阵. MM 引擎在层之间分时，在将中间结果写回到片外存储器之前，由单独的硬件应用非线性和池化操作. 片上存储器被组织为一个缓冲池，可以在运行时由不同的数据重用，从而达到较高利用率，而每个缓冲区的分配时间表作为设计空间探索的一部分进行处理. 最后，利用 OpenCL 的模块实现各控制逻辑以及与外部存储器和主机 CPU 的接口.

FFTCodeGen. FFTCodeGen 主要与一般的硬件框架有两个区别[63,69-70]. 首先，FFTCodeGen 针对异构 Intel HARP 平台进行了优化，框架在 CPU 和 FPGA 之间分割 CNN 工作负载，由 FPGA 执行 CNOV 层的计算，而 CPU 来计算剩余的模型层. 其次，FFTCodeGen 通过基于 FFT 的算法在频域中执行卷积. 利用这种方法，空间域中的卷积运算被映射到频域中的 Hadamard 逐元素乘积，从而有效地降低了计算复杂度[71].

4.5.3　各硬件框架的对比

计算性能是评价硬件加速器框架的核心指标. 本文将现有部分代表性工作的相关文献中抽取了性能指标进行对比，如表 5、表 6 所示. 对于 FINN 加速器，由于该加速器的设计是针对二值神经网络进行加速优化，与其它的设计有较为明显的差异性，故在此省去.

表 5　各个加速器的硬件资源利用情况

加速器		硬件使用情况		
		DSP	Logic*	BRAM**
FPDeep		— (~80% AVG)	—	— (~90% MAX)
AutoCodeGen		1436(39.9%)	115K(26.6%)	585(39.7%)
ALAMO		256(100%)	112K(48%)	2330(91%)
FPDNN	Float 16	1036(65%)	42K(25%)	919(46%)
	Float 32	264(17%)	164K(67%)	1343(95%)
FFTCodeGen		256(100%)	107K(46%)	1377(73%)

注：* 表示 Xilinx FPGA 基于 LUT，Altera FPGA 基于 ALM；
** 表示 Xilinx FPGA 基于 BRAM (36 Kb)，Altera FPGA 基于 M20K RAM (20 Kb).

表 6　各加速器间性能对比

加速器		硬件型号	计算模型	频率/MHz	内存类型	性能	
						GOP/s	GOP/J
FPDeep		Virtex7 VX690T	AlexNet	—	Off-chip	1220*	38.13*
AutoCodeGen		Virtex7 VX690T	AlexNet	100	—	222.1	8.96
ALAMO		Stratix V GXA7	AlexNet	100	4 GB DDR3	114.5	—
FPDNN	Float16	Stratix V GSMD5	VGG19	150	4 GB DDR3	364.36	14.57
	Float32					81	3.24
FFTCodeGen		Stratix V GXA7	AlexNet	200	On-chip	780.6	—

注：* 代表的是对于 FPGA 集群来说平均每片的结果.

从表 6 可以看出，相较于片外存储器，片上内存带来的高通信速度在最后的计算性能上有着比较明显的提升. 从 ALAMO 和 FFTCodeGen 这两个工作的性能对比可以得出，仅仅只是内存的方式发生了改变，在 FFTCodeGen 的频率较提升 ALAMO 提升一倍的情况下，其实验性能却变为了 7 倍之多. 而在计算精度这方面，可以看到 FPDNN 在他们的实验中做出了对比，在仅仅将计算精度由 16 位提升到 32 位，也即两倍的情况下，经过实验得到的计算性能就由原来的 364.36 下降到了 81，也就是

16 位的计算性能基本等于 32 位的 4.5 倍. 与此同时,硬件框架的设计模式其实也对最后的结果会有一些影响. 由于 Stream 模式相较于 Single Engine 模式来说,最大的不同在于模型的映射,Stream 更倾向于将整个网络进行映射,故而联合表 5 和表 6 中对比 AutoCodeGen 和 ALAMO 两项,在硬件型号不同的情况下,AutoCodeGen 基本上用掉了相当于 ALAMO 6～7 倍的 DSP,而 ALAMO 则由于 Single Engine 的特性,有着更多的数据需要缓存,因此近乎用掉了相当于 AutoGenCode 4 倍多的 BRAM. 与此同时可以看到对于计算精度这方面,在不影响计算结果的情况下的降低精度的必然性. 对比 FPDNN 的两个实验的硬件利用,降低精度不仅大幅提升计算核心的尺寸,即 DSP 利用率,同时也降低对数据通信的需求.

综上所述,其实这两种框架设计模式各有利弊,在使用时可以依据硬件的条件进行选择. 但是其实对于商业用户来说其实可以依据 FPDeep 的方式构建 FPGA 集群,此时虽然使用了 Stream 方式进行了设计,但对于单片 FPGA 来说更类似与 Single Engine,而且在最终的实验结果上可以看出,使用 FPGA 集群的 FPDeep 不仅拥有更好的性能,而且整体的资源利用率也得到了进一步的提升.

5 硬件加速器的机遇与挑战

5.1 FPGA 的优势与不足

早在 20 世纪 60 年代,Gerald Estrin 等人就提出了可重构计算的概念,而直到 1985 年,Xilinx 才推出了第一款 FPGA 芯片,距今已经有三十多年的时间. 早期虽然 FPGA 平台的并行化程度高,并有可重构特性,但是由于其重新配置成本和高编程复杂性,FPGA 相关研究并未受到充分重视. 近年来,随着人工智能及其应用的不断发展,高层综合工具等 EDA 软件和高效编程库逐渐丰富,FPGA 上能够集成的计算资源也越来越多,使得业界更多的研究人员正在投入到基于 FPGA 的高性能计算加速器的研究,成为目前机器学习领域加速器的一个研究热点. 目前 FPGA 平台作为加速器的主要优势是:

(1) 高能效. 由于直接可以对逻辑功能进行硬件编程和快速优化,FPGA 可以在快速定制化的同时实现高性能低功耗. 高能效给领域专用的体系结构设计带来了巨大优势,一方面在终端或者边缘计算领域等功耗受限的场景提供硬件支持,另外在数据中心或云服务中心这类高耗能设备中也可以大幅降低电力能源消耗,并同时降低服务器的散热压力. 从多项研究的实验结果中可以看出,利用 FPGA 加速各类算法可以达到百倍于同算力 CPU 平台的能效提升,与同算力的 GPU 平台相比可以获得十倍以上的能效提升[65].

(2) 高并行化. 高并行化是选择 FPGA 平台加速深度学习的主要特性. 由于 FPGA 集成了丰富的逻辑硬件单元,可以使用并行化算法实现硬件逻辑结构的快速优化,利用任务级并行或者数据级并行等策略可以有效提升大多数的机器学习算法的并行性[25,27].

(3) 灵活性. FPGA 的可重构特性是其区别于 ASIC 硬件的重要天然优势之一. 由于 FPGA 的可重构性,它可以实现面向特定领域的快速定制. 例如,在硬件设计和应用程序设计完成后,如性能、功耗、面积等指标未达到理想状态,则可针对 FPGA 进行快速软硬件迭代优化,重新配置使基于 FPGA 的硬件加速器能够不断演进,满足变化的需求.

(4) 安全性. 在当前的人工智能时代,数据的生成速度越来越快,因此数据安全性变得尤为重要. 目前,关于数据和计算机的安全防护的通常在软件层面,无法消除底层因为安全或者攻击带来的隐患. 采用 FPGA 等可定制芯片进行计算机,特别是处理器的安全加固与漏洞保护,则可以从硬件架构级别更好地增强安全性.

虽然对于目前利用 FPGA 去加速机器学习算法有众多优势,但依然有一些关键问题有待解决.

(1) 重构开销大. FPGA 平台的可重构性是一把双刃剑. 虽然它在计算加速方面带来了性能功耗等诸多优势,但 FPGA 的重新配置的时间也是不容忽视的. 一般来说 FPGA 的重构过程分为静态重构和动态重构两种模式. 静态重构,即编译时配置,是在任务运行之前实现 FPGA 硬件加速器的加载与配置并在任务运行的过程中固定硬件逻辑形式. 在这种模式下,FPGA 大多作为面向特定应用的加速器原型系统和验证平台. 与此相比,动态重构也被称为运行时重新配置,其通常使用上下文配置模式动态重新加载硬件. 在任务执行过程中,FPGA 中的硬件模块会根据需要重新配置部分逻辑,在此过程中可以维持目前程序的运行,但需要考虑动态部分重

构的代价和开销[72]. 此种模式的优势在于可以在运行时根据应用的状态不断调整硬件架构，实现硬件逻辑的自演进. 目前对 FPGA 动态重构方法的研究通过将 FPGA 中的可重构模块进行划分，并将一部分模块运行计算，另一部分运行重构过程来实现流水化处理从而完成在单 FPGA 平台对神经网络全模型的映射[73-75]. 但从实验数据来看，重构开销相对于计算时间来说依然非常可观，例如文献[75]中指出对于神经网络卷积层的重构时间为 155 ms，而该层的执行计算耗时仅有 2.7 ms，因此，如何充分利用部分重构策略，实现计算和重构的流水化，以掩盖 FPGA 的动态重构开销还需要进一步探索.

(2) 编程门槛高. 虽然可重构计算体系结构的概念由来已久，而且已经有了更为成熟的工作，但可重构计算在以前并没有得到普及. 原因主要有两个：一个是从可重构计算出现到 21 世纪初的 40 年时间是摩尔定律的黄金时代，在此期间，技术每年都会更新. 因此，架构更新带来的性能改进并不像技术更新那样直接和有力；另一个是因为没有成熟框架，传统的 CPU 编程采用高级抽象编程语言. 然而，可重构计算需要硬件编程，通常使用硬件编程语言(Verilog，VHDL)，这将花费程序员很多时间掌握. 近年来，随着 EDA 软件的不断完善，高层综合工具的不断普及，以及基于 FPGA 的高层编程框架不断涌现，FPGA 的编程墙问题得到了一定的缓解. 然而目前 EDA 软件中的高层综合功能，如要生成效率较高的硬件代码，仍然需要程序要具备深厚的体系结构专业知识才能胜任，因此如何提高软件程序员在 FPGA 等平台下的编程效率，仍需要进一步探索.

5.2　展望及进一步研究方向

随着目前大数据、云计算、物联网等领域的蓬勃发展，数据规模飞速增长，机器学习算法需要处理的数据量大幅提高. 一方面，大规模的数据作为输入，使得算法不断优化，其精度和效率随之提升，同时，算法需要实际处理的应用问题规模也越来越大，因此依赖 GPGPU、FPGA 等高性能计算的硬件平台和设备. 在算法设计时，传统的机器算法模型和框架中的高性能代码大多由程序员手动完成，导致了算法和模型结构日新月异，新的硬件平台也是层出不穷，从而带来各种开发研究效率相关的问题. 对于算法模型的统一和优化问题，目前可以通过编译器技术和设计空间搜索等方法来优化，然而对于如何克服硬件平台中 FPGA 的诸多困难则需要研究人员再进一步探讨. 总的来说，本文认为进一步的主要研究方向涵盖以下几个方面：

(1) 内存优化. 从目前的研究来看，由于机器学习算法的大量应用有着数据密集的特点，其存储部件的能量消耗甚至能达到总能量的 50%～80%[41]. 同时相较于片外存储器，片上内存带来的带宽提升在最终的计算性能上有着比较明显的提升. 在前述的实验结果中，表 6 可以看到 ALAMO 和 FFTCodeGen 这两种设计框架的性能对比，FFTCodeGen 针对 ALAMO 进行了存储的优化，在频率提升 1 倍的情况下，性能提升了 7 倍. 因此对于目前的研究方向中，设计高效的机器学习计算框架并减少存储的能量消耗是目前的关键问题之一. 对于内存优化，目前常用的技术手段有乒乓机制、数据压缩、刷新率控制、优化内存单元及访存模式、设计细粒度内存电源管理策略和使用非易失性内存等[19,21]. 在 FPGA 这种资源受限的平台上，特别可以通过分析其计算过程来进行缓存管理策略的针对性优化，从而达到增加数据复用率，提升性能的目的[41]. 与此同时，还可以按照数据复用程度将缓存进行区域划分，通过降低高复用数据的频繁替换来减少时间[5]，也可以利用硬件流水的顺序分割数据的读写路径，节约吞吐需求[73].

(2) 数据优化. 由于 FPGA 的可定制特性，在算法实现时可以控制其数据位宽来进行精度和性能的权衡. 使用较低带宽的数据单元可以很明显得到平台计算性能的提升[76]，比如在表 6 中可以看到 FPDNN 的实验中做出了对比，在仅仅将计算精度由 32 位降低到 16 位，也即两倍的情况下，实验得到的计算性能由 81 增长到 364.36，也就是 16 位精度数据的实验计算性能基本约等于 32 位的 4.5 倍. 但是由此带来的计算结果误差也不容忽视，因此为不同的机器学习算法设计对应的硬件加速器，需要有针对性的选取数据位宽，从而把握两者之间的权衡，而 BitFusion[29] 则是在数据输入时确定数据位宽动态调整计算核心的尺寸，从而在不影响计算精度的情况下获得更高的并行度. 与此同时，也有通过聚类算法对神经网络模型的相似权重进行聚类，可以大幅减少权重数据量，从而仅利用片上缓存便可以计算整个模型，而将准确率的损失维持在可控范围之内[74]. 并且不仅可以对计算数据进行优化，也可以对模型本身进行压缩，比如 4.1 节提到的 ESE 就利用了负载平衡感知剪枝方法[31]将计算模型尺寸压

缩 20 倍的同时使用单独的调度器将模型编码并分割到多个 PE 来完成加速计算.

(3)频率优化. 目前大部分研究中基于 FPGA 平台的加速器工作频率大多在 100 MHz～300 MHz 范围内,与基于专用器件的加速器能够达到的频率还有较大差异,其主要原因是其频率受到了片上 SRAM 和 DSP 等逻辑器件单元之间的路由限制. 如何解决或避免这类问题进而提升基于 FPGA 计算平台的频率也是 FPGA 相关研究人员目前关注的问题.

(4)FPGA 集群. 如果可以合理处理调度和分配问题,利用多个 FPGA 集成应该能达到更加优秀的效果[55]. 目前此方向的研究仍不多见,有进一步的探索空间. 从设计模式上来看,在设计基于 FPGA 的加速器平台时,选取 Stream 方式和 Single Engine 方式[54]来进行加速器的部署方式可以依据已有的硬件条件来进行. 两种模式的不同点主要体现在模型的映射方式,其中 Stream 更倾向于将整个网络进行映射,故而可以看到 Stream 架构的加速器往往会用到更多的 DSP 资源,而 Single Engine 架构的加速器则考虑更多的本地缓存数据,因此其使用的片上存储 BRAM 资源较大. 与此同时,对于计算精度来说,在不影响计算结果的情况下降低精度不仅可以大幅提升计算核心的尺寸,即 DSP 的利用率,同时也减少了加速器对数据通信的需求. 由于 FPGA 集群的硬件资源相对比较丰富,因此 Stream 方式在针对每一层神经网络进行优化,其带来的计算性能优势就会比较明显. 在最终的实验结果上可以看出,使用 Stream 方式的 FPDeep 加速器,从性能和资源利用率等方面都优于基于 Single Engine 方式的加速器[55].

(5)加速器兼容性. 由于目前机器学习算法众多,不同算法间存在较大差异,在加速器的硬件设计中不可避免的需要进行针对化定制. 为了考虑不同算法的兼容性,PuDianNao 的思路值得加速器设计人员进一步借鉴,即通过分析不同算法的计算流程,将算法分为多个普适阶段,也即算法核心算子,在使用某算法时跳过不必要的阶段来大大增强加速器兼容性和并行化能力[5],从而构建支持更多算法的通用加速器. 从 Amdahl 定律来说,算法的计算核心一般是算法最耗时的计算过程,因此加速 Kernel 能够显著缩短整个算法的执行过程. 具备了面向 Kernel 的专用加速器设计,设计人员就可以利用 FPGA 对多个算法的 Kernel 固化到多个计算单元上来兼容不同加速目标的执行.

(6)异构计算平台的研究. 异构计算的概念涵盖较广,如 ASIC、FPGA、GPU、DSP、NPU 等,目前仍面临一系列的挑战性问题,如何进行更好地设计空间探索[40,77-78],以及怎样较好地配合利用 GPU、FPGA 以及 NPU 等异构计算资源也是提升目前异构计算平台效能的关键问题之一.

6 总 结

随着机器学习应用的广泛普及,针对机器学习算法加速器的研究已经成为计算机体系结构领域研究的一个热点. 目前已经有专门针对各种机器学习算法的加速器[5],如何将机器学习加速器进行针对性的硬件优化、软件适配、及应用落地是围绕该领域展开的研究重点.

从目前的计算机硬件发展趋势来看,我们可以预见到面向机器学习等专用领域的体系结构会快速蓬勃发展. 未来硬件加速器的发展方向也逐渐清晰:首先,面向领域的专用硬件平台会提供越来越高效且易用的编程接口. 第二,将加速器放置在内存附近,实现存内计算,以进一步减轻访存和数据重用的压力. 第三,硬件加速器目前还主要聚焦于如何降低机器学习算法中推断过程的延迟,需逐步开展面向训练过程的优化. 第四,由于 FPGA 的功耗和计算性能的优异表现,在物联网设备和智能边缘计算设备中会得到广泛采用. 第五,在新兴的如自动驾驶汽车领域中,FPGA 和 NPU 等专门加速器也在开始逐渐应用. 在该领域中,利用这些硬件加速器处理由高级驾驶员辅助系统产生的大量时间序列信息来增加自动驾驶的性能. 第六,作为人工智能革命的重要部分,以 Qualcomm 855、Kirin 980 和 A12 Bionic 等芯片为代表的集成解决方案也会逐渐大规模应用,在广泛推广之后可以实现人人都是计算节点的未来愿景.

近年来类似对机器学习加速器的综述还有文献[76,79-80]等. 其中文献[79-80]两文主要针对神经网络加速器的进展进行了汇总,而本文则更加深入地介绍了机器学习领域的硬件加速器发展情况;文献[76]主要针对基于 FPGA 的深度学习加速器做出了总结,而本文对机器学习以及 ASIC 加速器做了更多的介绍与分析,并且进一步对比了目前硬件

设计框架的研究.至此,本文的主要工作和内容是对调研的可定制硬件平台实现的机器学习加速器进行了汇总和分析,然后简单介绍了各优化方法的设计和实现的硬件加速器并对未来的该领域发展方向做出展望.

致　谢　本课题得到国家重点研发计划(2017YFA-0700900,2017YFA0700903)、国家自然科学基金(61379040)、江苏省自然科学基金(BK20181193)、中国科学院青年创新促进会(2017497)以及中央高校基本科研基金(WK215011003)提供相关的支持与帮助.感谢所有审议人员的反馈和建议!

参　考　文　献

[1] Bekkerman R, Bilenko M, Langford J. Scaling Up Machine Learning: Parallel and Distributed Approaches. UK: Cambridge University Press, 2011

[2] Gonzalez J E, Low Y, Gu H, et al. PowerGraph: Distributed graph-parallel computation on natural graphs//Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12). Hollywood, USA, 2012: 17-30

[3] Li C, Xue Y, Wang J, et al. Edge-oriented computing paradigms: A survey on architecture design and system management. ACM Computing Surveys, 2018, 51(2): 1-34

[4] Chen T, Moreau T, Jiang Z, et al. TVM: An automated end-to-end optimizing compiler for deep learning//Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). Carlsbad, USA, 2018: 578-594

[5] Liu D, Chen T, Liu S, et al. PuDianNao: A polyvalent machine learning accelerator. ACM SIGARCH Computer Architecture News, 2015, 43(1): 369-381

[6] Jouppi N P, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit//Proceedings of the 44th Annual International Symposium on Computer Architecture. Toronto, Canada, 2017: 1-12

[7] Zhu Hu-Ming, Li Pei, Jiao Li-Cheng, et al. The overview of the parallelization in deep neural network. Chinese Journal of Computers, 2018, 41(8): 171-191(in Chinese)
(朱虎明,李佩,焦李成等.深度神经网络并行化研究综述.计算机学报,2018,41(8):171-191)

[8] Chu C, Kim S K, Lin Y A, et al. Map-reduce for machine learning on multicore. Advances in Neural Information Processing Systems, 2007, 19: 281

[9] Choudhary A N, Honbo D, Kumar P, et al. Accelerating data mining workloads: Current approaches and future challenges in system architecture design. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2011, 1(1): 41-54

[10] Huang Shan, Wang Bo-Tao, Wang Guo-Ren, et al. The survey of the optimization technique for MapReduce. Journal of Frontiers of Computer Science & Technology, 2013, 7(10): 885-905(in Chinese)
(黄山,王波涛,王国仁等. MapReduce 优化技术综述. 计算机科学与探索,2013,7(10):885-905)

[11] Liu Ying, Lv Fang, Wang Lei, et al. The research and development of heterogeneous parallel programming model. Journal of Software, 2014, 25(7): 1459-1475(in Chinese)
(刘颖,吕方,王蕾等.异构并行编程模型研究与进展.软件学报,2014,25(7):1459-1475)

[12] Tsoi K H, Luk W. Axel: A heterogeneous cluster with FPGAs and GPUs//Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey, USA, 2010: 115-124

[13] Sujeeth A, Lee H J, Brown K, et al. OptiML: An implicitly parallel domain-specific language for machine learning//Proceedings of the 28th International Conference on Machine Learning (ICML-11). Washington, USA, 2011: 609-616

[14] Auerbach J, Bacon D F, Burcea I, et al. A compiler and runtime for heterogeneous computing//Proceedings of the 49th Annual Design Automation Conference. San Francisco, USA, 2012: 271-276

[15] Long G, Yang J, Zhu K, et al. Fusionstitching: Deep fusion and code generation for tensorflow computations on GPUs. arXiv preprint arXiv:1811.05213, 2018

[16] Srivastava P, et al. PROMISE: An end-to-end design of a programmable mixed-signal accelerator for machine-learning algorithms//Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). Los Angeles, USA, 2018: 43-56

[17] Cadambi S, Majumdar A, Becchi M, et al. A programmable parallel accelerator for learning and classification//Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques. Vienna, Austria, 2010: 273-284

[18] Unnikrishnan D, Virupaksha S G, Krishnan L, et al. Accelerating iterative algorithms with asynchronous accumulative updates on FPGAS//Proceedings of the 2013 International Conference on Field-Programmable Technology (FPT). Kyoto, Japan, 2013: 66-73

[19] Guan Y, Yuan Z, Sun G, et al. FPGA-based accelerator for long short-term memory recurrent neural networks//Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC). Chiba, Japan, 2017: 629-634

[20] Li P, et al. E-RNN: Design optimization for efficient recurrent neural networks in FPGAs//Proceedings of the 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). Washington, USA, 2019: 69-80

[21] Liu Qinrang, Liu Chongyang. Calculation optimization for convolutional neural networks and FPGA-based accelerator design using the parameters sparsity. Journal of Electronics & Information Technology, 2018, 40(6): 1368-1374

[22] Lu L, Liang Y, Xiao Q, et al. Evaluating fast algorithms for convolutional neural networks on FPGAs//Proceedings of the 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). Napa, USA, 2017: 101-108

[23] Zhang C, Wu D, Sun J, et al. Energy-efficient CNN implementation on a deeply pipelined FPGA cluster//Proceedings of the 2016 International Symposium on Low Power Electronics and Design. San Francisco, USA, 2016: 326-331

[24] Low Y, Gonzalez J E, Kyrola A, et al. GraphLab: A new framework for parallel machine learning. arXiv preprint arXiv: 1408.2041, 2014

[25] Blelloch G E. Programming parallel algorithms. Communications of the ACM, 1996, 39(3): 85-97

[26] Shan Y, Wang B, Yan J, et al. FPMR: MapReduce framework on FPGA//Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey, USA, 2010: 93-102

[27] Wang C, Gong L, Yu Q, et al. DLAU: A scalable deep learning accelerator unit on FPGA. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, 36(3): 513-517

[28] Song Linghao, Mao Jiachen, Zhuo Youwei, et al. HyPar: Towards hybrid parallelism for deep learning accelerator array//Proceedings of the 25th IEEE International Symposium on High Performance Computer Architecture. Washington, USA, 2019: 56-68

[29] Sharma H, Park J, Suda N, et al. Bit Fusion: Bit-level dynamically composable architecture for accelerating deep neural network//Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). Los Angeles, USA, 2018: 764-775

[30] Tong D, Sun L, Matam K, et al. High throughput and programmable online traffic classifier on FPGA//Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey, USA, 2013: 255-264

[31] Han S, Kang J, Mao H, et al. ESE: Efficient speech recognition engine with sparse LSTM on FPGA//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2017: 75-84

[32] Cadambi S, Durdanovic I, Jakkula V, et al. A massively parallel FPGA-based coprocessor for support vector machines //Proceedings of the 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines. Napa, USA, 2009: 115-122

[33] Papadonikolakis M, Bouganis C S. Novel cascade FPGA accelerator for support vector machines classification. IEEE Transactions on Neural Networks and Learning Systems, 2012, 23(7): 1040-1052

[34] Baker Z K, Prasanna V K. Efficient hardware data mining with the Apriori algorithm on FPGAs//Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05). Napa, USA, 2005: 3-12

[35] Memik G, Choudhary A. An FPGA implementation of decision tree classification//Proceedings of the 2007 Design, Automation & Test in Europe Conference & Exhibition. Nice, France, 2007: 1-6

[36] Hussain H M, Benkrid K, Erdogan A T, et al. Highly parameterized k-means clustering on FPGAs: Comparative results with GPPs and GPUs//Proceedings of the International Conference on Reconfigurable Computing and FPGAs. Cancun, Mexico, 2011: 475-480

[37] Hussain H M, Benkrid K, Ebrahim A, et al. Novel dynamic partial reconfiguration implementation of k-means clustering on FPGAs: Comparative results with GPPs and GPUs. International Journal of Reconfigurable Computing, 2012, 2012: 1

[38] Lin M, Lebedev I, Wawrzynek J. High-throughput Bayesian computing machine with reconfigurable hardware//Proceedings of the 18th annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey, USA, 2010: 73-82

[39] Chen T, Du Z, Sun N, et al. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. ACM SIGPLAN Notices, 2014, 49(4): 269-284

[40] Suda N, Chandra V, Dasika G, et al. Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks//Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'16). New York, USA, 2016: 16-25

[41] Zhang C, Li P, Sun G, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks//Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2015: 161-170

[42] Qiu J, Wang J, Yao S, et al. Going deeper with embedded FPGA platform for convolutional neural network//Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2016: 26-35

[43] Azizimazreah A, Chen L. Shortcut mining: Exploiting cross-layer shortcut reuse in DCNN accelerators//Proceedings of the 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). Washington, USA, 2019: 94-105

[44] Hegde K, Yu J, Agrawal R, et al. UCNN: Exploiting computational reuse in deep neural networks via weight repetition//Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). Los Angeles, USA, 2018: 674-687

[45] Chi P, Li S, Xu C, et al. PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. ACM SIGARCH Computer Architecture News, 2016, 44(3): 27-39

[46] Song L, Qian X, Li H, et al. PipeLayer: A pipelined ReRAN-based accelerator for deep learning//Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA). Austin, USA, 2017: 1552

[47] Han L, Liew C S, Van Hemert J, et al. A generic parallel processing model for facilitating data mining and integration. Parallel Computing, 2011, 37(3): 157-171

[48] Chung E S, Hoe J C, Mai K. CoRAM: An in-fabric memory architecture for FPGA-based computing//Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey, USA, 2011: 97-106

[49] Chung E S, Papamichael M K, Weisz G, et al. Prototype and evaluation of the CoRAM memory architecture for FPGA-based computing//Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey, USA, 2012: 139-142

[50] Dai G, Huang T, Chi Y, et al. GraphH: A processing-in-memory architecture for large-scale graph processing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 38(4): 640-653

[51] Betkaoui B, Thomas D B, Luk W, et al. A framework for FPGA acceleration of large graph problems: Graphlet counting case study//Proceedings of the 2011 International Conference on Field-Programmable Technology. New Delhi, India, 2011: 1-8

[52] Weisz G, Nurvitadhi E, Hoe J. GraphGen for CoRAM: Graph computation on FPGAs//Proceedings of the Intersections of Computer Architecture and Reconfigurable Logic. Davis, USA, 2013

[53] Chung E S, Davis J D, Lee J. LINQits: Big data on little clients//Proceedings of the 40th Annual International Symposium on Computer Architecture. Tel-Aviv, Israel, 2013: 261-272

[54] Venieris S I, Alexandros K, Christos-Savvas B. Toolflows for mapping convolutional neural networks on FPGAs. ACM Computing Surveys, 2018, 51(3): 1-39

[55] Geng T, Wang T, Sanaullah A, et al. A framework for acceleration of CNN training on deeply-pipelined FPGA clusters with work and weight load balancing//Proceedings of the 28th International Conference on Field Programmable Logic and Applications (FPL). Dublin, Ireland, 2018: 394-403

[56] Liu Z, Dou Y, Jiang J, et al. Automatic code generation of convolutional neural networks in FPGA implementation//Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT). Xi'an, China, 2016: 61-68

[57] Fraser N J, Umuroglu Y, Gambardella G, et al. Scaling binarized neural networks on reconfigurable logic//Proceedings of the 8th Workshop and 6th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms. Stockholm, Sweden, 2017: 25-30

[58] Umuroglu Y, Fraser N J, Gambardella G, et al. FINN: A framework for fast, scalable binarized neural network inference//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2017: 65-74

[59] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift//Proceedings of the International Conference on Machine Learning. Lille, France, 2015: 448-456

[60] Hameed R, Qadeer W, Wachs M, et al. Understanding sources of inefficiency in general-purpose chips//Proceedings of the 37th Annual International Symposium on Computer Architecture. New York, USA, 2010: 37-47

[61] Ma Y, Cao Y, Vrudhula S, et al. An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks//Proceedings of the 27th International Conference on Field Programmable Logic and Applications (FPL). Ghent, Belgium, 2017: 1-8

[62] Guan Y, Liang H, Xu N, et al. FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates//Proceedings of the 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). Napa, USA, 2017: 152-159

[63] Zeng H, Chen R, Zhang C, et al. A framework for generating high throughput CNN implementations on FPGAs//Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2018: 117-126

[64] Ma Yufei, Cao Yu, Vrudhula S, Seo J S. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'17). New York, USA, 2017: 45-54

[65] Ma Y, Kim M, Cao Y, et al. End-to-end scalable FPGA accelerator for deep residual networks//Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS). Los Alamitos, USA, 2017: 1-4

[66] Ma Y, Suda N, Cao Y, et al. Scalable and modularized RTL compilation of convolutional neural networks onto FPGA//Proceedings of the 26th International Conference on Field Programmable Logic and Applications (FPL). Lausanne, Switzerland, 2016: 1-8

[67] Ma Y, Suda N, Cao Y, et al. ALAMO: FPGA acceleration of deep learning algorithms with a modularized RTL compiler. Integration, 2018, 62: 14-23

[68] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA, 2016: 770-778

[69] Zeng H, Zhang C, Prasanna V. Fast generation of high throughput customized deep learning accelerators on FPGAs//Proceedings of the 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig). Cancun, Mexico, 2017: 1-8

[70] Zeng H, Chen R, Prasanna V K. Optimizing frequency domain implementation of CNNs on FPGAs. University of Southern California, Technical Report, 2017

[71] Zhang C, Prasanna V. Frequency domain acceleration of

convolutional neural networks on CPU-FPGA shared memory system//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. New York, USA, 2017: 35-44

[72] Davis J J, Cheung P Y K. Reducing overheads for fault-tolerant datapaths with dynamic partial reconfiguration//Proceedings of the 22nd Annual International Symposium on Field-Programmable Custom Computing Machines. Boston, USA, 2014: 102-103

[73] Ye H C, Chen G S. A resource-sharing & pipelined design scheme for dynamic deployment of CNNs on FPGAs//Proceedings of the 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT). Shandong, China, 2018: 1-3

[74] Hemmat M, Davoodi A. Dynamic reconfiguration of CNNs for input-dependent approximation//Proceedings of the 20th International Symposium on Quality Electronic Design (ISQED). Santa Clara, USA, 2019: 176-182

[75] Kästner F, Janßen B, Kautz F, et al. Hardware/software codesign for convolutional neural networks exploiting dynamic partial reconfiguration on PYNQ//Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). Vancouver, Canada, 2018: 154-161

[76] Guo K, Zeng S, Yu J, et al. [DL] A survey of FPGA-based neural network inference accelerators. ACM Transactions on Reconfigurable Technology and Systems, 2019, 12(1): 1-26

[77] Motamedi M, Gysel P, Akella V, et al. Design space exploration of FPGA-based deep convolutional neural networks//Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC). Macao, China, 2016: 575-580

[78] Motamedi M, Gysel P, Ghiasi S. PLACID: A platform for FPGA-based accelerator creation for DCNNs. ACM Transactions on Multimedia Computing, Communications, and Applications, 2017, 13(4): 1-21

[79] Chen Gui-Lin, Ma Sheng, Guo Yang. A survey of neural network hardware acceleration. Journal of Computer Research and Development, 2019, 56(2): 240-253(in Chinese)
(陈桂林, 马胜, 郭阳. 硬件加速神经网络综述. 计算机研究与发展, 2019, 56(2): 240-253)

[80] Qin Zhi-Yong. Key technology and challenge of neural network accelerator in aerospace//Proceedings of the 6th Aerospace Electronics Strategic Research Forum. Beijing, China, 2019, (1): 28-31(in Chinese)
(秦智勇. 航天神经网络加速器关键技术与挑战//中国航天电子技术研究院科学技术委员会. 第六届航天电子战略研究论坛论文集. 北京, 中国, 2019, (1): 28-31)

WANG Chao, Ph. D., associate professor. His research interests focus on deep learning processor.

WANG Teng, M. S. His research interests focus on neural network accelerator.

MA Xiang, M. S. His research interests focus on neural network accelerators.

ZHOU Xue-Hai, Ph. D., professor. His research interests focus on embedded systems and high performance computing.

Background

With the increasing production of massive data information and the widespread use of data mining applications, how to access data information efficiently and steadily and speed up the implementation of data mining applications have become the key issues that need to be solved urgently in academia and industry. Machine learning algorithms as the core component of data mining applications, using existing hardware and software means to accelerate machine learning algorithms has become a research hotspot. Current acceleration platforms can be summarized into four categories: custom logic circuits (such as FPGA/ASIC), general graphics processing unit (GPGPU), cloud computing platform and heterogeneous computing platform. These acceleration platforms often show different parallel granularity and are suitable for different application scenarios. They can also combine to form heterogeneous systems to give full play to the processing capabilities of different acceleration devices. Our research team has been working on high-performance computing and parallel computing for many years. This paper briefly reviews the history of accelerated machine learning computing. The design and implementation of various FPGA based machine learning accelerators are introduced in order from customized to general. Then, we summarize the state of the art of parallel accelerated computing based on FPGA research work. The future research directions are also considered and analyzed.