深度神经网络并行化研究综述

朱虎明 李 佩 焦李成 杨淑媛 侯 彪

(西安电子科技大学智能感知与图像理解教育部重点实验室 西安 710071)

(西安电子科技大学智能感知与计算国际联合研究中心 西安 710071)

(西安电子科技大学智能感知与计算国际合作联合实验室 西安 710071)

摘 要 神经网络是人工智能领域的核心研究内容之一.在七十年的发展历史中,神经网络经历了从浅层神经网络到深度神经网络的重要变革.深度神经网络通过增加模型深度来提高其特征提取和数据拟合的能力,在自然语言处理、自动驾驶、图像分析等问题上相较浅层模型具有显著优势.随着训练数据规模的增加和模型的日趋复杂,深度神经网络的训练成本越来越高,并行化成为增强其应用时效性的重要技术手段.近年来计算平台的硬件架构更新迭代,计算能力飞速提高,特别是多核众核以及分布式异构计算平台发展迅速,为深度神经网络的并行化提供了硬件基础;另一方面,日趋丰富的并行编程框架也为计算设备和深度神经网络的并行化架起了桥梁.该文首先介绍了深度神经网络发展背景和常用的计算模型,然后对多核处理器、众核处理器和异构计算设备分别从功耗、计算能力、并行算法的开发难度等角度进行对比分析,对并行编程框架分别从支持的编程语言和硬件设备、编程难度等角度进行阐述.然后以 AlexNet 为例分析了深度神经网络模型并行和数据并行两种方法的实施过程.接下来,从支持硬件、并行接口、并行模式等角度比较了常用的深度神经网络开源软件,并且通过实验比较和分析了卷积神经网络在多核 CPU 和 GPU 上的并行性能.最后,对并行深度神经网络的未来发展趋势和面临的挑战进行展望.

关键词 深度神经网络;并行计算;异构计算;模型并行;数据并行

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2018.01861

Review of Parallel Deep Neural Network

ZHU Hu-Ming LI Pei JIAO Li-Cheng YANG Shu-Yuan HOU Biao

(*Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an* 710071)

(*International Research Center of Intelligent Perception Computation, Xidian University, Xi'an* 710071)

(*International Collaboration Joint Laboratory in Intelligent Perception and Computation, Xidian University, Xi'an* 710071)

Abstract Neural network is one of the main fields of research in artificial intelligence. In the past seventy years, the development of neural network has experienced two important stages: shallow neural network and deep neural network. Overfitting usually occurs when shallow neural networks have been used to solve very complex problems. Compared with shallow neural network, deep neural network has obvious advantages in feature extraction and data fitting while becoming deeper. In the meanwhile, deep neural network has been widely applied in many industry areas, such as nature language processing, automatic drive and image analysis and so on. However, the training cost of deep neural network grows with the increasing the training data size and complexity of neural network models. In addition parallelization has become a necessity to reduce

收稿日期:2017-05-07;在线出版日期:2018-01-03.本课题得到国家自然科学基金(61303032)、国家"九七三"重点基础研究发展计划项目基金(2013CB329402)、国家自然科学基金重大研究计划(91438201,91438103)和教育部"长江学者和创新团队发展计划"(IRT_15R53)资助.**朱虎明**,男,1978 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为高性能计算及其应用和大规模并行机器学习等.E-mail:zhuhum@mail.xidian.edu.cn.**李 佩**,女,1992 年生,硕士研究生,主要研究方向为高性能计算.**焦李成**,男,1959 年出生,博士,教授,博士生导师,IEEE Fellow,中国计算机学会(CCF)高级会员,主要研究领域为智能感知、图像理解等.**杨淑媛**,女,1978 年生,博士,教授,博士生导师,主要研究领域为智能信号与图像处理、机器学习等.**侯 彪**,男,1974 年生,博士,教授,博士生导师,主要研究领域为合成孔径雷达图像处理.

the training time of deep neural networks recently. The architecture quick update of computing platforms and the leap in their computing capability, especially the development of multi-core and many-core computing devices and emergence of distributed heterogeneous computing technology provide suitable hardware resources for accelerating deep neural network. A parallel programming model allow users to specify different kinds of parallelism that could easily be mapped to parallel hardware architectures and that facilitate expression of parallel algorithms. On the other hand, parallel programming frameworks rapidly evolve to meet the performance demands of high performance computing with continuous development. Combining an appropriate pair of software and hardware and fully exploiting fine-grained and coarse-grained parallelism for efficient neural network acceleration are the challenging tasks. This paper starts with an introduction of neural network model, optimization algorithm for solving cost function, the most popular open source software framework and research progress in both academia and industry. This paper then discusses hardware platforms and parallel programming model for applications of deep neural network. Hardware such as multi-core central processing unit, graphics processing unit, many integrated core, field-programmable gate array and application-specific integrated circuit is summarized from three aspects of power, computing power and challenges in developing parallel algorithms. Parallel programming models, whose open source and commercial implementations include compute unified device architecture, open computing language, open multiple processing, message passing interface and Spark, are compared in programming language, available hardware and programming difficulty for parallelizing neural network. In addition, the principle of deep neural network model and data parallelization is described and how can use it to parallelize AlexNet is provided. Then a comparison of the six open source software system for deep neural network is presented in the parallelization strategies, supporting hardware, parallel mode and so on. The software systems under consideration are Caffe, TensorFlow, MxNet, CNTK, Torch and Theano. Next, the state-of-the-art papers of parallel neural network are reviewed and multi-level parallel methods for the training and inference process on different computing devices are summarized. Parallel convolutional neural network for image classification is implemented by TensorFlow, and the experiments are conducted in multi-core central processing unit and graphics processing unit with classic MNIST and CIFAR10 dataset to demonstrate the acceleration capability. Finally, future directions and challenges of parallel deep neural networks are discussed and analyzed.

Keywords deep neural network; parallel computing; heterogeneous computing; model parallelism; data parallelism

1 引 言

实现人工智能是人类长期以来一直追求的梦想,而神经网络是人工智能[1-2]研究领域的核心之一.它的发展经历了两个重要阶段,浅层神经网络阶段和深度神经网络阶段.浅层模型只有一层隐含层或者没有隐含层节点,理论分析相对成熟,在许多应用中获得成功,如网页搜索排序和各类推荐系统等.然而随着样本数量的增大以及特征维度的增加,浅层模型逐渐不能满足应用的需求.自 2006 年 Hinton 和 Salakhutdinov[3]在《Science》杂志上发表的论文解决了多层神经网络训练的难题后,学术界和工业界对深度神经网络的研究热情高涨,并取得了突破性进展.

近年来,深度神经网络由于优异的算法性能,已经广泛应用于图像分析、语音识别、目标检测、语义分割、人脸识别、自动驾驶等领域[4-9].深度神经网络之所以能获得如此巨大的进步,其本质是模拟人脑的学习系统,通过增加网络的层数让机器从数据中学习高层特征,目前网络的深度有几百层甚至可达上千层,日趋复杂的网络模型为其应用的时效性带

来了挑战.为减少深度神经网络的训练时间,基于各种高性能计算平台设计并行深度神经网络算法逐渐成为研究热点.

由于功耗墙的存在,用于计算的 CPU 硬件架构从单核变成多核,并行计算的方式也从指令级并行变成线程级并行.近年来,以 GPU、MIC 和 FPGA 为代表的异构计算平台具有高能效的优点,被广泛用于不同研究领域的加速.异构计算的理念也被用来制造超级计算机[10],如我国研制的异构计算集群天河一号(采用 GPU 作为加速器)其计算能力在 2010 年 11 月排世界第一,天河二号(采用 MIC 作为协处理器)其计算能力在 2013 年 6 月排世界第一.采用国产异构多核 CPU 制造的神威太湖之光超级计算机在 TOP500 世界超算大会夺得世界第一.由于并行计算设备的多样性,并行编程框架也日趋丰富,目前主要的编程框架有统一计算设备架构(Compute Unified Device Architecture,CUDA)、开放计算语言(Open Computing Language,OpenCL)、共享存储编程模型(Open Multiple Processing,OpenMP)、消息传递接口(Message Passing Interface,MPI)、Spark 等[11].硬件的快速发展为深度神经网络的并行化提供了物质基础,并行编程框架为其并行化架起了桥梁,因此如何结合深度神经网络的算法特点,利用并行编程框架来设计能充分发挥多核众核平台计算能力的并行化方法显得十分迫切.

本文首先简要地介绍了深度神经网络发展的背景,然后介绍了目前多核众核计算平台和并行编程框架的发展概况,并对深度神经网络的模型并行、数据并行以及目前常用的开源深度学习软件系统中的并行化方法进行阐述,同时对当前深度神经网络的并行化研究工作进行归纳总结,在此基础上展望了深度神经网络并行化可能的发展方向以及面临的挑战.

2　神经网络发展概况

神经元是人工神经网络的基本处理单元,一般是多输入单输出的单元,结构模型如图 1 所示,其中,x_i 表示输入信号,w_{ij} 表示输入信号 x_i 与神经元 j 之间的连接权重,b_j 为神经元的偏置,y_j 表示神经元的输出.输入信号与输出值之间的对应关系式如下式所示:

$$y_j = f\left(b_j + \sum_{i=1}^{n}(x_i \times w_{ij})\right) \tag{1}$$

$f(\cdot)$ 为激活函数,一般可用 Sigmoid 函数、ReLU 函数、$\tanh(x)$ 函数、径向基函数等.常用的神经网络有多层感知器(MultiLayer Perceptron,MLP)[12]、限制玻尔兹曼机(Restricted Boltzmann Machine,RBM)[13]、径向基神经网络 RBF[14]等.

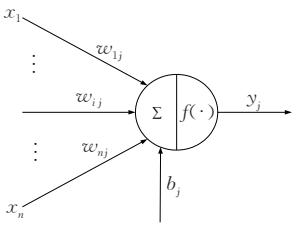


图 1　神经元结构模型图

MLP 是一种非线性分类器,是由输入层、隐含层(一层或多层)以及输出层构成的神经网络模型.图 2 给出了包含一个隐含层的多层感知器网络拓扑结构.输入层神经元接收输入信号,层与层之间是全连接,每个连接都有一个连接权值,同层间的神经元互不相连.图中 w_{ij}^{k} 表示第 k 层神经元 i 与第 $k+1$ 层神经元 j 之间的连接权重,y_m^l 表示第 l 层神经元 m 的输出.在 MLP 的训练过程中,将一个特征向量作为输入,将该向量传递到隐含层,然后通过权重和激活函数计算结果,并将结果传递到下一层,直到最后传递给输出层,可通过 BP(Back Propagation)算法[15]对网络权重进行微调.

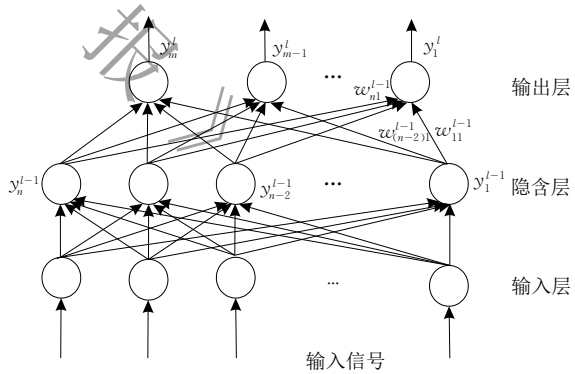


图 2　MLP 网络拓扑结构

RBM 是由 Smolensky 在 1986 年提出的一种随机的产生式的人工神经网络模型,能够依据输入数据,学习得到一种概率分布.标准型的 RBM 具有二值的隐藏单元和可视单元,图 3 网络结构中包含 i 个可视节点和 j 个隐藏节点,其中每个可视节点只和 j 个隐藏节点相关,可视节点的状态只受 j 个隐藏节点的影响,对于每个隐藏节点,只受 i 个可视节点的影响.

由于训练时数据数量少,再加上传统的神经网

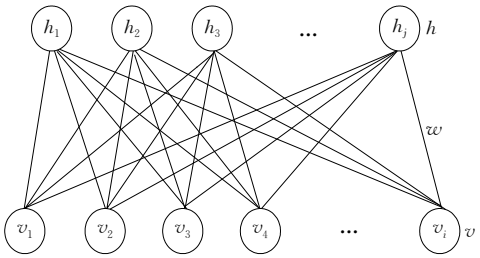


图 3 RBM 结构示意图

络是一个浅层的结构，网络简单，对输入的表达能力有限，因此容易过拟合. 2006 年加拿大多伦多大学的 Hinton 教授提出了基于“逐层训练”和“精调”的两阶段策略，解决了深度神经网络中参数训练的难题[3]. 之后纽约大学的 LeCun、蒙特利尔大学的 Bengio 和斯坦福大学的 Ng 等人对深度神经网络展开研究[16]，并提出了深度自编码器(Deep Auto Encoder，DAE)[17-19]、深度置信网(Deep Belief Networks，DBN)[20-22]、卷积神经网络(Convolutional Neural Networks，CNN)[23-25]等深度模型，且在多个领域得到广泛应用.

自编码器(Auto Encoder，AE)是以输出值等于输入值为目标的一种无监督的人工神经网络模型，通常包括一个输入层、一个隐层、一个输出层. 当自编码器包含多个隐层时即形成了 DAE，该网络的隐含层结点数通常明显少于输入节点数，构成一个压缩式的网络结构. DAE 学习过程中容易出现过拟合问题，一种有效的解决方法是去噪自动编码器(Denoising autoencoder)[26]，即人为在训练样本上施加噪声作为网络输入，输出依然为无噪声的样本，由此学到的网络对噪声具有很好的鲁棒性.

DBN 是由多层无监督的 RBM 和一层有监督的 BP 网络组成的一种生成模型. 其训练过程通常是贪婪式的逐层训练，在预训练阶段采用逐层训练的方式对各层中的 RBM 进行训练，不仅使得 DBN 的高效学习成为可能，而且还可以避免网络收敛到局部最优. 微调阶段采用有监督的学习方式，利用 BP 网络对 RBM 通过预训练得到的特征向量进行分类，在 BP 的前向传播过程中，输入特征向量被逐层传播到输出层，得到预测的分类类别. 将实际得到的分类结果与期望值比较得到误差，并将该误差逐层向后回传进而对整个网络的权值进行微调. 在具体的应用领域，微调阶段的目标函数可以是无监督的或有监督的方法.

CNN 是从生物学上视觉皮层的研究中获得启发而产生的，其重要特性是通过局部感受野、权值共享以及时间或空间亚采样等思想减少了网络中自由参数的个数，从而获得了某种程度的位移、尺度、形变不变性. 图 4 所示为经典深度 CNN 网络结构图，其包含 5 个卷积层和 3 个全连接层，每个卷积层包含了激活函数 ReLU 以及局部响应归一化处理，然后再经过降采样(重叠池化处理). 该网络引入了新的非线性激活函数 ReLU 替代之前普遍采用的 Sigmoid 或 tanh 函数，有利于更快速的收敛，减少了训练时间，同时在最后两个全连接层引入 Dropout 防止过学习的训练策略. 在卷积层之后，高层逻辑推理通过全连接层完成，即全连接层的神经元与前一层的所有输出相连接. 全连接层后还需要使用代价函数来度量深度神经网络训练输出值和真实值之间的差异，在不同的应用中使用不同的代价函数[27].

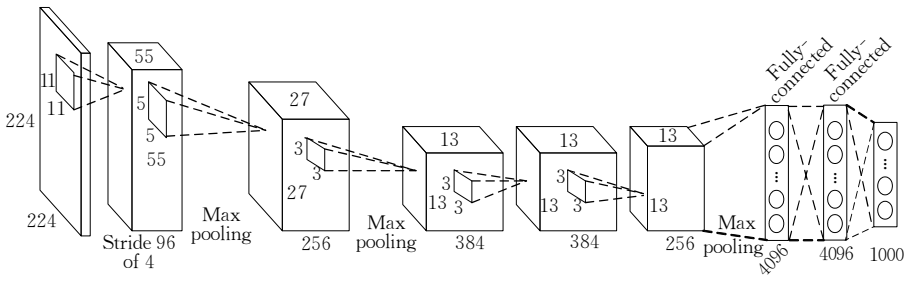


图 4 经典的 CNN 网络结构

神经网络常用的代价函数有二次代价函数、交叉熵和对数似然函数. 二次代价函数和交叉熵代价函数分别定义为式(2)和(3)，其中 x 为样本，n 为样本总数，y 表示期望输出值，a 为输出值. 交叉熵函数相比于二次函数具有收敛速度快，更容易获得全局最优的特点. 使用 Softmax 作为激活函数时，常使用对数似然函数作为代价函数，定义如式(4)，其中 a_k 为第 k 个神经元的输出值，y_k 为第 k 个神经元对应的真实值，取值为 0 或 1.

$$C=\frac{1}{2n}\sum_x \|y(x)-a(x)\|^2 \tag{2}$$

$$C=-\frac{1}{n}\sum_x [y\ln a+(1-y)\ln(1-a)] \tag{3}$$

$$C=-\sum_k y_k \log a_k \tag{4}$$

在深度神经网络中，为了求解代价函数，需要使用优化算法，常用的算法有梯度下降法、共轭梯度法、LBGFS 等[28]，目前最常用的优化算法是梯度下降算法，该算法的核心是最小化目标函数，在每次迭代中，对每个变量按照目标函数在该变量梯度的相反方向更新对应的参数值. 其中，参数学习率决定了函数到达最小值的迭代次数. 梯度下降法有三种不同的变体：批量梯度下降法(Batch Gradient Descent，BGD)、随机梯度下降法(Stochastic Gradient Descent，SGD)、小批量梯度下降法(Mini-Batch Gradient Descent，MBGD). 对于 BGD，能够保证收敛到凸函数的全局最优值或非凸函数的局部最优值，但每次更新需在整个数据集上求解，因此速度较慢，甚至对于较大的、内存无法容纳的数据集，该方法无法被使用，同时不能以在线的形式更新模型；SGD 每次更新只对数据集中的一个样本求解梯度，运行速度大大加快，同时能够在线学习，但是相比于 BGD，SGD 易陷入局部极小值，收敛过程较为波动；MBGD 集合了上面两种方法的优势，在每次更新时，对 n 个样本构成的一批数据求解，使得收敛过程更为稳定，通常是训练神经网络的首选算法.

近年来，各种深度神经网络模型如雨后春笋般涌现出来，如 Krizhevsky 等人[29]在 2012 年设计的包含 5 个卷积层和 3 个全连接层的 AlexNet，并将卷积网络分为两部分，在双 GPU 上进行训练；2014 年 Google 研发团队[30]设计的 22 层的 GoogleNet，同年牛津大学的 Simonyan 和 Zisserman[31]设计出深度为 16 层～19 层的 VGG 网络，2015 年微软亚洲研究院的何凯明等人[32]提出了 152 层的深度残差网络 ResNet，而最新改进后的 ResNet 网络深度可达 1202 层，2016 年生成式对抗网络 GAN 获得广泛关注[33]. 随着网络模型种类的逐渐增多，网络深度也从开始的几层到现在的成百上千层，虽然大大提高了精确率，但也使得深度神经网络的训练时间越来越长，成为其快速发展和广泛应用的一大阻碍.

学术界和产业界对相关研究成果的开源加速了深度神经网络的快速发展. 其中，代表性的开源平台有[34]：加州大学伯克利分校的贾扬清开发的 Caffe 具有速度快、能够支持不同硬件平台(GPU、CPU)的优点，但是由于遗留的架构问题，使得它不够灵活且对递归网络和语言建模支持很差；2017 年 4 月开源的 Caffe2，新增加的特性包括采用了计算图来表示神经网络，支持 IOS 和 Android 系统，增加了自然语言处理、手写识别和时间序列预测的循环神经网络 RNN 和长短期记忆网络 LSTM，支持多节点多 GPU 并行. Facebook 开发的 Torch 实现并且优化了基本的计算单元，有较好的灵活性，缺点是接口为 lua 语言，需要花费时间进行学习. 2017 年新版本 PyTorch 使用 Python 作为前端，因此可以方便使用 Python 的相关机器学习库，PyTorch 支持动态构造计算图和自动求导等功能. 李牧等开源的 MXNet 注重灵活性和效率，同时提高了内存的使用率，支持 Android 系统；蒙特利尔大学开发的 Theano 是第一个使用符号张量图描述模型的架构，派生出大量深度学习 Python 软件包，非常灵活且对递归网络和语言建模有较好的支持；Google 开源的第二代深度学习框架 TensorFlow 使用了张量运算的计算图方法，支持异构分布式计算，具备很好的灵活性和可扩展性，目前已经成为最热门的深度学习开发平台；CNTK 是微软开源的深度学习工具包，将神经网络描述为有向图的结构，叶子结点表示输入或者网络参数，其它节点代表计算步骤，同时支持 GPU 和 CPU.

深度神经网络获得了学术界和工业界的广泛关注，在算法研究和应用方面不断取得进展. 在学术研究方面，2015 年《Nature》刊出了由 LeCun、Bengio 和 Hinton[35]撰写的深度学习综述文章，同年举办的知名学术会议 CVPR、NIPS、AAAI 和 ICLR 中深度神经网络的主题占主导地位. 2017 年以深度神经网络为核心的 DeepStack 算法在德州扑克游戏中击败人类职业玩家[36]. 在国内，相关的高校和科研单位在深度神经网络的研究和应用方面也取得了丰硕的成果，清华大学、中国科学院、百度和西安电子科技大学给出了深度学习[16,37-40]的研究综述，相关高校的研究学者将深度学习算法成功应用到遥感图像分类、多媒体检索、交通流预测和盲图像质量评价[41-44]等领域.

除了重要的学术研究意义，工业界在应用方面也成果丰硕，如 2016 年基于深度神经网络的 AlphaGo 在分布式框架中最多采用 1920 块 CPU 和 280 块 GPU，最终打败了围棋世界冠军李世石[45]，目前 Google、Facebook、Microsoft、IBM 等国际巨头以及国内的百度、阿里巴巴、腾讯、科大讯飞等互联网巨头争相布局深度学习，并且成功应用于多种产品之中，例如谷歌 Now、微软 OneNote 手写识别、Cortana 语音助手、讯飞语音输入法等.

虽然深度神经网络由于优异的性能得到了广泛的使用,在许多的应用领域取得了成功,但其日趋复杂的网络模型为其应用的时效性带来了挑战.为减少深度神经网络的训练和测试时间,针对各种应用场景,在不同架构的并行计算硬件平台上,利用合适的软件接口来设计并行深度学习计算系统并优化其性能成为研究热点.

3 软硬件发展概况

3.1 硬件架构

用于深度学习的硬件设备在计算性能、并行机理和应用开发周期等方面存在巨大的差异,计算能力和带宽是决定硬件应用性能的主要因素.计算能力指标一般使用 FLOPS,即每秒执行的 32 位浮点(FP32)运算次数.CPU 计算性能一般采用的指标为每秒十亿次浮点运算(Giga Floating Point Operations per Second,GFLOPS)和每秒万亿次浮点运算(Tera Floating Point Operations per Second,TFLOPS).但是 FPGA 用户可以自定义各种精度的数据类型,数据可能是 FP32,INT16、INT8 等多种格式,不一定是浮点计算能力,经常使用每秒十亿次运算(Giga Operations per Second,GOPS)和每秒万亿次运算(Tera Operations per Second,TOPS)指标.

(1)多核 CPU:随着频率墙、功耗墙、存储墙等问题越来越突出,单核 CPU 很难继续通过提高时钟频率来提升性能,因此具有多核结构的计算设备逐渐成为主流.多核结构指的是在同一个处理器上集成两个或两个以上的计算内核,不同计算内核之间相互独立,可以并行的执行指令[46].以 2017 年发布的 Intel Xeon E7-8894 v4 为例,其采用 14 nm 工艺,包含 24 个物理核,最高支持 48 线程,最大内存带宽 85 GB/s,单精度浮点计算能力为 921.6 GFLOPS,支持 AVX 2.0 指令,可以实现一个时钟周期处理八个浮点数的乘加操作,实现单指令多数据流并行,有效提高程序的执行效率.

(2)GPU:GPU 以前主要用于图形学处理,由于其强大的计算能力现在已经被用作加速器来加速计算密集型应用.GPU 是以大量线程并行执行面向高吞吐量设计,具有高带宽、高并行性的特点,因此 GPU 适用于大量数据的并行计算[47].目前 GPU 厂商主要有 NVIDIA、AMD、Intel、高通、ARM 等公司,不同公司生产的 GPU 在硬件架构、功耗、性能以及应用场景等方面存在巨大的差异.比如高通的 Adreno 和 ARM 的 Mali 架构最新嵌入式 GPU 功耗只有几瓦,计算能力大约 500 GFLOPS.而 NVIDIA 公司最新的 Tesla 系列的 Volta 100 采用 12 nm FFN 工艺制造,有 5210 个流处理器,可提供 7.5 TFLOPS 的双精度计算能力和 15 TFLOPS 单精度性能.针对深度学习新增加了支持混合精度 FP16/FP32 计算的 640 个 Tensor Core,能够为训练、推理应用提供 120 Tensor TFLOPS 计算能力.GPU 设计了鲜明的层次式存储,使用好层次式存储是进行性能优化的关键.GPU 存储单元包括全局存储、纹理存储、常量存储、共享存储、局部存储、寄存器等,各存储单元的使用依赖于算法的访存模式和存储单元的特性.到目前为止,NVIDIA 通用计算的 GPU 产品经历了 Tesla、Fermi、Kepler、Maxwell、Pascal、Volta 等架构,架构的快速迭代,相应的硬件逻辑发生变化,为算法实施和性能优化带来进一步的挑战.

(3)MIC:Intel 公司 2012 年推出了第一代 MIC 架构的 Xeon Phi 融合处理器 KNC(Knights Corner),拥有 57 个以上的 CPU 物理核心,每个物理核心可以并行执行 4 个硬件线程,核心频率约为 1.1 GHz,并且包含一个 512 bit 线宽的矢量处理单元 VPU,可提供约 1 TFLOPS 双精度峰值计算能力;2016 年推出的第二代 MIC 架构的 Xeon Phi 融合处理器 KNL(Knights Landing)[48],核心频率约 1.3 GHz,双精度浮点性能超过 3 TFLOPS,单个芯片最大支持 72 个 CPU 物理核心,每个物理核心支持 4 个硬件线程,并包含 2 个 512 bit 线宽的 VPU,能更有效率的提升整体性能、满足高度并行化的高性能计算应用,但是需要考虑硬件架构的特性以优化代码才能充分发挥硬件性能.

(4)FPGA:由成百上千的加法器、乘法器和数字信号处理器 DSP 等模块组成,可以同时进行大规模的并行运算,具有高性能、低功耗、可编程等特点.FPGA 作为一种计算密集型加速部件,通过将算法分解并映射到相应的硬件模块上进行加速.流水结构是 FPGA 作为加速器的一大优势,可以很好地和深度神经网络算法相匹配,充分利用算法网络结构内部的并行性,提高运算速度的同时减小功耗.Altera公司的 Stratix 10 系列产品采用 14 nm 三栅极工艺开发,最高可配置 5760 个 DSP,11 520 个 18×19 规格的乘法器,单精度浮点性能达到 10 TFLOPS.Xilinx 的 UltraScale 系列产品[49]最高可配置 12 288 个 DSP,其支持的 INT8 低精度计算相比 INT16 计算能效得到了提升.

(5)专用加速器：为了最大化计算速度和最小化能量消耗，针对深度学习设计专用集成电路(Application-Specific Integrated Circuit，ASIC)并将其应用于大规模云计算数据中心和嵌入式计算设备成为热门研究方向.专用加速器具有高性能、低功耗、面积小等特点，而且量化生产时成本低，主要缺点是其设计周期比较长.Intel 已经面向市场推出了深度神经网络加速器——神经计算棒(Neural Compute Stick，NCS)，其采用 28 nm 工艺生产，支持 FP16 精度和 Caffe 深度学习软件，计算能力为 100 GFLOPS，功耗 1 W.

表 1 对以上硬件架构从功耗、计算能力、并行算法开发的难易程度等角度进行比较.

表 1　硬件架构对比分析

硬件	硬件型号	功耗/W	计算能力(FLOPS)	开源软件支持	并行算法开发难度	开发周期
CPU	Intel Xeon E7-8894 v4	165	921.6G SP	好	容易	短
GPU	Tesla V100	300	15T SP	好	中等	短
MIC	Xeon Phi7250	215	3T DP	少	容易	中
FPGA	Stratix 10	75	10T SP	少	难	较长
ASIC	TPU	45	23T 16 位整数矩阵乘法	少	难	长

3.2　并行编程框架

(1)CUDA 是 2007 年由 NVIDIA 公司推出的只能运行在本公司各种型号 GPU 上的并行编程语言[50]，使用扩展的 C 语言来进行 GPU 编程.自 2007 年 CUDA 1.0 版本诞生后，由于大大降低了 GPU 通用编程的难度，因此大量的研究者尝试利用 GPU 加速各个领域的算法.此后 CUDA 版本快速迭代，通用计算能力越来越强，比如 2009 年的 CUDA 3.0 加入了对 C++编程语言的支持，2012 年的 CUDA 5.0 增加了动态并行的新特性，2013 年的 CUDA 6.0 支持统一寻址，而 2017 年最新发布的 CUDA 8.0.61 支持 GPU 直接同步，使得 GPU 可以在没有 CPU 辅助的情况下交换数据.CUDA 并行编程模型采用两级并行机制，即 Block 映射到流多处理器并行执行和同一个 Block 内的 Thread 映射到流多处理器的 CUDA 核上并发执行.

(2)OpenCL 是 Khronos 组织制定的异构计算统一编程标准[51]，得到了 AMD、Apple、Intel、NVIDIA、TI 等公司的支持，因此可以运行在多核 CPU、GPU、DSP、FPGA 以及异构加速处理单元上.OpenCL 计算模型在具体硬件上执行时，由各个厂家的运行环境负责将代码在线编译成机器码并建立软硬件映射机制.当 OpenCL 执行的核心 kernel 启动后会创建大量的线程同时执行，每个线程即工作单元(work-item)完成 kernel 函数定义的操作.当映射到 OpenCL 硬件上执行时，采用两级并行机制，work-group 并发运行在异构计算设备的计算单元上，同一个 work-group 里的多个 work-item 相互独立并在处理单元上并行执行.

(3)OpenMP 是基于共享内存和多线程的并行编程模型，在使用时需要程序员在程序可用于并行的部分添加并行编译的关键字，运行环境依据关键字将计算任务映射到多线程上并行执行.OpenMP 并行技术具有良好的可移植性，不需要对串行代码进行大量的修改，降低了并行编程的难度和复杂度，具有很强的灵活性，可以较容易的适应不同的并行系统配置.然而 OpenMP 程序是通过将 for 循环分解到多个线程上进行并行，可以通过调整线程数目和调度方式(静态调度、动态调度)等手段来优化性能.执行并行任务的线程调度由 OpenMP 运行环境控制，因此当线程数量很多时，并行程序的可扩展性经常表现一般[52].在具体应用时要想提高并行效率，需要根据算法特点和硬件平台的特性，对代码进行一定的优化.

(4)MPI 是一种基于消息传递的并行程序编程框架[53].MPI 主要应用于集群计算环境，其支持的计算节点数可达上万个.作为一个跨语言的通讯协议，MPI 支持点对点和广播两种通信方式.MPI 程序执行时，在集群的每个节点上启动多个进程，节点间的进程通过高速通信链路(如以太网或 InfiniBand)显式地交换消息，并行协同完成计算任务.MPI 广泛使用在高性能计算行业，但是基于 MPI 的并行程序通常在算法上有较大改动，编程难度较大，并且容错性不足，如果一个进程出现问题导致整个应用需要重新进行计算.

(5)Spark 是加利福尼亚大学伯克利分校 AMP 实验室在 2009 年开源的一种通用并行计算框架，最大可支持上千个节点的并行数据处理[54].它扩展了广泛使用的 MapReduce 计算模型，并添加了交互式查询以及流处理功能，同时支持 Scala、Java 和

Python语言，易于使用，能够与 Yarn、Mesos、Hive、HBase、HDFS 等多个框架进行很好的兼容. Spark 将各种类型的数据结构都统一抽象为 RDD 结构. 它基于内存计算的特点使其与 Hadoop 相比，在迭代式算法上优势明显，高效的容错机制使其在面对故障问题时可及时恢复正常. 目前 Spark 已被广泛应用于大数据处理领域.

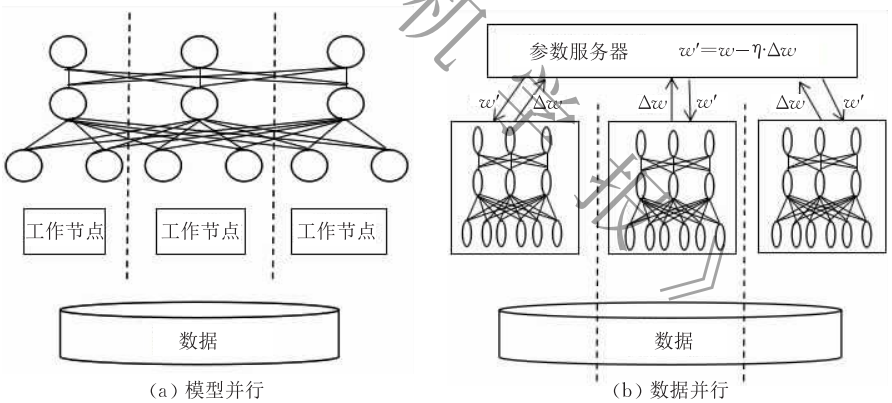
表 2 对这几种并行编程框架从开放性、开发难度和深度学习软件支持力度等方面进行分析比较.

表 2 并行编程框架的比较

编程框架	出现时间	开放性	主要支持语言	支持硬件	编程难度	开源深度学习软件支持
CUDA	2007	企业私有	C,C++	NVIDIA GPU	容易	Caffe,TensorFlow,MXNet,CNTK,Torch,Theano
OpenCL	2008	API 标准	C,C++	GPU,CPU FPGA,DSP MIC	难	Caffe,Theano
OpenMP	1997	API 标准	C,C++,Fortran	CPU,MIC	容易	Theano
MPI	1992	API 标准	C,C++,Fortran	CPU,MIC	难	CNTK,S-Caffe
Spark	2010	开源软件	Java,Scala,python	CPU	容易	Intel BigDL,SparkNet

4 深度神经网络的模型并行和数据并行

对深度神经网络的并行化目前主要有两种方法：模型并行和数据并行[55]，如图 5 所示. 模型并行(图 5(a))是指将网络模型分解到各个计算设备上，依靠设备间的共同协作完成训练. 数据并行(图 5(b))是指对训练数据做切分，同时采用多个模型实例，对多个分片的数据并行训练，由参数服务器[56-57]来完成参数交换. 在训练过程中，多个训练过程相互独立，模型的变化量 Δw 需要传输给参数服务器，由参数服务器负责更新为最新的模型 $w'=w-\eta\cdot\Delta w$，其中 η 为学习率，然后再将最新的模型 w' 分发给训练程序. 多数情况下，模型并行带来的通信开销和同步开销超过数据并行，因此加速比也不及数据并行，但是对于单个计算设备内存无法容纳的大模型来说，模型并行是一个很好的选择.



(a) 模型并行 (b) 数据并行

图 5 深度神经网络的模型并行和数据并行原理图

随机梯度下降算法(SGD)由于使用简单、收敛速度快、效果可靠等优点在深度神经网络算法中得到了普遍应用. 在大数据背景下，深度神经网络的数据并行更多地是通过分布式随机梯度下降算法，对于该算法中参数更新方式的选择，目前主要有同步 SGD 和异步 SGD 两种机制. 同步 SGD[58-59]需要利用所有节点上的参数信息，而慢节点所带来的同步等待使得数据并行时的加速比并不理想. 异步 SGD 虽然单次训练速度快，但是其固有的随机性使得网络在训练过程中达到相同收敛点耗费的时间更长，且在训练后期可能会出现震荡现象. Xing 等人[59-61]针对机器学习中比较耗时的迭代算法，提出了一种新的协议 SSP 来缓解同步 BSP 中慢节点所带来的同步等待，通过引入一个参数来约束快节点和慢节点之间迭代步伐的差值. 相比于异步模式，同步等待开销一定程度上限制了网络训练速度. Goyal 等人[62]提出了一种将参数批量大小值提高的分布式同步 SGD 训练方法，采用了线性缩放规则(linear scaling rule)作为批量大小函数来调整学习率，在训练的开始阶段使用较小的学习率，在批量大小为 8192 时在 Caffe2 的系统上训练 ResNet-50 网络，训练数据集使用 ImageNet，该训练在 256 块 Tesla P100 GPU(实验硬件平台为：Facebook 的 Big Basin 服务器，每个服务器安装有 8 块 GPU 卡，卡之间使

用 NVLink 互联技术，服务器之间使用 50 Gbit 带宽的以太网连接）上花费 1 h 就能完成，识别精度与小批量相当.

深度神经网络的网络结构复杂，参数多，训练数据量大，这些都为并行化工作带来了挑战. 近年流行的卷积神经网络有 AlexNet、VGG、GoogleNet 和 ResNet 网络等，其网络结构信息如表 3 所示. 其中，AlexNet 网络为 8 层，拥有超过 6000 万个参数，训练使用的 ImageNet 数据集有 120 万张图片，为了加快训练速度和将来能使用更大的网络，Krizhevsky 等人使用了两个 NVIDIA GeForce GTX580 GPU 对其进行了模型并行. VGG 网络结构在 AlexNet 上发展而来，VGG 网络使用多个小滤波器卷积层（滤波器大小为 3×3）与激活层交替的结构来替代单个大滤波器卷积层，这样的结构能更好地提取出深层特征，VGG 网络有着比 AlexNet 更深的层数和更多的参数数量，如 VGG-19 网络有 19 层和 1 亿 3800 万个参数，因此 VGG 网络提供了比 AlexNet 更高的精度. GoogleNet 在卷积神经网络的基础上加入了 inception 模块，inception 模块将不同大小的滤波器和池化模块堆栈在一起，并使用较小尺寸的滤波器替代了大的滤波器. Inception 模块的使用使得 GoogleNet 既能保留网络结构的稀疏性，又能利用稠密矩阵的高计算性能，还能通过不断调整自身结构以加深网络的深度. Inception V2 加入了 Batch Normalization 技术来减少内部数据分布变化，并使用了两个 3×3 的卷积核来替代 5×5 的卷积核来减少参数数量[63]. InceptionV3 主要的思想就是分解大尺寸卷积为多个小卷积乃至一维卷积，比如将 7×7 的卷积核分解为一维的卷积（7×1，1×7），这种分解既减少了参数数量又减少了算法的计算量[64]. Inception V4 版本中将 Inception 和 ResNet 结合既加速训练又获得了性能提升[65]. ResNet 引入了残差网络结构解决了加深网络层数时梯度消失的问题，因此 ResNet 网络深度最高达到了 1202 层.

表 3　经典卷积神经网络结构参数

神经网络	网络层数	Top-5 错误/%	卷积层数	卷积核大小	全连接层数	全连接层大小	参数数量/M
AlexNet	8	16.40	5	11,5,3	3	4096,4096,1000	60.0
GoogleNet	22	6.70	21	7,1,3,5	1	1000	7.0
ResNet-152	152	4.49	151	7,1,3,5	1	1000	2.4
VGG-19	19	7.30	16	3	3	4096,4096,1000	138.0

下面以 AlexNet 为例简要说明模型并行和数据并行的特点和实施要点. AlexNet 共有 8 层，包括 5 个卷积层和 3 个全连接层. 表 4 列出了 AlexNet 的神经元数量、参数数量以及 Batchsize 为 128 时一次前向计算各层向下一层传输数据量的情况. 卷积层神经元个数计算公式为 $N_{cn}=T\times S^2$，其中 T 为卷积核个数，S 为卷积层池化操作前特征图大小. 卷积层参数个数计算使用公式 $N_{cw}=T\times K^2\times D+T$（偏置参数个数），$K$ 为卷积核大小，D 为卷积层输入特征图个数. 每个特征图上的卷积核偏置共享，因此偏置参数个数与卷积核个数相同，注意计算第二四五层时由于两个 GPU 不通信，因此参数数量要减半. 全连接层与卷积层连接时参数个数计算公式 $N_{fcw}=T\times s_p^2\times d_{cn}^l+d_{cn}^l\cdots$（偏置参数个数），其中 s_p 为卷积层池化操作后特征图大小，d_{cn}^l 为全连接层神经元个数. 全连接层与全连接层连接时参数个数计算公式为 $N_{ffw}=d_{cn}^{l-1}\times d_{cn}^l+d_{cn}^l$（偏置参数个数），其中 d_{cn}^l 为第 l 层神经元个数. 卷积层输出数据个数计算公式为 $N_{cd}=B\times T\times s_p^2$，其中 B 为 Batchsize. 全连接层输出数据个数计算公式为 $N_{fcd}=B\times d_{cn}^l$. 假设一个数据占用 4 个字节，则 n 个数据的数据量为 $4n$ 字节. 反向传播时第 l 层向第 $l-1$ 层需要传递残

表 4　AlexNet 网络神经元、参数数量和数据量

网络层次	卷积核大小	卷积核个数	神经元个数	特征图大小	参数个数/数据量	输出数据个数/数据量
卷积层 1	11×11	96	290400	55×55	34.9 K/0.13 MB	8957952/34.2 MB
卷积层 2	5×5	256	186624	27×27	307 K/1.1 MB	5537792/21.1 MB
卷积层 3	3×3	384	64896	13×13	885 K/3.4 MB	8306688/31.7 MB
卷积层 4	3×3	384	64896	13×13	664 K/2.5 MB	8306688/31.7 MB
卷积层 5	3×3	256	43264	13×13	442 K/1.7 MB	1179648/4.5 MB
全连接层 1	—	—	4096	—	37.8 M/144 MB	524288/2 MB
全连接层 2	—	—	4096	—	16.8 M/64 MB	524288/2 MB
全连接层 3	—	—	1000	—	4 M/15.6 MB	128000/0.05 MB
总计	—	1376	659272	—	61 M/232 MB	33465344/127.2 MB

差,其数据量等于前向传播时第 $l-1$ 层向第 l 层传输数据量除以 Batchsize. 从表 4 容易发现前向传播时卷积层输出数据量大,全连接层虽然参数数量多,但是输出数据量小. 模型并行是将网络结构均分到多个不同的设备上. 以在 3 块 GPU 上进行 AlexNet 的模型并行为例,将卷积层 1 的 96 个滤波器等分到 3 个 GPU 上,卷积层 2 的 256 个滤波器在 GPU0 划分 86 个滤波器,GPU1 和 GPU2 划分 85 个滤波器. 其它卷积层与全连接层的划分方法与此类似. Softmax 分类函数放置在 GPU0 上,如图 6 所示.

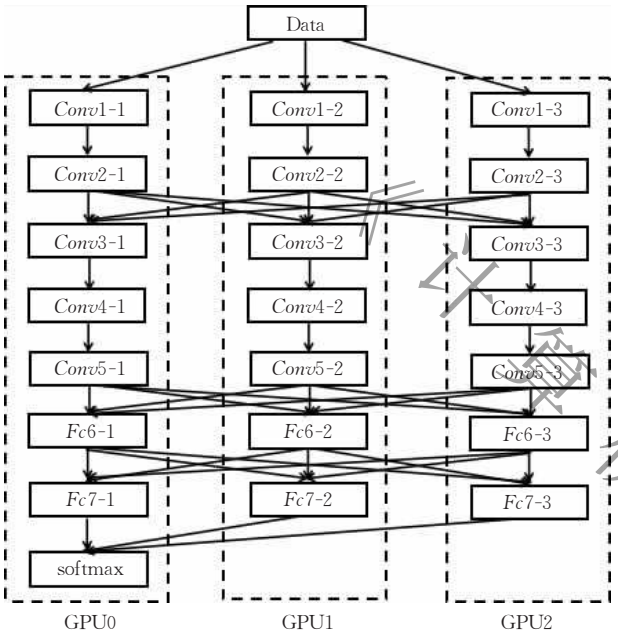


图 6 AlexNet 模型并行示意图

前向传播过程中,每个 GPU 需要将各自的卷积层第二层和第五层以及全连接层的输出结果发送到其它 GPU 上. 在 Batchsize 为 128 时,第二层卷积层到第三层卷积层通信量为 42.25 MB. 第五层卷积层与第一层全连接层的通信量为 9 MB. 第一层全连接层与第二层全连接层的通信量为 4 MB. 第二层全连接层与第三层全连接层的通信量为 1.33 MB. 在整个前向过程中,共需要进行四次通信,向下一层传输的数据为 56.58 MB,卷积层之间通信量占总通信量的 74.67%. 全连接层之间占 9.42%. 卷积层与全连接层之间通信量 9 MB,占 15.91%. 反向传播过程中各层输出的数据量远小于前向传播过程,因此可忽略. 在 40 Gb/s 的网络带宽下,一个 Batch 前向传播不考虑并行同步开销所需的通信时间约为十几毫秒,而且通信时间随 GPU 数量线性增加. 一个 Batch 前向、反向以及参数更新时所需的时间约为两三百毫秒. 因此,通信时间与计算时间比例还是比较高,这使得模型并行的可扩展性受限.

数据并行是在不同计算设备上用不同数据训练同一个模型. 以在三个计算节点上进行 AlexNet 数据并行为例,Batchsize 为 128,输入图像为 $227\times227\times3$. 由于每一个参数对应一个梯度更新值,所以梯度个数与参数个数相同. 因此,单个计算节点与参数服务器的通信量为 232 MB. 参数服务器在接收到 3 个计算节点的梯度时,进行参数更新后,将新的参数发送给各个计算节点. 参数服务器接收的数据量为 696 MB. AlexNet 进行一次参数更新时,共需要 1392 MB 的数据传输量. AlexNet 在 GPU 上计算时一个 Batch 前向、反向以及参数更新时所需的时间约为两三百毫秒,而在 40 Gb/s 的带宽下,进行一次 Batch 所需的通信时间约为 285 ms. 同步参数更新机制时的通信时间与计算时间比太高限制了并行深度神经网络的可扩展性. 因此,可采用半同步或异步的方式更新梯度以减少通信等待时间,还可以采用压缩权值的方法减少通信量.

5 深度神经网络开源软件系统并行化方法

由于深度神经网络在各个领域的广泛应用,工业界与学术界都推出了相关的开源软件系统,使研究者可以快速的将深度神经网络应用到自己的研究领域内. 目前,深度神经网络开源的软件系统主要有:Caffe、TensorFlow、MXNet、CNTK、Torch、Theano 等,具体如表 5 所示. 这些软件对深度神经网络并行时使用了一些相似的方法,在 CPU 上都使用了高性能多线程库来对网络进行训练,在 GPU 上都支持 cuDNN[66] 库来对网络进行加速. 在分布式并行深度神经网络的实现上,使用了数据并行或模型并行.

Caffe[67] 由加州大学伯克利视觉和学习中心开发,是一个清晰、可修改的、快速的深度神经网络框架. Caffe 支持硬件平台为 CPU 与 GPU,它使用 CPU 的高性能多线程库(Altas,OpenBLAS,Intel MKL)对网络进行训练;当使用 GPU 时,Caffe 使用 cuDNN 进行训练. 在并行实现上,采用数据并行的方式对深度神经网络进行加速,实现了单节点多设备的数据并行,通过树形拓扑连接策略进行多个 GPU 间的参数交换,有效地减少了通信开销.

TensorFlow[68] 是 Google 基于 DistBelief 研发的第二代人工智能学习系统,其灵活的架构使得用

户无需重写代码就可以将 TensorFlow 部署在不同的异构系统上. TensorFlow 使用计算图来表示机器学习算法. 在并行实现上，用户首先需要创建一个抽象的 TensorFlow 集群对象，它用来分布式地执行计算图. TensorFlow 支持模型和数据两种并行方式. 对于模型并行，首先定义计算图的结构、集群的配置以及将模型的不同部分在不同的计算设备上的映射. 每个抽象集群包括多个 task，每个 task 代表深度神经网络的一个计算任务，用户将 task 映射在不同的设备上. 对于数据并行，一个抽象集群可以划分为多个 job，一个 job 是一个特定的任务，job 一般分为 worker job 和 parameter job，一个 job 又包含多个 task. 在 TensorFlow 中，worker job 负责在每个节点上进行独立的训练一个深度神经网络，parameter job 负责实现类似于参数服务器的参数交换和梯度更新的功能.

MXNet[69]是一个高效灵活的多语言机器学习库，可被部署在从移动设备到分布式 GPU 集群的多种异构系统上. MXNet 通过 KVStore 实现参数更新与数据交换，通过引擎实现资源调度与任务管理. KVStore 是基于参数服务器的分布式的键值对的存储，用来在多个设备之间进行数据交换. 引擎是一个资源管理与任务调度器，用来调度执行 KVStore 的操作和管理参数更新机制. MXNet 采用两层架构的数据并行策略，第一层是在一个工作节点内的多个设备之间采用参数服务器进行数据并行，第二层是不同工作节点之间的数据并行. 深度神经网络训练时，首先将单个节点的多个设备上的参数进行汇总，随后单个节点将数据发送到参数服务器进行参数更新. 由于单个节点内的通信带宽远大于节点间的通信带宽，因此两级并行策略可以有效地减少对带宽的需求.

微软开源的 CNTK[70]是一个简单易用的深度学习工具包，在 CPU 和 GPU 上支持 Intel MKL 和 cuDNN 编程模型进行训练. CNTK 将神经网络通过有向图表示成一系列计算步骤，有向图的叶子节点表示输入值或网络参数，其他节点表示在其输入上的矩阵操作. CNTK 采用数据并行方法，实现了支持同步和异步参数更新机制的参数服务器. 同时，CNTK 可以部署在多 GPU 或 CPU 上的异构集群上用来进行深度神经网络的分布式训练. 它支持 4 种类型的分布式 SGD 算法：DataParallelSGD、BlockMomentumSGD、ModelAveragingSGD、DataParallelASGD. 同时可使用 1bitSGD 算法，该算法将梯度值进行量化，减少了传输的数据量.

Torch[71]得益于 Facebook 开源的 fbcunn 深度学习模块，是使用 LuaJIT 开发的简单高效的科学计算框架，支持多 GPU 和 CPU 训练. 在 CPU 上使用 OpenMP、Intel MKL 和 Pthread 进行多线程训练，在 GPU 上支持 cuDNN 和 OpenCL[72]. Torch 支持数据并行和模型并行两种策略，对于数据并行，在每个设备上独立的训练一个模型副本. 在进行参数更新时，把每个设备上的梯度传输到参数设备上进行参数更新，支持使用 1bitSGD 算法来减少传输的数据量. 模型并行是将训练数据复制到所有子模块，并在不同的设备上运行模型的不相交部分.

Theano[73]是一个基于 Python 的机器学习库，首次采用符号图来描述机器学习算法，支持 CPU 和 GPU 进行网络训练. 在 CPU 上使用 OpenMP 进行多线程训练. 在 GPU 上使用 cuDNN，采用数据并行的方法，支持多 GPU 的同步参数更新机制. 深度学习开源软件技术更新快，支持的硬件架构和软件平台也越来越多，特别是对嵌入式平台和 Python 语言的支持成为重点发展方向. 目前 TensorFlow、Caffe2、PyTorch，MXNet 都实现了利用计算图来表示深度神经网络. 表 5 对这几个开源深度学习软件进行了对比.

表 5　深度神经网络开源软件系统比较

开源软件	开源时间	支持硬件	并行接口	支持模型库	并行模式	多节点
Caffe	2013.12	CPU,GPU	CUDA,OpenCL	CNN,RNN,LSTM	数据并行	支持
TensorFlow	2015.11	CPU,GPU,MIC	CUDA	CNN,RNN,LSTM	数据并行 模型并行	支持
Torch	2017	CPU,GPU	CUDA	CNN,RNN,LSTM	数据并行	不支持
Theano	2007	CPU,GPU	CUDA,OpenCL	CNN,RNN,LSTM	数据并行	不支持
MXNet	2015.9	CPU,GPU	CUDA	CNN,RNN,LSTM	数据并行 模型并行	支持
CNTK	2016.1	CPU,GPU	CUDA	CNN,RNN, LSTM	数据并行	支持

6 深度神经网络并行化研究现状

深度神经网络的应用分为训练过程和推理过程.对训练过程进行并行设计时,在单节点上,利用多核和众核技术(GPU,MIC)进行并行加速;而对于单节点无法完成的大规模深度神经网络的训练过程,一般是结合 MPI 或 Spark 完成节点间的数据通信,通过分布式系统完成对整个模型的训练.而并行加速推理过程时,目前研究重点是通过专用加速器或 FPGA 进行加速,具有功耗低、速度快的优点.下面对近年来深度神经网络的并行化研究现状进行归纳总结:

(1) GPU

使用 GPU 来加速深度神经网络算法的训练过程,一般是使用 CUDA 或 OpenCL 将算法移植在 GPU 上,通过数据并行或模型并行,或者采用两者相结合的方法并行加速. Raina 等人[74]第一个提出利用 GPU 对 DBN 进行加速;腾讯的深度学习平台 Mariana[75]包括 DNN 的 GPU 数据并行框架,CNN 的 GPU 数据并行和模型并行框架,以及 DNN 的 CPU 集群框架,主要以 GPU 集群为主,每个节点配置 4 或 6 块 Tesla GPU,实现多 GPU 的并行加速. Omry 等人[76]在单个服务器上使用 4 个 NVIDIA GeForce GTX Titan GPU 实现了数据并行和模型并行两种方法的结合,最终在训练 ImageNet 的 1000 分类网络时相比于单个 GPU 达到了 2x 加速比. Li 等人[77]使用单块 Tesla K40c GPU 来加速 DBN,对于算法中的预训练阶段,将输入样本分批输入,同一批次的样本分配到不同线程并行计算,对于算法的微调过程,采用小批量的梯度下降算法,同一批次的不同样本在不同线程上并行计算.相比于 Intel Core i7-4790K 在预训练过程中获得了 14～22x 加速比,在微调过程中获得了 12～33x 加速比,而且最终的性能优于 CPU 上的 OpenBLAS 库和 GPU 上的 CUBLAS 库. Li 等人[78]研究了训练数据在 GPU 内部使用 NCHW 和 CHWN 等不同存储结构时对计算性能的影响,并设计了高效的数据存储模式.特别分析了卷积神经网络中不同算子的计算和内存访问特性,发现在某些参数下卷积不仅仅是计算受限的,也是内存受限的算子,池化层和 Softmax 函数都是内存受限的算子. Cui 等人[79]针对 Caffe 中数据并行加速算法时的通信瓶颈问题,提出了 GeePS C++库,用于管理基于 GPU 的深度学习应用程序的参数数据和本地数据,重叠计算和通信时间,最终在使用 8 块 Tesla K20 GPU 情况下相比于单 GPU,训练 AdamLike 网络获得了 6x 加速比,训练 GoogleNet 获得了 5x 加速比.在视频分类应用上,在使用 8 块 Tesla K20 GPU 情况下训练 RNN,相比于单 GPU 获得了 7x 加速比. Gu 等人[80]针对通过 OpenCL 扩展 Caffe 框架,使用多命令队列等方法进行性能优化,使用 AlexNet 在 AMD R9 Fury 上测试,测试结果表明与 NVIDIA GPU 相比,性能/成本相当. Bottleson 等人[81]针对 CNN 网络,也提出了 OpenCL 加速 Caffe 框架,通过在卷积层采用 GEMM、空域、频域三种不同的优化手段,使用 AlexNet 网络在 Intel 集成 GPU 上进行实验,相比于 Intel CPU,经过不同优化方法后获得 2.5～4.6x 的加速比.

(2) MIC

使用 MIC 对深度神经网络训练过程并行加速时,一般使用 OpenMP 并行编程语言,将算法中的循环部分通过 OpenMP 和向量化指令做多线程和 SIMD 并行.MIC 常用的有两种编程模式,一种是卸载模式:将一部分程序运行在 CPU 上,另一部分程序被卸载到 MIC 上并行执行,其中所需的数据也要通过 PCI-E 传递给 MIC;另一种是由 MIC 独立运行全部程序的原生模式. Viebke 等人[82]在 Intel Xeon Phi 7120P 上以原生模式训练 CNN,采用数据并行的方式,在 244 线程下相比于单线程达到了 103.5x 的加速比,相比于 Intel Xeon E5-2695v2 CPU 串行版本获得了 14.07x 的加速比. Liu 等人[83]在 Intel Xeon Phi 5110P 上以卸载模式运行 CNN,采用批量梯度下降法,将同一批次的图片分配给不同的线程进行训练,在 240 线程下相比于单线程达到了 131x 的加速比,相比于 Intel Xeon E5-2697 CPU 串行版本获得了 8.3x 的加速比. Olas 等人[84]在 Intel Xeon Phi 7120P 上以原生模式运行 DBN,将学习样本分组成包,并表示为矩阵,利用 Intel MKL 库中的矩阵乘法程序优化,最终在 240 线程下相比于 Intel Xeon E5-2695v2 CPU 串行版本获得了 13.6x 的加速比. Zlateski 等人[85]提出 3 维卷积算法,在 Intel CPU 上取得了近乎线性加速比,在众核处理器 5110P 上取得了 90x 的加速比.

(3) ASIC

设计专用加速器时,一般是通过分析算法特性,为其设计相应的硬件电路,充分利用算法中可并行的部分,为其中相应的操作分配具体的计算元件,从

而获得更高的加速效果.例如卷积层和全连接层中每个神经元的输出是由若干个权重和对应的输入值做的乘法累加操作,可以设计一定大小的处理单元阵列,每个处理单元负责运行乘法操作,通过将输入特征图与权重矩阵转换成数据流输入到每一个处理单元上完成各自的乘法操作便可以实现高度的并行化.因此专用加速器相比于 CPU、GPU 具有很高的加速比,例如 Chen 等人[86-87]针对 CNN 设计了专用加速器 DianNao,相比于 128-bit 2 GHz SIMD 处理器加速了约 117x.之后在 DianNao 的基础上,扩大芯片规模,设计了专用加速器 DaDianNao,相比于 NVIDIA K20M GPU 加速了 450.65x.此外,Han 等人[88-89]提出了深度压缩神经网络技术,将深度神经网络经过压缩,减少权值存储所需的空间,降低内存读取量,更加利于加速器的加速和降低能耗,并在此基础上设计出可加速压缩神经网络模型的专用加速器 EIE,相比于 Intel Core i7 5930k CPU 加速了 189x,相比于 NVIDIA GeForce GTX Titan X GPU 加速了 13x.目前已经投入使用的 Google 张量处理单元[90](Tensor Processing Unit,TPU)第一代 2016 年发布,功耗最大 40 W,用于谷歌街景和 AlphaGo 等应用场景.2017 年 5 月公开的第二代 CLOUD TPU 已经部署在 Google 云计算平台 Compute Engine 上,其提供了 65 536 个 8 位矩阵乘法单元,处理能力达 92 TOPS.

(4) FPGA

FPGA 开发时最常用的编程语言为硬件描述语言(HDL),如 Verilog 和 VHDL,但要使用这些语言编程需要有数字化设计和电路的专业知识,开发难度大,所以 FPGA 厂商开始支持更加抽象化的编程语言 OpenCL.程序员将算法中的可并行部分的任务进行划分,通过调用 OpenCL 的应用程序接口(API)将任务映射到 FPGA 的多个处理单元上,从而实现算法在 FPGA 上的并行加速.目前基于 OpenCL 实现的 FPGA 加速深度神经网络的工作有:Suda 等人[91]在 FPGA 上实现了 VGG 和 AlexNet 两种网络,卷积的权值采用 8-bit 精度,并且将卷积操作转为矩阵乘法然后设计了基于 OpenCL 的并行算法.在 P395-D8 平台上运行 VGG 网络计算性能为 117.8 GOPS,运行 AlexNet 计算性能为 72.4 GOPS,相比于 Intel i5-4590 3.3 GHz CPU 加速比分别为 5.5x 和 9.5x.Zhang 等人[92]在 FPGA 上实现了 VGG 网络,最终在 Altera Arria 10 上运行整个 VGG 网络计算性能达到了 1.79TOPS,相比于 Intel Xeon E5-1630V3 CPU 得到了 4.4x 的加速比.Aydonat 等人[93]在 FPGA 上实现了 AlexNet 网络,优化片外内存带宽要求,并引入 Winograd 转换方法减少了卷积操作中的乘法累加运算次数,最终在 Arria 10 上运行 AlexNet 网络达到了 1382 GFLOS 的峰值性能.使用 OpenCL 虽然提高了开发效率,但可能牺牲了更高的片上存储器利用率,因此执行效率依然存在很大的优化空间.

深度神经网络算法的网络层数和模型的复杂度逐年增加,设计高效的算法成为关键,使用低精度的神经网络权值和稀疏的网络连接成为趋势.只采用 1 和−1 作为权值的二值神经网络算法由于减少了存储空间和内存带宽的要求迅速发展起来,如 Courbariaux[94]提出的 BinaryConnect 网络,网络训练时系数是单精度,推理时系数从单精度概率抽样变为二值,从而获得加速,算法在门牌号码数据集(SVHN)上的预测准确率与单精度网络相当,Nurvitadhi 等人在 FPGA 上实施了只采用 1 和−1 作为权值的二值神经网络算法,实验结果表明 FPGA 与 CPU 和 GPU 相比取得了更好的每瓦特性能,ASIC 在性能和能效上比 CPU、GPU 和 FPGA 都高[95].二值网络尽管计算效率很高,但是算法的精度损失也大,使用 16 位和 8 位数据类型可以在计算效率和计算复杂度之间达到很好的平衡.Nurvitadhi 等人在 GPU 和 FPGA 平台上对比分析了稠密矩阵和稀疏矩阵在不同数据精度下的计算性能和能效,实验结果表明 FPGA 在低精度的算法上性能和能效都好于 GPU[96].

(5) 多节点并行化

早期的深度神经网络多节点并行化主要在 CPU 集群上计算,经典的工作包括 Google 的 Dean 等人在 CPU 集群上开发了 Google 的深度学习软件框架 DistBelief,支持数据并行和模型并行策略,数据并行时设计了适合大规模分布式训练的异步随机梯度下降法——Downpour SGD,将训练集划分为若干子集,并对每个子集运行一个单独的模型副本,各模型副本之间通过“参数服务器组”交换梯度信息,参数服务器组维护了模型参数的当前状态.异步随机梯度下降法异步性表现为(1)模型副本之间相互独立运行;(2)参数服务器组各节点之间相互独立.作者将 Downpour SGD 和 Adagrad 自适应学习率结合在一起,实验结果表明这种结合具有很好的效果.模型并行时,全连接网络使用 8 个计算节点时,获得 2.2 倍的加速,使用更多计算节点时,由于

网络开销导致加速比开始下降[55]. Song 等人在国产超级计算机太湖之光上利用数据并行策略训练 DBN 网络,在四个计算节点上训练 MNIST 数据集,取得的性能是 Intel 至强 E5-2420v22.2 G CPU 的 23 倍[97].

Wu 等人[98]在拥有 36 个服务器节点的超级计算机 Minwa 上,通过 CUDA 和 MPI 使用 32 块 Tesla K40m GPU 对深度卷积神经网络加速,采用模型并行和数据并行混合的方式,重叠通信和计算时间,最终相比于单个 GPU 获得了 24.7x 的加速比. Iandola 等人[99]在拥有 128 块 NVIDIA Tesla K20s GPU 的集群上,采用数据并行的方式加速深度神经网络,通过使用规约树算法提高了参数交换的效率和可扩展性,进一步通过增大同一批次的样本数量来减少通信总量,最终在训练 Network-in-Network网络时,相比于单 GPU 获得 39x 的加速比,训练 GoogleNet 相比于单 GPU 获得了 47x 的加速比. Awan 等人[100]利用协同设计的方法使用 12 个节点 80 个 Tesla K80 GPU 设计了分布式深度学习框架 S-Caffe. 采用数据并行的方式,通过对原有 Caffe 框架中的工作流程进行修改,将其与 MVAPICH2-GDR MPI 相结合,利用 CUDA 在 GPU 设备上加速计算任务,而 MPI 负责进程间通信,最大化重叠计算时间以及多级数据传输和梯度聚合的通讯时间. 实验结果表明在多节点 80 个 K80 GPU 训练时,该框架相比于 OpenMPI 实现了 133x 的加速比,相比于 MVAPICH2 实现了 2.6x 的加速比,具有良好的可扩展性.

百度搭建的 Paddle 是一个基于 Spark 的异构分布式深度学习系统,支持数据并行和模型并行,将数据分布到不同节点的 GPU 上,通过参数服务器协调各机器进行训练[101]. 当 Spark 与深度学习框架结合时,主要是基于数据并行的方法,在 Spark 每个节点中运行单个深度学习框架引擎,通过节点内的模型副本维护以及节点间的全局模型更新来达到深度神经网络训练的有效性和快速性. Moritz 等人[102]实现了 Spark 与 Caffe 的结合,并通过引入控制参数减少同步次数改进了传统的同步 SGD 并行化机制. 雅虎分别实现了 Spark 与 Caffe、TensorFlow 的结合,并修改了 Spark 执行器之间的通信方式来提升性能. 其中 TensorFlowOnSpark 框架支持模型并行和数据并行,同步和异步的训练,同时对深度学习框架有的程序具有很好的兼容性.

尽管用于深度学习的硬件架构和产品种类繁多,但是由于 CUDA 良好的软件生态系统,特别是 NVIDIA 持续推出支持深度学习新特性的新架构 GPU 硬件和深度学习加速库 cuDNN,目前开源的深度学习软件都支持 cuDNN,这就使得目前深度学习应用大部分都是运行在 NVIDIA GPU 上. 随着深度学习应用的领域越来越多,在应用时除了 GPU 外,综合考虑运行深度学习的设备在功耗、体积、价格和开发周期等因素,基于 FPGA 和 ASIC 的深度学习研究逐渐成为新的热点.

7 实验测试

由于 Google 公司的大力支持,TensorFlow 开源后用户数量迅速扩大. 本文以 TensorFlow 为开发平台,使用 MNIST 和 CIFAR10 数据集训练深度神经网络,在 CPU 和 GPU 计算平台上进行实验. 根据实验结果,分析了不同计算设备下的计算效率.

MNIST 数据集由手写数字图像组成,共有十个类别,灰度图像大小为 28×28,共有 60 000 个样例作为训练数据集,10 000 个样例作为测试集. CIFAR10 数据集共有十个类别,彩色图像大小为 32×32×3,共有 50 000 个样例作为训练数据集,10 000 个样例作为测试集.

本文对两种数据集分别采用两种深度的网络结构来训练,MNIST 数据集使用的网络深度为 4 层,包括 2 个卷积层和 2 个全连接层,在每个卷积层后面使用了局部响应归一化操作和 2×2 窗口大小的最大值池化. CIFAR10 数据集采用的网络深度为 5 层,包括 2 个卷积层和 3 个全连接层,在每个卷积层后面都使用了 3×3 窗口大小的最大值池化,并在第一个卷积层后面使用了 Dropout 操作. 两种网络具体参数分别如表 6 和表 7 所示.

表 6 MNIST 手写体数据集网络参数

层类型	特征图数	特征图大小	神经元个数	卷积核大小	参数个数
卷积 1	32	28×28	25088	5×5	832
卷积 2	64	14×14	12544	5×5	51264
全连接 1	—	512	512	—	1606144
全连接 2	—	10	10	—	5130

表 7 CIFAR10 数据集网络参数

层类型	特征图数	特征图大小	神经元个数	卷积核大小	参数个数
卷积 1	64	24×24	36864	5×5	4864
卷积 2	64	12×12	9216	5×5	102464
全连接 1	—	384	384	—	885120
全连接 2	—	192	192		73920
全连接 3	—	10	10	—	1930

本文实验在西安电子科技大学高性能计算中心完成，实验使用计算设备配置如表 8 所示. 测试了两种数据集和对应的深度神经网络分别在 CPU 串行、CPU 多线程模型并行以及单 GPU 下完成单批数据的训练时间，用于比较 CPU 多线程模型并行以及单 GPU 带来的加速效果，实验结果如表 9 所示. 实验中 CIFAR10 数据集使用小批量梯度下降算法，其中 Batchsize 参数赋值为 128，以 0.1 的标准差产生正态分布的随机数来初始化权重. 并使用 3 个全连接层参数的 L2 范数作为正则项，权重衰减为 0.0005，学习率为固定值 0.1，迭代次数固定为 10000 次. 对于 MNIST 数据集，使用小批量梯度下降算法来训练网络，其中 Batchsize 的大小为 64. 以 0.1 的标准差产生正态分布的随机数来初始化权重. 使用 2 个全连接层的参数的 L2 范数作为正则项，权重衰减为 0.0005. 并使用动量法更新参数，动量大小为 0.9. 使用指数衰减学习率，初始学习率为 0.01，每训练完 50000 个样本衰减一次，衰减因子为 0.95. 迭代次数固定为 5000 次. 通过测得训练单批样本所需的时间来比较性能. CIFAR10 经过 10000 次迭代精确率达到 80%，MNIST 经过 5000 次迭代精确率达到 99%. 由表 9 实验结果可以看出，MNIST 和 CIFAR10 数据集在 CPU 多线程模型并行下训练相比于 CPU 串行分别获得了 5.9 和 6.9 倍的加速比，可以得出 TensorFlow 在 24 核 CPU 多线程下模型并行的计算效率为 24.5% 和 28.7%，没有完全发挥出多核 CPU 的并行计算能力. TensorFlow 在 CPU 上的算子并行调度机制目前还没有公开，但实验表明进一步可以优化调度策略. 在单 GPU 加速下，MNIST 和 CIFAR10 数据集的训练时间相比于 CPU 串行分别获得了 27.9 和 31.0 倍的加速比，证明了 GPU 良好的加速能力.

表 8　实验运行环境

CPU	2×12-core Intel Xeon E5-2692
单 CPU 单精度计算能力	422.4 GFLOPS
GPU	2×NVIDIA Tesla K20
单 GPU 单精度计算能力	3.52 TFLOPS
RAM	64 GB DDR3 1600 MHz ECC
操作系统	Red Hat Enterprise Linux Server 6.4
深度学习软件	TensorFlow 1.1.0-rc2
CUDA 版本	8.0
cuDNN 版本	5.1
Python 版本	2.7.13

表 9　不同设备的单批数据训练时间（单位：ms）

	CPU 串行	CPU 多线程	单 GPU 加速
MNIST	496	83	16
CIFAR10	867	126	31

在 CPU 多线程模型并行中，TensorFlow 程序默认占用所有可以使用的内存资源和 CPU 资源. 在程序中可以通过设置函数 intra_op_parallelism_threads 中的 intra 参数来控制运算符内部的并行，当运算符为单一运算符并且内部可以实现并行时，如矩阵乘法，reduce_sum 之类的操作，可以通过设置 intra 参数来控制运算符内部并行计算的线程数. 同时可以设置函数 inter_op_parallelism_threads 中的 inter 参数控制多个运算符之间的并行计算，当有多个运算符之间无数据依赖，互相独立，TensorFlow 会并行地计算它们，使用 inter 参数来控制并行的线程数. 可以通过控制这两个参数来控制每个操作符使用并行计算的线程个数，以满足不同的并行应用场景. 系统默认设置两个参数都为 0，即占用最大 CPU 资源.

本文测试了 intra、inter 两个参数对 CPU 多线程训练两种数据集计算效率的影响，结果如表 10 所示. 由表 10 可以看出，参数为默认值时，CPU 运行效率最优. Inter 参数在设为 2 时相比于设为 1 时有明显的性能提升，但再增加该参数值，时间不会降低. Intra 参数从 1～24，运行时间都有明显的下降，且下降幅度逐渐降低. 并且随着两个参数大小的增加，CPU 负载逐渐升高，当 inter 为 2，intra 为 24 时训练时间接近于默认情况，且负载也接近于默认情况下的负载. 对比得到 intra 参数带来的影响大于 inter 参数，分析原因是因为卷积神经网络中的多数运算符可在内部并行，如卷积以及全连接层的矩阵乘法操作，而不同运算符之间有数据依赖，如池化操作依赖于与其相邻的卷积操作，全连接层的矩阵乘法操作依赖于上一层的输出结果，因此不利于操作符之间的并行. 总之，TensorFlow 在 CPU 上的模型并行效率还有很大的提升空间.

表 10　两种参数不同数值下的 CPU 多线程性能

inter	intra	CIFAR10		MNIST	
		单批次时间/ms	CPU 负载/%	单批次时间/ms	CPU 负载/%
0	0	126	1482	83	1538
1	1	867	100	496	100
2	1	743	125	388	143
3	1	760	125	396	144
4	1	755	125	400	144
2	2	563	208	356	202
2	4	331	382	203	165
2	8	215	668	133	625
2	16	157	1137	96	1116
2	24	130	1458	81	1540

8 深度神经网络并行化的挑战和展望

深度神经网络带来了机器学习的一个新浪潮，受到了从学术界到工业界的广泛重视.然而由于深度神经网络算法流程复杂、迭代次数多、计算复杂度高的特点使得深度神经网络在并行化时存在一些挑战和瓶颈，下面我们对深度神经网络未来的并行化发展提出几点值得探索的方向：

(1)基于 OpenCL 的并行深度神经网络算法的性能可移植性

由于开发语言的多样性，使得针对一种异构计算硬件开发的并行深度神经网络算法在另外一种并行计算硬件上运行时必须投入大量人力资源进行代码重写和性能优化.目前的解决方法之一是采用跨平台的编程语言 OpenCL，虽然 OpenCL 能够运行在这些不同的异构硬件上，实现了代码的可移植性.然而由于这些异构计算硬件内部结构的巨大差异，使得 OpenCL 撰写的统一代码还不能实现性能的可移植性，在一些异构计算硬件上运行时算法的性能距离硬件的理论计算峰值差异比较大.因此，基于 OpenCL 的并行深度神经网络的性能可移植性特别是异构并行计算程序的自调优技术亟需研究者解决[103-104].

(2)深度神经网络模型并行中任务的自动划分

已有的研究成果表明深度神经网络模型并行化主要针对已设计好的神经网络结构采用手工划分网络并将其映射到不同的计算设备上，手工划分网络由于对任务负载的运行时间估计不够精准容易导致计算节点上的负载不均衡[29,65,105].要想实现网络模型的自动划分并且达到负载均衡，还面临着如何构建精准的深度神经网络算子性能模型以及设计任务调度算法的难题.

深度神经网络目前主要采用计算图[106-107]来表示，为了将计算图中的计算任务映射到多个并行计算的硬件系统上，首先需要对计算图中的深度神经网络算子构建性能模型，考虑硬件体系结构对程序运行时间的影响，分析神经网络算子的执行过程，获得影响其程序性能的因素，最终构建深度神经网络中各个算子的性能模型.通过对性能模型求解，得出每个算子在并行计算设备上的执行时间.然后结合深度神经网络的结构特点，设计合理有效的任务调度算法，通过调度算法得到深度神经网络的模型划分策略，根据得到的模型划分策略将计算图中的计算任务映射到并行计算的硬件系统上.

(3)深度神经网络数据并行面临挑战

对于深度神经网络数据并行未来发展的趋势，可从两个方向出发：第一是从算法角度，设计收敛速度快通信代价低的分布式随机梯度下降算法；第二是解决集群中不同节点间的通信瓶颈问题.

① 多节点之间参数更新方式的选择

目前对分布式随机梯度下降算法中参数更新的方式主要有同步和异步两种机制，同步 SGD 由于较慢节点所带来的同步等待使得数据并行时的加速比并不理想，异步 SGD 收敛精度不够且在训练后期可能出现震荡现象.Xing 等人提出的针对机器学习领域迭代算法的 SSP[59-61]虽然可以解决上述问题，但是对于参数规模越来越庞大的深度神经网络来说，仍无法满足要求，需进一步优化.因此，需要设计出一种新的参数更新机制，在保证算法精度和收敛速度的前提下，减少同步等待所带来的开销.

② 异构计算节点间通信瓶颈

在处理海量数据和复杂模型时，通过高速网络将多台异构计算机互联起来组成异构计算集群，目前互联的网络有千兆以太网、万兆以太网和 Infiniband 网络等.集群的网络带宽如 Infiniband 现在最高可达 100 Gb/s，但是与异构计算节点内部的内存交换速度相比，还是相差很大.现有的深度神经网络框架如 TensorFlow 等在解决节点间同步问题时通常使用基于参数服务器的通信方案，该方案中通信开销将随着系统中节点数的增加而增大，从而影响系统的并行计算性能.因此在设计基于异构集群的并行深度神经网络算法时必须考虑节点之间的网络带宽带来的性能影响，目前一种典型的优化方法就是发送节点先对梯度进行压缩，然后通过网络传输，接收节点对收到的梯度进行解压缩，最后利用梯度更新权值，无损压缩由于没有改变梯度值因此不影响深度神经网络算法的收敛性，但是由于压缩比受限，其带来的并行计算性能提升也相对有限.有损压缩成为目前减少网络通信代价的热点方向，但是由于有损压缩改变了梯度值，因此其会影响算法的收敛性，减少通信代价和保证算法收敛性之间的平衡成为关键[108-109].

(4)基于新形态计算机的深度神经网络加速研究

由于摩尔定律的放缓和应用的日趋复杂，传统的冯·诺伊曼体系结构在应用于深度神经网络等人工智能应用时存在很大的局限性，因此研究新型计算机越来越迫切[110-111].基于新型材料的 ReRAM

(Resistive Random Access Memory)被认为是今后替代当前 DRAM,作为密度更大、功耗更小的下一代存储的技术之一. Chi 等人[112]充分利用了 ReRAM 既能作为存储器件,又能进行模拟计算的特性,将其用于加速神经网络计算中的矩阵乘法,在卷积网络、多层感知器和 VGG-D 网络上实验表明了其加速和节能的有效性,特别是由于其采用了内存处理(Processing-In-Memory,PIM)架构,其访问数据的时间可忽略不计.

9 总　结

深度神经网络给人工智能的发展带来了希望,但是随着训练数据集的增大和网络规模的日趋复杂,导致深度神经网络的训练时间越来越长. 因此,深度神经网络的并行化是加速人工智能发展的重要基础. 本文通过对当前深度神经网络并行化技术进行归纳总结,以及对模型并行、数据并行原理的阐述和常用开源软件系统中并行化方法进行了分析,在此基础上,列出了深度神经网络并行化存在的挑战并对未来的发展趋势进行了展望. 可以预见,随着异构计算平台的快速发展以及并行化技术难题的不断解决,深度神经网络应用的时效性问题会得到不断的突破,并将成功应用到更多的领域中.

致　谢　感谢《计算机学报》编辑部和审稿人的宝贵意见!

参　考　文　献

[1] Zeng Yi, Liu Cheng-Lin, Tan Tie-Niu. Retrospect and outlook of brain-inspired intelligence research. Chinese Journal of Computers, 2016, 39(1): 212-222(in Chinese)
(曾毅,刘成林,谭铁牛. 类脑智能研究的回顾与展望. 计算机学报, 2016, 39(1): 212-222)

[2] Huang Tie-Jun, Shi Lu-Ping, Tang Hua-Jin, et al. Research on multimedia technology 2015—Advances and trend of brain-like computing. Journal of Image and Graphics, 2016, 21(11): 1411-1424(in Chinese)
(黄铁军, 施路平, 唐华锦等. 多媒体技术研究: 2015——类脑计算的研究进展与发展趋势. 中国图象图形学报, 2016, 21(11): 1411-1424)

[3] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks. Science, 2006, 313(5786): 504-507

[4] Kang L, Ye P, Li Y, Doermann D. Convolutional neural networks for no-reference image quality assessment// Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition. Columbus, USA, 2014: 1733-1740

[5] Jiao Li-Cheng, Zhao Jin, Yang Shu-Yuan, et al. Deep Learning, Optimization and Recognition. Beijing: Tsinghua University Press, 2017(in Chinese)
(焦李成, 赵进, 杨淑媛等. 深度学习、优化与识别. 北京:清华大学出版社,2017)

[6] Ren S Q, He K M, Girshick R, et al. Faster R-CNN: Towards real-time object detection with region proposal networks// Proceedings of the Neural Information and Processing System. Montreal, Canada, 2015: 91-99

[7] Chen L C, Papandreou G, Kokkinos I, et al. Semantic image segmentation with deep convolutional nets and fully connected CRFs. arXiv:1412.7062, 2014

[8] Schroff F, Kalenichenko D, Philbin J. FaceNet: A unified embedding for face recognition and clustering//Proceedings of the IEEE Conference onComputer Vision and Pattern Recognition. Boston, USA, 2015: 815-823

[9] Chen X Z, Kundu K, Zhang Z, et al. Monocular 3D object detection for autonomous driving//Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA, 2016: 2147-2156

[10] Amini S M, Smith J, Maciejewski A A. Stochastic-based robust dynamic resource allocation for independent tasks in a heterogeneous computing system. Journal of Parallel and Distributed Computing, 2016, 97: 96-111

[11] Viñas M, Fraguela B B, Andrade D. High productivity multi-device exploitation with the heterogeneous programming Library. Journal of Parallel and Distributed Computing, 2016, 101: 51-68

[12] Emmerson M D, Damper R I. Determining and improving the fault tolerance of multilayer perceptrons in a pattern-recognition application. IEEE Transactions on Neural Networks, 1993, 4(5): 788-793

[13] Nair V, Hinton G E, Farabet C. Rectified linear units improve restricted Boltzmann machines//Proceedings of the 27th International Conference on Machine Learning. Haifa, Israel, 2010: 807-814

[14] Park J, Sandberg I. Universal approximation using radial-basis-function networks. Neural Computation, 1991, 3(2): 246-257

[15] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors. Nature, 1986, 323: 533-536

[16] Jiao Li-Cheng, Yang Shu-Yuan, Liu Fang, et al. Seventy years beyond neural networks: Retrospect and prospect. Chinese Journal of Computers, 2016, 39(8): 1697-1716(in Chinese)
(焦李成, 杨淑媛, 刘芳等. 神经网络七十年: 回顾与展望. 计算机学报, 2016, 39(8): 1697-1716)

[17] Rifai S, Vincebt P, Muller X, et al. Contractive auto-encoders: Explicit invariance during feature extraction//Proceedings of the 28th International Conference on Machine Learning. Bellevue, USA, 2011: 833-840

[18] Hinton G E, Krizhevsky A, Wang S. Transforming auto-encoders// Proceedings of the International Conference on Artificial Neural Networks. Espoo, Finland, 2011: 44-51

[19] Chen M, Xu Z, Winberger K Q, Sha F. Marginalized denoising auto-encoders for domain adaptation//Proceedings of the International Conference on Machine Learning. Edinburgh, UK, 2012: 767-774

[20] Lee H, Grosse R, Ranganath R, Ng A Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations//Proceedings of the International Conference on Machine Learning. Montreal, Canada, 2009: 609-616

[21] Lee H, Pham P, Largman Y, Ng A Y. Unsupervised feature learning for audio classification using convolutional deep belief networks//Proceedings of the Neural Information and Processing System. Vancouver, Canada, 2009: 1096-1104

[22] Ranzato M, Boureau Y, LeCun Y. Sparse feature learning for deep belief networks//Proceedings of the Neural Information and Processing System. Vancouver, Canada, 2007: 1185-1192

[23] Jain V, Seung S H. Natural image denoising with convolutional networks//Proceedings of the Neural Information and Processing System. Vancouver, Canada, 2008: 769-776

[24] Le Q, Ngiam J, Chen Z H, et al. Tiled convolutional neural networks//Proceedings of the Neural Information and Processing System. Vancouver, Canada, 2010: 1279-1287

[25] Taylor G, Fergus R, LeCun Y, Bregler C. Convolutional learning of spatio-temporal features//Proceedings of the European Conference on Computer Vision. Heraklion, Greece, 2010: 140-153

[26] Vincent P, Larochelle H, Lajoie I, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 2010, 11: 3371-3408

[27] Chen L J, Qu H, Zhao J H, et al. Efficient and robust deep learning with Correntropy-induced loss function. Journal of Neural Computing and Applications, 2016, 27(4): 1019-1031

[28] Le Q V, Ngiam J, Coates A, et al. On optimization methods for deep learning//Proceedings of the 28th International Conference on Machine Learning, ICML 2011. Bellevue, USA, 2011: 265-272

[29] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks//Proceedings of the Neural Information and Processing Systems. Lake Tahoe, USA, 2012: 1097-1105

[30] Szegedy C, Liu W, Jia Y Q, et al. Going deeper with convolutions//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, USA, 2015: 1-9

[31] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition//Proceedings of the International Conference on Learning Representations. San Diego, USA, arXiv: 1409.1556v6, 2014

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

[33] Goodfellow I J, Pouget A, Mirza M, et al. Generative adversarial nets//Proceedings of the Neural Information and Processing System. Montreal, Canada, 2014: 2672-2680

[34] Shi S, Wang Q, Xu P, et al. Benchmarking state-of-the-art deep learning software tools. arXiv: 1608.07249v7, 2016

[35] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436-444

[36] Matej M, Martin S, Neil B. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. Science, 2017, 356(6337): 508-513

[37] Zhang Chang-Shui. Challenges for machine Learning. Scientia Sinica, 2013, 43(12): 1612-1623(in Chinese)
(张长水. 机器学习面临的挑战. 中国科学：信息科学, 2013, 43(12): 1612-1623)

[38] Zhou Fei-Yan, Jin Lin-Peng, Dong Jun. Review of convolutional neural network. Chinese Journal of Computers, 2017, 40(1): 1-23(in Chinese)
(周飞燕，金林鹏，董军. 卷积神经网络研究综述. 计算机学报, 2017, 40(1): 1-23)

[39] Yu Kai, Jia Lei, Chen Yu-Qiang. Deep learning: Yesterday, today and tomorrow. Journal of Computer Research and Development, 2013, 50(9): 1799-1804(in Chinese)
(余凯，贾磊，陈雨强. 深度学习的昨天、今天和明天. 计算机研究与发展, 2013, 50(9): 1799-1804)

[40] Jiao Li-Cheng, Zhao Jin, Yang Shu-Yuan, et al. Research advances on sparse cognitive learning computing and recognition. Chinese Journal of Computers, 2016, 39(4): 835-852 (in Chinese)
(焦李成，赵进，杨淑媛等. 稀疏认知学习、计算与识别的研究进展. 计算机学报, 2016, 39(4): 835-852)

[41] Chen Y S, Lin Z H, Zhao X, et al. Deep learning-based classification of hyperspectral data. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2014, 7(6): 2094-2107

[42] Zhao X Y, Li X, Zhang Z F. Multimedia retrieval via deep learning to rank. IEEE Signal Processing Letters, 2015, 22(9): 1487-1491

[43] Huang W H, Song G J, Hong H K, Xie K Q. Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. IEEE Transactions on Intelligent Transportation Systems, 2014, 15(5): 2191-2201

[44] Hou W L, Gao X B, Tao D C, Li X L. Blind image quality assessment via deep learning. IEEE Transactions on Neural Networks and Learning Systems, 2015, 26(6): 1275-1286

[45] Silver D, Huang A, Maddison C J. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529(7587): 484-489

[46] Hong S, Oguntebi T, Olukotun K. Efficient parallel graph exploration on Multi-core CPU and GPU//Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques. Washington, USA, 2011: 78-88

[47] Nickolls J, Dally W J. The GPU computing era. IEEE Micro, 2010, 30(2): 56-69

[48] Sodani A, Gramunt R, Corbal J, et al. Knights Landing: Second-generation Intel Xeon Phi product. IEEE Micro, 2016, 36(2): 34-46

[49] Ahmad S, Boppana V, Ganusov I, et al. A 16-nm multiprocessing system-on-chip field-programming gate array platform. IEEE Micro, 2016, 36(2): 48-62

[50] Garland M, Grand S L, Nickolls J, et al. Parallel computing experiences with CUDA. IEEE Micro, 2008, 28(4): 13-27

[51] Javier Diaz, Camelia Munoz-Caro, Alfonso Nino. A survey of parallel programming models and tools in the multi and many-core era. IEEE Transactions on Parallel and Distributed System, 2012, 23(8): 1369-1386

[52] Lwainsky C, Shudler S, Calotoiu A, et al. How many threads will be too many? On the scalability of OpenMP implementations//Proceedings of the 21st European Conference on Parallel Processing. Vienna, Austria, 2015: 451-463

[53] Dinan J, Balaji P, Buntinas D, et al. An implementation and evaluation of the MPI 3.0 one-sided communication interface. Concurrency and Computation: Practice and Experience, 2016, 28(17): 4385-4404

[54] Zaharia M, Xin R S, Wendell P, et al. Apache Spark: A unified engine for big data processing. Communications of the ACM, 2016, 59(11): 56-65

[55] Dean J, Corrado G S, Monga R, et al. Large scale distributed deep networks//Proceedings of the Neural Information and Processing System. Lake Tahoe, USA, 2012: 1223-1231

[56] Cui H, Cipar J, Ho Q, et al. Exploiting bounded staleness to speed up big data analytics//Proceedings of the Usenix Conference on Usenix Technical Conference. Philadelphia, USA, 2014: 37-48

[57] Li M, Andersen D G, Park J W. Scaling distributed machine learning with the parameter server//Proceedings of the International Conference on Big Data Science and Computing. Beijing, China, 2014: 583-598

[58] Krizhevsky A. One weird trick for parallelizing convolutional neural networks. arXiv: 1404.5997v2, 2014

[59] Cipar J, Ho Q, Kim J K, et al. Solving the straggler problem with bounded staleness//Proceedings of the Usenix Conference on Hot Topics in Operating Systems. New Mexico, USA, 2013: 22-22

[60] Ho Q, Cipar J, Cui H, et al. More effective distributed ML via a stale synchronous parallel parameter server//Proceedings of the Neural Information and Processing System. Lake Tahoe, USA, 2013: 1223-1231

[61] Xing E P, Ho Q, Xie P. Strategies and principles of distributed machine learning on big data. Engineering Press, 2016, 2(2): 179-195

[62] Goyal P, Piotr D, Girshick R, et al. Accurate, large minibatch SGD: Training ImageNet in 1 Hour. arXiv: 1706.02677v1, 2017

[63] Sergey Ioffe, Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift//Proceedings of the 32nd International Conference on Machine Learning. Lille, France, 2015: 448-456

[64] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe. Rethinking the inception architecture for computer vision//Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Nevada, USA, 2016: 2818-2826

[65] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arxiv: 1602.07261, 2016

[66] Chetlur S, Woolley C, Vandermersch P, et al. cuDNN: Efficient primitives for deep learning. arXiv: 1410.0759v3, 2014

[67] Jia Y Q, Shelhamer E, et al. Caffe: Convolutional architecture for fast feature embedding//Proceedings of the 22nd ACM International Conference on Multimedia. Orlando, USA, 2014: 675-678

[68] Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv: 1603.04467v1, 2016

[69] Chen T Q, Li M, Li Y, et al. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv: 1512.01274v1, 2015

[70] Seide, Frank, Agarwal, Amit. CNTK: Microsoft's open-source deep-learning toolkit//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, USA, 2016: 2135-2135

[71] Collobert R, Bengio S, Marithoz J. Torch: A modular machine learning software library. Swiss: Idiap Research Institute, Research Report: IDIAP-RR 02-46, 2002

[72] Perkins H. Cltorch: A hardware-agnostic backend for the Torch deep neural network library, based on OpenCL. arXiv: 1606.04884v1, 2016

[73] Team T, Alrfou R, Alain G, et al. Theano: A python framework for fast computation of mathematical expressions. arXiv: 1605.02688v1, 2016

[74] Raina R, Madhavan A, Ng A Y. Large-scale deep unsupervised learning using graphics processors//Proceedings of the International Conference on Machine Learning. Montreal, Canada, 2009: 873-880

[75] Zou Y, Jin X, Li Y. Mariana: Tencent deep learning platform and its applications. Proceedings of the VLDB Endowment. Seoul, Korea, 2014, 7(13): 1772-1777

[76] Yadan O, Adams K, Taigman Y. Multi-GPU Training of ConvNets. arXiv: 1312.5853v4, 2013

[77] Li T, Dou Y, Jiang J, et al. Optimized deep belief networks on CUDA GPUs//Proceedings of the International Joint Conference on Neural Networks. Killarney, Ireland, 2015: 1-8

[78] Li C, Yang Y, Feng M, et al. Optimizing memory efficiency for deep convolutional neural networks on GPUs//Proceedings of the 2016 International Conference for High Performance Computing, Networking, Storage and Analysis. Salt Lake City, USA, 2017: 633-644

[79] Cui H, Zhang H, Ganger G R, et al. GeePS: Scalable deep learning on distributed GPUs with a GPU-specialized parameter server//Proceedings of the 11th European Conference on Computer Systems. London, UK, 2016: 1-16

[80] Gu J, Liu Y, Gao Y, et al. OpenCL caffe: Accelerating and enabling a cross platform machine learning framework//Proceedings of the 4th International Workshop on OpenCL. Vienna, Austria, 2016: 1-5

[81] Bottleson J, Kim S Y, Andrews J, et al. clCaffe: OpenCL accelerated Caffe for convolutional neural networks//Proceedings of the 2016 IEEE 30th International Parallel and Distributed Processing Symposium Workshops. Chicago, USA, 2016: 50-57

[82] Viebke A, Memeti S, Pllana S, Abraham A. CHAOS: A parallelization scheme for training convolutional neural networks on Intel Xeon Phi. Journal of Supercomputing, 2017, 5: 1-31

[83] Liu J, Wang H, Wang D, et al. Parallelizing convolutional neural networks on Intel many integrated core architecture//Proceedings of the 28th International Conference on Architecture of Computing Systems. Porto, Portugal, 2015: 71-82

[84] Olas T, Mleczko W K, Nowicki R K, et al. Adaptation of deep belief networks to modern multicore architectures//Proceedings of the 11th International Conference on Parallel Processing and Applied Mathematics. Krakow, Poland, 2016: 459-472

[85] Zlateski A, Lee K, Seung H S. Scalable training of 3D convolutional networks on multi- and many-cores. Journal of Parallel and Distributed Computing, 2017, 106: 195-204

[86] Chen T, Du Z, Sun N, et al. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning//Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems. Salt Lake City, USA, 2014: 269-284

[87] Chen Y, Luo T, Liu S, et al. DaDianNao: A machine-learning supercomputer//Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture. Cambridge, UK, 2015: 609-622

[88] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv:1510.00149, 2016

[89] Han S, Liu X, Mao H, et al. EIE: efficient inference engine on compressed deep neural network//Proceedings of the ACM/IEEE International Symposium on Computer Architecture. Seoul, South Korea, 2016: 243-254

[90] Jouppi N, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processor. arXiv: 1704.04760v1, 2017

[91] 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. California, USA, 2016: 16-25

[92] Zhang J, Li J. Improving the performance of OpenCL-based FPGA accelerator for convolutional neural network//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. California, USA, 2017: 25-34

[93] Aydonat U, O'Connell S, Capalija D, et al. An OpenCL™ deep learning accelerator on arria 10//Proceedings of the 2017 ACM/SIGDA International Symposium on Field—Programmable Gate Arrays. California, USA, 2017: 55-64

[94] Courbariaux M, Bengio Y, David J P. Binary Connect: Training deep neural networks with binary weights during propagations//Proceedings of the 29th Annual Conference on Neural Information and Processing System. Montreal, Canada, 2015: 3123-3131

[95] Nurvitadhi E, Sheffield D, Sim J, et al. Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC//Proceedings of the International Conference on Field—Programmable Technology. Xi'an, China, 2016: 77-84

[96] Nurvitadhi E, Venkatesh G, Sim J, et al. Can FPGAs beat GPUs in accelerating next-generation deep neural networks//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2017: 5-14

[97] Song K, Liu Y, Wang R, et al. Restricted Boltzmann machines and deep belief networks on Sunway cluster//Proceedings of the 18th IEEE International Conference on High Performance Computing and Communications. Sydney, Australia, 2016: 245-252

[98] Wu R, Yan S, Shan Y, et al. Deep image: Scaling up image recognition. arXiv: 1501.02876v5, 2015

[99] Iandola F N, Moskewicz M W, Ashraf K, et al. FireCaffe: Near-linear acceleration of deep neural network training on compute clusters//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA, 2016, 37: 2592-2600

[100] Awan A A, Hamidouche K, Hashmi J M, Panda D K. S-Caffe: Co-designing MPI runtimes and Caffe for scalable deep learning on modern GPU clusters//Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Austin, USA, 2017: 193-205

[101] Yu K. Large-scale deep learning at Baidu//Proceedings of the 22nd ACM International Conference on Information and Knowledge Management. San Francisco, USA, 2013: 2211-2212

[102] Moritz P, Nishihara R, Stoica I, et al. SparkNet: Training deep networks in Spark. arXiv: 1511.06051v4, 2015

[103] Falch T L, Elster A C. Machine learning-based auto-tuning for enhanced performance portability of OpenCL applications. Concurrency & Computation Practice & Experience, 2017, 29(8): 1-20

[104] Tsai Y M, Luszczek P, Kurzak J, Dongarra J. Performance-Portable autotuning of OpenCL kernels for convolutional layers of deep neural networks//Proceedings of the 2016 Machine Learning in HPC Environments. Salt Lake City, USA, 2017: 9-18

[105] Mirhoseini A, Pham H, Le Q V, et al. Device placement optimization with reinforcement learning//Proceedings of the International Conference on Machine Learning. Sydney, Australia. arXiv: 1706.04972v2, 2017

[106] Looks M, Herreshoff M, et al. Deep learning with dynamic computation graphs//Proceedings of the International Conference on Learning Representations. Palais des Congrès Neptune, Toulon, France. arXiv: 1702.02181v2, 2017

[107] Neubig G, Dyer C, Goldberg Y, et al. DyNet: The dynamic neural network toolkit. arXiv: 1701.03980, 2017

[108] Alistarh D, Grubic D, Li J, et al. QSGD: Communication-optimal stochastic gradient descent, with applications to training neural networks. arXiv: 1610.02132, 2016

[109] Seide F, Fu H, Droppo J, et al. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs//Proceedings of the 15th Annual Conference of the International Speech Communication Association. Singapore, 2014: 1058-1062

[110] Jiao Li-Cheng, Li Yang-Yang, Liu Fang, et al. Quantum Computation, Optimization and Learning. Beijing: Science Press, 2017(in Chinese)
(焦李成，李阳阳，刘芳等. 量子计算、优化与学习. 北京：科学出版社，2017)

[111] Potok T E, Schuman C, Young S R, et al. A study of complex deep learning networks on high performance, neuromorphic, and quantum computers. arXiv: 1703.05364, 2017

[112] Chi P, Li S, Xu C, et al. PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory//Proceedings of the 43rd International Symposium on Computer Architecture. Seoul, Korea, 2016: 27-39

ZHU Hu-Ming, born in 1978, Ph.D., associate professor. His research interests include high performance computing and massively parallel machine learning.

LI Pei, born in 1992, M. S. candidate. Her research interest is high performance computing.

JIAO Li-Cheng, born in 1959, Ph. D., professor, Ph. D. supervisor. His research interests include intelligent perception and image understanding.

YANG Shu-Yuan, born in 1978, Ph. D., professor, Ph. D. supervisor. Her research interests include intelligent signal and image processing, machine learning.

HOU Biao, born in 1974, Ph. D., professor, Ph. D. supervisor. His research interest is SAR image processing.

Background

In the last few years, deep neural network has led a great performance on a variety of problems, such as automatic driving, image analysis and speech recognition. However, with the increasing accuracy requirements and complexity for the practical applications, the size of the neural networks becomes explosively large scale, training large deep model is computationally expensive and time-consuming. Therefore, it's necessary to accelerate large deep learning model in parallel. Recently, heterogeneous computing platforms, such as GPU, MIC and FPGA, are widely used in different research fields and provide the hardware foundation for the parallelization of the deep neural network. The parallel programming framework which include CUDA, OpenCL, OpenMP, MPI, Spark and so on is a bridge between heterogeneous computing platforms and deep neural network. Therefore, it becomes more and more important to efficiently accelerate the deep neural network using heterogeneous system architectures.

This work is partially supported by the National Science Foundation of China (61303032), the National Basic Research Program of China (2013CB329402), the Major National Scientific Research Projects (91438201 and 91438103), the Program for Changjiang Scholars and Innovative Research Team in University (IRT_15R53). Our research team has been working on high performance computing and massively parallel machine learning for many years. Related works have been published in international journals and conferences, such as IJHPCA, CCPE, IGARSS, etc.

In this paper, we give a brief review of the development of parallel and distributed deep learning in the past. We mainly focus on introducing the parallel hardware architecture, software system, model parallelism and data parallelism for deep neural network. Followed by this, we summarize the state of the art of parallel deep neural network research work. We also discuss our thoughts and analysis on the future research directions.