深度学习加速技术研究①

杨旭瑜[1,2,3], 张 铮[4], 张为华[1,2,3]

[1](复旦大学 软件学院, 上海 201203)
[2](复旦大学 上海市数据科学重点实验室, 上海 201203)
[3](复旦大学 并行处理研究所, 上海 201203)
[4](解放军信息工程大学 数学工程与先进计算国家重点实验室, 郑州 450001)

摘 要: 深度学习是近年来机器学习的研究热点, 并已广泛应用于不同领域. 但由于训练模型复杂和训练集规模庞大等原因导致的深度学习性能问题已成为其发展的一大阻碍. 近年来计算机硬件的快速发展, 尤其是处理器核数的不断增加和整体运算能力的快速提高, 给深度学习加速提供了硬件基础, 然而其训练算法并行度低和内存开销巨大等问题使得加速研究工作困难重重. 首先介绍了深度学习的背景和训练算法, 对当前主要的深度学习加速研究工作进行归纳总结. 在此基础上, 对经典的深度学习模型进行性能测试, 分析了深度学习及并行算法的性能问题. 最后, 对深度学习的未来发展进行了展望.

关键词: 深度学习; 深度学习加速; 深度学习性能; 分布式学习; 硬件加速

Research on Deep Learning Acceleration Technique

YANG Xu-Yu[1,2,3], ZHANG Zheng[4], ZHANG Wei-Hua[1,2,3]

[1](Software School, Fudan University, Shanghai 201203, China)
[2](Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)
[3](Parallel Processing Institute, Fudan University, Shanghai 201203, China)
[4](State Key Laboratory of Mathematical Engineering and Advanced Computing, PLA Information Engineering University, Zhengzhou 450001, China)

Abstract: Deep learning has become a popular research direction recently and has been applied to many fields. But the performance issue caused by the complicated learning model and massive training data has becomes an obstacle of deep learning evolution. As the development of processor technology, the core number and performance of the processor have increased rapidly. However, the acceleration research is restricted mainly due to the low parallelism and high memory consumption of the training algorithm. The paper introduces the background of deep learning and its training algorithm, then summarizes current deep learning acceleration works. Further, it analyzes classic deep models training algorithm to explain the causes of deep learning performance problem. Based on the analysis, it lists the challenges about deep learning evolution and makes proposal to address them.

Key words: deep learning; deep learning acceleration; deep learning performance; distribution deep learning computing; hardware acceleration

人工智能是一门研究机器如何像人一样思考的科学, 已经有 60 年历史. 但在几年前, 人工智能技术还无法在厨房里区分出咖啡杯, 而今却已经能区分西伯利亚哈士奇和爱斯基摩哈士奇了[1]. 近年人工智能的快速发展得益于深度学习技术所带来的曙光. Google 公司将深度学习技术应用于 Android 的语音识别功能上, 将错误率降低了 25%. 微软于 2015 年发布的基于深度学习的图片识别研究成果, 其识别能力已经超过

① 基金项目:国家高技术研究发展计划(863)(2012AA010905);国家自然科学基金(61370081)
收稿时间:2015-12-28;收到修改稿时间:2016-01-27 [doi:10.15888/j.cnki.csa.005294]

人类的识别水平[2]. 2013 年《MIT 技术评论》将深度学习技术列为年度十大突破性技术之首.

深度学习之所以能带来如此巨大的进步主要在于它能学习到具有多层次抽象的特征. 此前, 机器学习使用领域专家精心设计的特征提取方法提取数据特征, 这些特征通常是比较低级和简单的特征(如图片的 SIFT 和 SURF 特征等). 由于简单特征的表达能力很有限, 所以导致后续处理效果不理想. 相比而言, 深度学习技术通过模拟大脑皮层的多层组织结构, 构造复杂的多层人工神经网络模型(微软 Adam 包含约 20 亿个连接), 通过大规模训练数据的多层次学习(约 1500 万张高维训练图片), 最终得到可以提取数据的不同抽象层度特征的深度模型. 因此, 深度学习在处理复杂的无约束问题如计算机视觉和语音识别等问题有巨大的优势.

但是天下没有免费的午餐, 深度学习成功的背后是巨大的计算资源消耗. 例如 Google 的图片识别模型包含约 10 亿个连接, 在 1000 台机器 16000 个核的计算集群上, 使用 1000 万张图片作为训练集, 需要 3 天才完成训练任务[3]. 所以, 深度学习给机器学习带来的进步, 是建立在巨大的计算资源消耗和庞大训练数据之上的. 百度首席科学家吴恩达教授认为深度学习的前沿正在向高性能计算转移.

近年处理器技术迅速发展, 处理器核数不断增加, 计算能力不断提高, 如 Intel E7-8890 v3 通用处理器包含 18 个物理核, 最高支持 36 个线程. NVDIA GTX TITAN X 显卡的流处理器已经达到 3072 个; Xilinx 2015 年发布的 Virtex UltraScale+ FPGA 的系统逻辑单元已经达到 3578 个. 硬件的快速发展给深度学习训练加速提供了优化基础, 然而由于深度学习模型复杂度高(Google 图片识别模型约 10 亿连接, 微软 Adam 约 20 亿连接), 训练数据规模庞大(Google 图片识别使用 1000 万张高维训练图片, 微软 Adam 使用约 1500 万张高维训练图片)以及训练算法并行度低的原因, 深度学习加速研究困难重重. 因此, 如何利用深度学习的算法特点来设计能高效率全面利用硬件计算能力的加速方法显得十分迫切.

本文对当前主要的深度学习加速研究工作进行归纳总结, 并对经典深度学习模型训练进行性能测试和分析, 阐述和解释了深度学习性能问题产生的原因. 本文的后续组织结构如下:首先介绍了深度学习的发展背景和部分应用, 然后介绍了深度学习的训练算法, 并对当前主要的深度学习加速研究工作进行归纳总结. 同时对经典的深度学习模型训练进行性能测试和分析, 进一步明确了深度学习的性能的主要限制. 在此基础上, 我们展望了深度学习可能的发展方向.

1 深度学习

深度学习是机器学习的一个新分支, 机器学习的目标是从数据中学习知识并应用于未来数据的预测, 而深度学习技术通过模拟大脑皮层的多层结构组织结构, 设计出复杂的多层人工神经网络模型, 通过大规模训练数据的多层次学习, 最终从数据中学到不同抽象程度的知识, 为后续的处理提供了强有力的支持.

机器学习发展至今经历了两个重要的阶段, 浅层学习阶段和深度学习阶段. 浅层学习阶段是指在 20 世纪 90 年代提出各种各样的浅层机器学习模型后, 它们被广泛应用到许多人工智能领域, 其理论被不断的完善和发展的阶段; 深度学习阶段是指从 2006 年以来, 多层神经网络的训练难题被攻克后, 各式各样的深度模型如雨后春笋般浮现, 并在许多人工智能应用领域取得突破性进展的阶段.

浅层学习阶段的模型结构简单, 基本上只有一层隐含层或没有隐含层节点, 如高斯混合模型(Gaussian Mixture Model, GMM), 隐式马可夫模型(Hidden Markov Model, HMM), 支持向量机(Support Vector Machine, SVM)等[4]. 浅层模型的理论分析相对成熟, 并在许多应用中获得成功, 如网页搜索排序, 垃圾邮件过滤系统和各类推荐系统等. 但对于复杂的无约束的问题如大规模图片识别, 语音识别等问题, 浅层模型显得力不从心.

深度学习阶段的模型相对复杂许多, 其理论基础是基于现代神经科学的研究成果提出的. 深度模型是通过模拟大脑皮层的多层神经结构而构建的复杂多层人工神经网络, 如深度卷积神经网络(Deep Convolution Neural Network, DCNN)[5], 深度置信网络(Deep Belief Networks, DBN)[6], 堆叠自动编码器(Stacked Auto Encoder, SAE)[7]等. 深度模型的多层次结构目前被证实在计算机视觉处理和语音识别方面有巨大的优势, 并被逐渐应用到更多领域如信息检索, 自然语言处理等.

1) 计算机视觉: 计算机视觉是研究机器理解和处

理视觉信息的学科, 如何提取机器视觉信息如图片和视频的特征成为该学科的难题, 而模仿大脑视皮层工作机制的深度卷积神经网络能提取视觉信息中的不同程度抽象的特征, 在计算机视觉应用里取得突破性进展. 其中, 微软 2015 年的图片识别研究在 2014 年模视觉识别挑战赛(ImageNet Large Scale Visual Recognition Challenge, ILSVRC)测试集上的识别能力已经超过人类的水平[2]. Yi Sun 等人基于卷积神经网络技术而设计人脸识别技术, 在户外脸部检测数据(Labeled Faces in the Wild, LFW)测试集上识别精度高达 99.15%, 识别能力也超过了人类的水平[8].

2) 语音识别: 语音识别是研究机器如何理解人类语言的学科, 以往的语音识别模型 HMM-GMM 忽略了原始语音数据内部原有的结构特征, 相比之下, 深度模型的多层结构可以更好地处理语音数据内部的相关性, 并拥有更大的弹性, 比传统的浅层模型更适合于语音处理任务. 如Dahl 等人提出的CD-DNN-HMM 模型比传统的 GMM-HMM 模型在实际应用数据上的识别错误率降低 23.2% [9]. 同时, Google 公司将深度学习技术应用到其 Android 智能手机的语音识别功能上, 使错误率降低了 25%.

3) 信息检索: 信息检索研究如何通过分析用户输入的查询信息, 从大量文档中迅速找到相关文档. 目前信息检索主要采用语义相关的方法将查询信息在语义层面上映射到相关文档. Huang 等人提出 DSSM 模型, 在大规模的实际数据上能够取得更好的准确度[10]. Shen 等人将卷积神经网络和 DSSM 模型结合起来, 设计了 C-DSSM 模型, 进一步提升了准确度[11].

除了以上列举的应用外, 深度学习还被应用于其他领域并取得非凡的成绩, 如自然语言处理[12] 和生物信息领域[13]等.

2 深度学习训练算法和加速技术

虽然深度学习算法准确性高, 在许多应用领域取得了成功, 但由于计算复杂, 深度学习算法也存在巨大的限制. 本章首先介绍了深度学习训练算法, 然后对硬件的快速发展进行概述, 最后对目前主要的深度学习加速研究进行归纳总结.

2.1 深度学习训练算法概述

深度学习发展十分迅速, 目前深度模型有深度卷积神经网络(Deep Convolution Neural Network, DCNN), 深度置信网络(Deep Belief Networks, DBN), 堆叠自动编码器(Stacked Auto Encoder, SAE)等, 它们均是由多层神经元节点层叠组成, 其训练算法主要分为两个过程: 初始化和迭代训练.

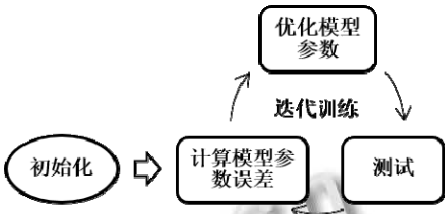

图 1 深度学习训练算法流程

1) 初始化: 初始化过程首先使用随机数初始化模型参数, 随后对某些模型如深度置信网络, 还需要进行预训练. 该过程非常重要, 可以帮助模型避免在后续的迭代训练过程中陷入局部最优.

2) 迭代训练: 迭代训练过程首先通过读取和处理训练数据并比较实际输出和预期输出, 得到训练误差. 然后使用误差反向传播(Error Back Propagation)方法, 将训练误差逐层反向传递, 计算出每个参数的参数误差, 随后通过参数优化算法更新模型参数. 最后测试是否满足迭代训练的终止条件来选择结束训练还是进行下一次迭代.

迭代训练具体包含 3 个子过程, 分别是计算模型参数误差, 优化模型参数以及测试过程, 它们的算法流程描述如下:

1) 计算模型参数误差: 计算参数模型误差过程首先读取训练数据作为模型第一层的输入, 并将第一层网络的输出结果作为下一层的输入, 以此类推直至最后一层输出计算结果, 并对比实际输出和预期输出之间的误差, 计算出最后一层神经元的参数误差和误差因子, 并将误差因子反向传至前一层网络. 此后, 每一层网络接受到前一层传来的误差因子, 计算出本层的参数误差和新误差因子, 并将新误差因子继续反向传递直至第一层为止. 总的来说, 这个过程将训练数据和预期输出作为输入, 通过比较预期输出和实际输出之间的误差来推导模型的参数误差.

2) 优化模型参数: 在模型参数误差计算完毕后, 优化算法使用特定的方法对参数误差进行处理, 并将每个处理后的参数误差累加到目前的参数上, 实现模型参数更新. 目前使用主要的参数优化方法是随机梯度下降算法(Stochastic gradient descent, SGD), 该算法

认为前一次迭代的参数误差对本次迭代的参数误差计算具有指导作用, 因此, 该算法使用动量系数 μ 和学习速率 α 将本次的参数误差和上一次的计算得到的参数误差融合起来, 计算出本次参数误差, 最后将参数误差累加到模型参数上, 完成参数更新. 该过程可以由以下公式表示, 其中 W 表示模型参数, t 表示迭代次数, V 表示优化后的模型参数误差:

$$a \quad V_t = \mu V_{t-1} - \alpha \nabla (W_{t-1}) \tag{1}$$

$$b \quad W_t = W_{t-1} + V_t \tag{2}$$

3) 测试: 在一次或者数次参数更新后, 使用测试数据对模型进行测试, 如果测试精度达到预期目标或迭代计数的数值已经达到最大值就终止整个迭代训练过程, 否则增加迭代计数, 并继续读取训练数据, 回到参数误差计算过程, 进行下一次迭代.

2.2 深度学习加速技术

目前处理器处理能力不断提高, 为了满足深度学习不断增长的性能需求, 许多学者对深度学习算法的加速进行了研究. 本节介绍了硬件快速发展的情况, 并对目前主要的深度学习加速研究进行归纳总结.

2.2.1 硬件发展情况

近年随着处理器的制作工艺不断提升, 晶体管越来越密集, 处理器核数越来越多, 使得处理器的整体计算能力越来越强. Intel 的至强处理器 E7-8890 v3 采用 22 纳米的制作工艺, 最后一级高速缓存(Last Level Cache)增长到 45MB, 拥有 18 个物理核, 最高支持 36 个线程. NVDIA 的 GTX TITAN X 显卡采用 28 纳米制作工艺, 其中流处理器已经达到 3072 个, 显存扩展到了 12GB, 显存带宽提升到 36.5GB/sec. Xilinx 于发布的 Virtex UltraScale+ FPGA 采用 16 纳米制作工艺, 系统逻辑单元已经达到 3578 个.

2.2.2 加速技术

硬件的快速发展给深度学习的加速提供了硬件基础. 当前对普通规模的深度学习训练的加速方法主要有两种思路, 第一种思路是设计专用处理器, 利用算法的特点来设计硬件电路, 最大化计算单元和高速存储单元的利用率. 另一种思路是发掘算法中的可并行部分, 利用多核技术进行并行加速.

对于无法使用单机完成的大规模深度学习训练, 其加速策略是通过分布式计算的方式, 将庞大的训练模型切分至不同节点, 每台机器只对模型的一部分进行训练, 最终完成整个模型的训练.

针对以上的 3 种不同加速方法, 对深度学习近年的加速研究进行归纳和总结:

1) 专用处理器加速: 专用处理器是针对一个算法或算法家族而设计的处理器, 具有功耗低, 速度快, 开发成本高的特点. 设计一个专用处理器首先需要分析目标算法的特性, 然后根据算法的特点进行电路设计, 最大化硬件资源利用率. Farabet 等人根据卷积算法和池化算法的窗口移动特点, 优化寄存器的使用, 最后通过加速卷积和池化计算来加速卷积神经网络算法[1]. Merolla 等人针对神经元结构设计硬件神经元, 将神经元之间的连接映射到锁存器或其他存储器, 最终通过把神经网络上的神经元映射到硬件神经元来加速神经网络算法[14]. Chen 等人从内存优化角度设计了一个 DNNs 硬件加速器, 其根据算法内存分配的特点设计了不同的存储单元, 不同种类数据使用不同的存储单元, 最后通过流水线的方式提高计算单元的利用率, 以低于通用处理器约 20 倍的功耗, 将计算速度提升了约 117 倍[15], 随后使用该加速器, 设计了一个多片的 DNNs 硬件系统以低于 NVDIA 20M 通用图像处理器大约 150 倍的功耗, 将计算速度提升了约 450 倍[16].

2) 并行加速: 并行加速利用多核处理器, 对算法中的可并行部分进行并行计算来实现算法加速. 由于深度学习训练算法每个过程之间互相依赖, 所以并行优化难度大, 目前的并行加速方法主要是通过并行加速深度学习训练算法中的高维数据计算来对整个算法进行加速. Vanhoucke 等人针对 x86 平台, 利用特殊的 SSE 指令对算法中的高维数据计算进行并行加速, 并通过定点运算代替浮点的方法进一步进行加速[17]. Raina 等人在 GPU 上实现了 DBNs 和 Sparse Coding 的非监督训练算法, 通过将批量的训练数据分配到不同 Block 处理, Block 内部使用多线程处理每一份训练数据的二级并行方式, 在 NVDIA GeForce GTX 280 上相对于双核 3.16GHz 的 CPU, DBN 的训练速度提高了 10~70 倍, Sparse Coding 的训练速度提高了 5~15 倍[18]. Jia 等人使用按需分配的内存管理策略, 使用 NVDIA 的深度学习加速库 cuDNN 对深度学习训练算法进行并行加速[19]. Krizhevsky 等人同将卷积神经网络切成两部分, 并分配到不同 GPU 上, 通过高速互联总线进行数据同步的方式对卷积神经网络进行并行加速[5].

3) 分布式学习系统: 分布式计算通过将计算任务

分解到多台机器节点的方式对计算任务进行加速. 分布式学习系统通过模型复制和模型切分的方式, 使用数据并行和模型并行对训练任务进行加速. 由于深度学习训练算法理论上不能异步处理训练数据, 这对分布式系统的性能会有巨大影响. Google 提出了一种异步优化方法, 可以利用高达 2000 的处理核进行学习训练[20]. Li 等人提出一种折中的延迟边界(bounded delay)方法, 在异步程度和算法收敛效率进行折中, 最后使得整体运行时间最短[21]. 模型复制和切分会导致模型参数的不一致, 目前许多分布式学习系统如 Google 的 DistBelief 系统[20]和微软的 Adam 系统 [22]使用参数服务器来维护最新的模型参数. 同时, 节点间通信量大、节点运行速度不一致导致和负载均衡也是分布式学习系统要解决的问题. 节点间通信量大影响系统整体性能, Li 等人通过过滤影响小的参数误差来减轻通信传输, 并通过累计备份的方式, 减少容错的通信开销[21]. 微软的 Adam 使用传递误差梯度来代替参数误差的方式来减少通信量[22]. 节点运行速度不一致将导致节点等待问题. 微软 Adam[22]让每个线程处理不同的训练数据并且完成一定量训练后不再等待慢机器而提前开始下一批训练的方式缓解机器节点速度不一致的影响. 分配不均的负载也会降低资源利用率, 微软 Adam[22]将训练系统的服务器细分成数据处理服务器、训练服务器和参数服务器, 来均衡各个部分的计算.

3 深度学习训练算法性能分析

为了深入分析深度学习训练加速不理想的问题, 本章对经典的深度模型从计算复杂度, 内存消耗和并行加速这 3 方面进行测试分析, 进一步明确了深度学习的性能的主要限制.

3.1 测试环境

本次性能测试分析实验使用的机器具体配置如表 1 机器配置所示.

表 1 机器配置

类别	配置
处理器 CPU	Intel E5-2650 v3 2.30GHz
内存	26GB
操作系统	Ubuntu 14.04
GPU	NVIDIA K80

该机器运行 Ubuntu 14.04 操作系统, 配置有 2 个主频为 2.30GHz 的 Intel E5-2650 v3 CPU, 每个 CPU 包含 12 个物理核, 一共 24 个物理核, 并配置 26GB 内存以及一个包含 4992 个流处理器的 NVIDIA K80 显卡.

首先, 使用 OpenCV 的 JPEG 解码算法读取每一张, 重复 100 次. 随后, 使用图片作为模型训练图片进行训练, 也重复 100 次. 对以上程序, 我们使用 VTune 记录程序的运行时间和执行指令数, 最后计算出均摊到每张图片上的处理时间和执行指令数, 以 JPEG 解码算法的测试数据为基准, 分别与 NIN, AlexNet 以及 GoogLeNet 模型的测试结果进行比较, 其结果如图 2 所示.

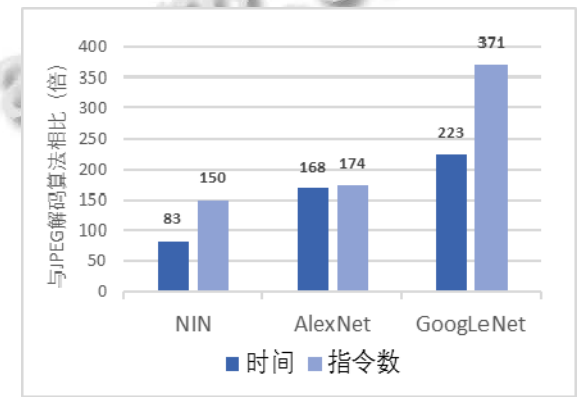

图 2 算法复杂度测试结果

NIN 模型训练训练一张图片的时间大约相当于解码 83 张 JPEG 图像, 执行的指令数是 JPEG 解码算法的 150 倍; AlexNet 模型训练一张图片的时间相当于解码 168 张 JPEG 图像, 执行的指令数是 JPEG 解码算法的 174 倍; GoogLeNet 模型训练一张图片的时间相当于解码 223 张 JPEG 图片, 执行的指令数是 JPEG 解码算法的 371 倍.

不同模型由于其参数大小和层数不相同, 其处理每一张训练图片的时间也不同. GoogLeNet 模型训练一张图片的时间接近于 NIN 模型训练一张图片时间的 3 倍. 但是即使是处理速度较快的 NIN 模型, 其训练一张图片的时间和执行的指令数都是 JPEG 图片解码算法的近百倍. 而随着深度学习的发展, 计算模型会越来越大, 越来越复杂, 这将导致未来深度模型算法的计算复杂度越来越高. 所以计算复杂度高是导致深度学习性能问题产生的原因之一.

3.3 内存消耗分析

深度学习训练算法的内存消耗主要包括三部分, 第一部分是模型参数产生的内存消耗, 第二部分是模

型中层与层之间信息传递产生的内存消耗, 第三部分是由于编程实现需要而产生的内存消耗. 本次内存消耗测试主要测试第一部分和第二部分内存开销.

本次内存消耗测试包含两个测试, 第一个测试通过在 ILSVRC12 训练集上使用 Batch 为 64 的随机梯度下降(SGD)算法对 NIN, AlexNet 以及 GoogLeNet 进行训练, 测试模型参数带来的内存开销和训练过程中的内部通信内存开销. 第二个测试针对不同 Batch 配置, 分别为 16, 32, 64, 128 和 256, 在 ILSVRC12 训练集上对以上模型进行训练, 记录了其内部通讯开销. 具体测试结果如下表 2 和表 3 所示.

表 2 Batch 为 64 时的内存消耗

模型	模型参数	内部通信
NIN	26MB	1771MB
AlexNet	218MB	837MB
GoogLeNet	52MB	6847MB

表 3 不同 Batch 配置的内部通信内存消耗

模型/Batch	16	32	64	128	256
NIN	442MB	885MB	1771MB	3542MB	7084MB
AlexNet	209MB	418MB	837MB	1674MB	3348MB
GoogLeNet	1711MB	3423MB	6847MB	13694MB	27388MB

如果用 4 字节的浮点数来表示每个模型的参数, NIN 模型参数内存消耗约为 26MB, 而 GoogLeNet 约为 52MB, AlexNet 高达约 218MB. 而模型参数产生的内存消耗相比于内部通信产生的内存消耗显得微不足道. 表 2 是 Batch 大小设置为 64 的内部通信测试结果. NIN 内部通信需要消耗 1771MB 的内存, 是其模型参数内存消耗的 68 倍, AlexNet 的内部通信消耗达到 837MB 内存, 是其模型参数内存消耗的 4 倍左右, GoogLeNet 的内部通信消耗更高达 6847MB 内存, 是其模型参数内存消耗的 131 倍.

其中 AlexNet 的模型参数较大而内部通信相对较小的原因是其相对包含较多的全连接层, 而 NIN 和 GoogLeNet 则包含相对较多的卷积层. 而卷积层的参数共享特点大大降低了模型的参数数量, 但并不能降低网络内部的通信量.

巨大的内存开销带来的数据通信问题会给算法加速设计带来许多阻碍, 也是导致深度学习性能问题的一个原因.

3.4 并行加速分析

深度学习训练算法计算复杂度高, 处理时间长, 因此对其进行加速显得十分必要. 本次加速测试使用 2014 年发表于国际顶级会议 ACM MM 的深度学习训练框架 Caffe[19]在包含 24 个物理核, 26GB 内存以及拥有 4992 个流处理器的显卡的测试机器上进行并行加速测试.

本次并行加速测试包含两个测试, 第一个测试使用在测试机器上使用 2 线程, 4 线程, 8 线程和 16 线程对 NIN, AlexNet 和 GoogLeNet 在 ILSVRC12 训练集上进行并行加速训练测试. 第二个测试在 NVIDA K80 GPU 上, 针对随机梯度下降算法的不同 Batch 配置分别为 1, 16, 32 和 64, 测试不同 Batch 配置对并行加速的影响. 关于多核平台上的加速比测试的结果如表 4 和图 3 所示.

表 4 多核平台上的加速比

模型/线程	2 线程	4 线程	8 线程	16 线程
NIN	1.35x	1.84x	2.17x	1.86x
AlexNet	1.28x	1.53x	1.76x	1.75x
GoogLeNet	1.17x	1.22x	1.13x	1.21x

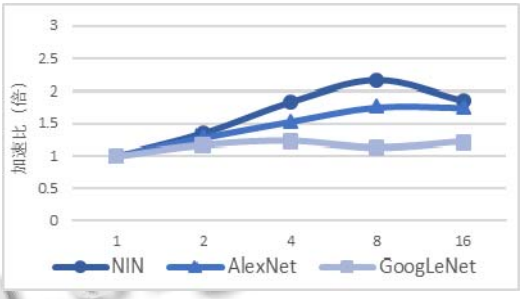

图 3 多核平台上的加速比

从测试结果可以看出, 在多核平台上, 模型的并行加速效果随着使用的核数的增多而非常缓慢的增长, 甚至反而变得更慢.

通过分析 Caffe 的源代码, 我们了解到 Caffe 通过对训练算法中的每一个高维数据计算进行并行加速来加速训练算法. 这样的策略会导致并行加速效果差以及糟糕的可拓展性, 因为高维数据计算的并行加速效果和输入关系密切, 而训练算法中高维数据计算的输入规模波动巨大, 所以加速效果参差不齐, 而且不断地启动线程和同步等待产生了额外开销, 如 NIN 模型和 AlexNet 模型的 16 线程的加速效果比 8 线程的加速效果更差. 这使得 Caffe 在多核平台上的加速效果和

并行拓展性十分糟糕.

为了测试训练参数对算法性能的影响, 我们使用不同的 Batch 训练参数, 分别为 1, 16, 32 和 64 在 NVIDA K80 GPU 在 ILSVRC12 训练集上对模型进行训练, 并以 Batch 配置为 1 的测试结果作为基准, 将 Batch 配置 16, 32 和 64 的测试数据与之比较, 并将加速比记录下来, 如表 5 所示.

表 5 不同 Batch 参数下的加速比

模型/Batch	16	32	64
NIN	5.09x	5.94x	6.35x
AlexNet	4.03x	5.02x	5.49x
GoogLeNet	5.43x	6.18x	6.50x

从表 5 得知, Batch 配置为 16, 32 和 64 时训练速度均比 Batch 配置为 1 时快大约 5~6 倍, 这是因为随着 Batch 的增加, 高维数据计算的输入也随之增加, 能更好地发挥并行的优势. 随着 Batch 的继续提高, 其运行速度会继续提升, 只是提升幅度非常小. 但是从内存开销测试中得知, Batch 的增大会导致内存开销迅速增长, 因此受到 GPU 显存的限制, Batch 的大小不能设置太大, 特别是对于 GoogLeNet 来说, 当 Batch 设置为 128 时, 其内存占用会超过 12G, 而 12G 已经是目前 NVDIA 显卡内存可以达到的最大值. 另外 Batch 的设置会影响训练算法的收敛效率, 一般来说, Batch 设置越大, 算法收敛速度越慢, 这意味着需要更多的迭代次数, 会导致整体模型训练时间变长.

所以, 虽然 Batch 的增加可以提高每张图片的平均处理速度, 但是由于内存的限制和 Batch 的增加对训练算法收敛效率的影响, 不能仅通过增加 Batch 大小来提高运行速度, 而是针对每一个模型和机器配置, 设置恰当的 Batch 参数.

从本次性能测试结果来看, 目前的并行加速训练算法的高维数据计算来达到加速整体训练算法的方式并不能很好地发挥硬件计算能力, 而且并行拓展性能差, 同时还会受到训练参数 Batch 的影响, 加速效果不稳定.

3.5 深度学习的性能问题

根据前面的分析, 可以发现深度学习的性能问题主要由以下 4 方面导致, 分别是: 算法并行度低, 计算复杂度高, 内存开销巨大以及迭代次数多.

1) 算法并行度低: 深度学习训练算法首先对模型进行初始化, 随后开始迭代训练过程. 迭代训练期间, 首先根据训练数据计算模型参数误差, 随后使用优化算法对参数误差进行优化处理, 并将处理后的参数误差累加到模型参数上来完成参数更新, 至此, 一次迭代训练完成. 最后对更新后的模型进行测试来决定是否继续进行迭代训练. 从算法流程得知, 下一次迭代训练必须在本次迭代训练完成后进行, 所以深度学习训练算法理论上不能并行处理多个迭代, 这给深度学习的并行加速带来许多阻碍. 而目前的并行加速训练算法的高维数据计算来达到加速整体训练算法的方式并不能很好地发挥硬件计算能力, 而且并行拓展性能差, 同时还会受到训练参数 Batch 的影响, 加速效果不稳定.

2) 计算复杂度高: 从前面的计算复杂度测试可以看出, 深度学习训练算法处理每个训练数据时执行的指令数多, 消耗的时间长. 相比较于 JPEG 解码算法, 相对简单的 NIN 模型和 AlexNet 模型的训练数据处理时间分别是 JPEG 解码算法的 83 倍和 168 倍, 而 GoogLeNet 的训练数据处理时间高达 JPEG 解码算法的 223 倍. 而随着深度学习的发展, 模型会越来越大, 越来越复杂, 这将导致未来深度模型算法的计算复杂度越来越高.

3) 内存开销巨大: 从内存开销测试中, 我们观察到模型参数产生的内存消耗已经相当巨大, 其中 AlexNet 模型的参数导致的内存开销就已经达到 218MB. 另外, 内部通信产生的内存消耗更大, 当 Batch 大小设置为 64 时, NIN 内部通信需要消耗是其模型参数内存消耗的 68 倍, GoogLeNet 的内部通信消耗是其模型参数内存消耗的 131 倍. 巨大的内存开销带来的数据通信问题会给算法加速设计带来许多阻碍.

4) 迭代次数多: 深度学习训练算法是个迭代算法, 一般需要进行多次迭代计算. 在 ImageNet 上对以上模型进行训练, GoogLeNet 大致需要 100 万次迭代, 意味着要处理多达 3 亿多张图片(每次迭代处理 32 张图片); AlexNet 和 NIN 均需要 45 万次迭代, 分别需要处理 1 亿多张图片和 3000 万张图片. 而随着模型的增大和训练数据的增多, 完成模型训练需要迭代的次数会更多, 如果不对深度学习训练算法进行加速, 那将无法在合理的时间内完成模型的训练工作, 这会严重影响深度学习模型的设计工作.

总的来说, 深度学习训练算法的并行度低, 计算

复杂度高, 内存开销巨大以及迭代次数多的特点是导致深度学习训练算法性能问题的原因, 也是使得加速研究工作困难重重的原因.

4 深度学习的挑战和展望

深度学习通过设计复杂的多层神经网络模型, 使用庞大的训练数据, 消耗大量的计算资源对其进行训练, 最终学习到提取数据的多层次抽象特征的能力, 给人工智能的研究带来了曙光. 而深度学习训练算法的并行度低, 计算复杂度高, 内存开销巨大以及迭代次数多的特点使得深度学习训练算法存在严重的性能问题, 亟待研究者解决.

目前学术界和产业界主要通过专用硬件, 并行技术以及分布式技术对深度学习训练进行加速, 然而这些加速技术存在一定的使用限制且仍有优化空间. 专用加速硬件需要将训练模型放入高速片上存储中, 因此会受模型大小所限制; 由于训练算法并行度低, 目前并行技术难以完全发挥硬件的计算能力; 分布式技术导致节点间大量通信问题以及节点运行速度的不一致性对系统整体性能有不可小觑的影响. 并且, 由于缺乏一个具有代表性的深度学习训练算法基本测试程序集, 对深度学习加速效果的评估工作难以进行, 阻碍深度学习加速研究的前进.

同时, 根据目前研究显示, 采用更多的训练数据, 增大训练模型成为了提高深度学习识别精度的重要方法. 越来越复杂的模型, 越来越多的训练数据最终需要更多的更强的计算能力. 这意味着深度学习的性能问题对深度学习发展带来的影响将会越来越大.

另外, 随着目前可穿戴设备和智能硬件的发展, 可分析的个人数据越来越多, 如手腕的加速计可以捕捉运动信息, 麦克风捕捉声音信息, 智能家电捕捉使用习惯信息等. 在计算能力和存储能力有限的移动设备上结合深度学习技术, 利用与日俱增的个人数据使得智能设备更加智能也是深度学习未来发展的重要方向.

因此, 针对以上的挑战, 我们对深度学习未来的发展提出了以下值得探索的方向:

1) 设计自适应的专用硬件: 专用硬件由于需要高效利用处理单元, 所以通常将整个模型载入到高速的片上存储以加快数据读写速度. 但是当模型无法载入片上存储时, 可能直接导致硬件失效或者处理速度大大下降. 为了适应不同大小的模型, 需要根据模型的大小来选择恰当的存储策略, 扩大专用处理器的使用范围.

2) 算法设计需体现更多并行元素: 原有深度学习算法设计过程中, 主要关注算法精度而没有把可并行性作为关键因素考虑, 因此目前深度学习算法不能充分利用多核计算能力. 为了更好地发挥多核平台的潜力, 在设计算法时, 除了关心算法的训练精度还要将算法的并行度纳入考虑范围.

3) 设计面向体系结构和系统的深度学习基准测试程序集: 测试程序集是体系结构和系统评估的标准, 具有代表性的测试程序集是设计高效体系结构和系统的基础. 虽然深度学习算法已经应用在越来越多的领域. 然而, 其面临的性能挑战依然突出, 如何为其设计高效的体系结构和系统也不明确. 因此, 需要设计一套具有代表性的面向体系结构和系统的深度学习算法的测试程序集.

4) 多层次组合的参数训练算法: 移动设备的计算能力和存储能力十分有限, 目前的深度学习模型相对复杂, 不论计算还是存储都超出移动设备的能力. 虽然可以通过云端进行处理, 但显然增加带宽等的方面的压力. 因此, 如果可以设计多层次的模型, 对处理任务进行划分, 如将复杂的训练过程划分到云端, 将相对简单的预测过程划分在移动端. 或者使用多重模型策略, 移动端使用简单模型, 云端使用复杂模型, 将两者结合起来, 共同完成数据学习和预测.

5 结语

深度学习给人工智能的发展带来了曙光, 但是深度学习所带来的进步, 是建立在庞大训练数据和巨大的计算资源消耗之上的. 因此, 加速深度学习算法是其发展的重要基础. 本文通对当前主要深度学习加速技术进行归纳总结, 以及对经典深度模型进行性能的测试分析来阐述和解释深度学习性能问题产生的原因. 在此基础上, 还列出了目前深度学习发展存在的挑战并对其未来的发展趋势进行了展望. 可以预见, 随着硬件的快速发展以及加速技术的进步, 深度学习的性能问题将会被更好地解决, 在更多的领域带来突破.

参考文献

1 Szegedy C, Liu W, Jia Y, et al. Going deeper with convolu-

tions. arXiv preprint arXiv:1409.4842, 2014.
2 He K, Zhang X, Ren S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv preprint arXiv:1502.01852, 2015.
3 Le QV. Building high-level features using large scale unsupervised learning. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2013. 8595–8598.
4 Deng L. A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Trans. on Signal and Information Processing, 2014, 3: e2.
5 Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 2012: 1097–1105.
6 Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Computation, 2006, 18(7): 1527–1554.
7 Vincent P, Larochelle H, Lajoie I, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research, 2010, 11: 3371–3408.
8 Sun Y, Chen Y, Wang X, et al. Deep learning face representation by joint identification-verification. Advances in Neural Information Processing Systems, 2014: 1988–1996.
9 Dahl G E, Yu D, Deng L, et al. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Trans. on Audio, Speech, and Language Processing, 2012, 20(1): 30–42.
10 Huang PS, He X, Gao J, et al. Learning deep structured semantic models for web search using clickthrough data. Proc. of the 22nd ACM International Conference on Conference on Information & Knowledge Management. ACM. 2013. 2333–2338.
11 Shen Y, He X, Gao J, et al. Learning semantic representations using convolutional neural networks for web search. Proc. of the Companion Publication of the 23rd International Conference on World Wide Web Companion. International World Wide Web Conferences Steering Committee. 2014. 373–374.
12 Sarikaya R, Hinton G E, Deoras A. Application of deep belief networks for natural language understanding. IEEE/ACM Trans. on Audio, Speech, and Language Processing, 2014, 22(4): 778–784.
13 Chicco D, Sadowski P, Baldi P. Deep autoencoder neural networks for gene ontology annotation predictions. Proc. of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics. ACM. 2014. 533–540.
14 Merolla P, Arthur J, Akopyan F, et al. A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. Custom Integrated Circuits Conference (CICC), 2011 IEEE. IEEE. 2011. 1–4.
15 Chen T, Du Z, Sun N, et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. ACM SIGPLAN Notices. ACM, 2014, 49(4): 269–284.
16 Chen Y, Luo T, Liu S, et al. Dadiannao: A machine-learning supercomputer. 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE. 2014. 609–622.
17 Vanhoucke V, Senior A, Mao M Z. Improving the speed of neural networks on CPUs. Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop. 2011, 1.
18 Raina R, Madhavan A, Ng AY. Large-scale deep unsupervised learning using graphics processors. Proc. of the 26th Annual International Conference on Machine Learning. ACM. 2009. 873–880.
19 Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding. Proc. of the ACM International Conference on Multimedia. ACM. 2014. 675 – 678.
20 Dean J, Corrado G, Monga R, et al. Large scale distributed deep networks. Advances in Neural Information Processing Systems, 2012: 1223–1231.
21 Li M, Andersen DG, Park JW, et al. Scaling distributed machine learning with the parameter server. Proc. OSDI. 2014. 583–598.
22 Chilimbi T, Suzue Y, Apacible J, et al. Project adam: Building an efficient and scalable deep learning training system. The 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14). 2014. 571–582.