**【深度学习系列1】 深度学习在腾讯的平台化和应用实践**

mary  发表于 2014.8.4     3510浏览    0讨论

**引言：**深度学习是近年机器学习领域的重大突破，有着广泛的应用前景。随着Google公开Google Brain计划，业界对深度学习的热情高涨。腾讯在深度学习领域持续投入，获得了实际落地的产出。我们准备了四篇文章，阐述深度学习的原理和在腾讯的实践，介绍腾讯深度学习平台Mariana，本文为第一篇。

深度学习（Deep Learning）是近年来机器学习领域的热点，在语音识别、图像识别等领域均取得了突破性进展。腾讯提供广泛的互联网服务，在2014年第一季度，即拥有3.96亿月活跃用户的微信，8.48亿月活跃用户的QQ，以及6.44亿月活跃用户的Qzone等平台，拥有海量数据和众多应用，深度学习在腾讯有广泛的潜在应用场景。

深度学习是近年来机器学习领域最令人瞩目的方向。自2006年深度学习界泰斗Geoffrey Hinton在Science杂志上发表Deep Belief Networks [1]的论文后，重新激活了神经网络的研究，开启了深度神经网络的新时代。学术界和工业界对深度学习热情高涨，并逐渐在语音识别、图像识别、自然语言处理等领域获得突破性进展。深度学习在语音识别领域获得相对20%到30%的准确率提升，突破了近十年的瓶颈。2012年图像识别领域在ImageNet图像分类竞赛中取得了85%的top5准确率 [2]，相比前一年74%的准确率有里程碑式的提升，并进一步在2013年获得89%的准确率。目前Google、Facebook、Microsoft、IBM等国际巨头，以及国内百度、阿里巴巴等互联网巨头争相布局深度学习。

深度学习通过构建深层神经网络，来模拟人类大脑的工作原理。如图1所示，深层神经网络由一个输入层，数个隐层，以及一个输出层构成。每层有若干个神经元，神经元之间有连接权重。每个神经元模拟人类的神经细胞，而结点之间的连接模拟神经细胞之间的连接。

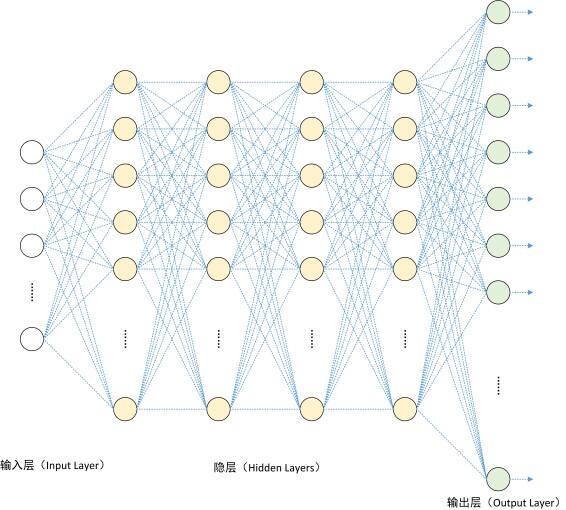


                                   图1：深度神经网络的结构

然而，深度学习要在腾讯落地，面临巨大的挑战。

首先，深度神经网络模型复杂，训练数据多，计算量大。一方面，DNN需要模拟人脑的计算能力，而人脑包含100多亿个神经细胞，这要求DNN中神经元多，神经元间连接数量也相当惊人。从数学的角度看，DNN中每个神经元都包含数学计算（如Sigmoid、ReLU或者Softmax函数），需要估计的参数量也极大。语音识别和图像识别应用中，神经元达数万个，参数数千万，模型复杂导致计算量大。另一方面，DNN需要大量数据才能训练出高准确率的模型。DNN参数量大，模型复杂，为了避免过拟合，需要海量训练数据。两方面因素叠加，导致训练一个模型耗时惊人。以语音识别为例，目前业界通常使用样本量达数十亿，以CPU单机需要数年才能完成一次训练，用流行的GPU卡也需要数周才能完成训练。

其次，深度神经网络需要支持大模型。以图像识别为例，已有工作证明通过增加卷积层的filter数量，加大模型深度等，可以获得更好的模型质量，但模型参数也随之增加。然而，以ImageNet 2012竞赛冠军的网络为例，其占用3.99 GB的显存，已接近主流GPU的显存容量，试图增大模型则会超过GPU显存范围。因此，如何支持更大模型是一个大的挑战。

再次，深度神经网络训练收敛难，需要反复多次实验。深度神经网络是非线性模型，其代价函数是非凸函数，容易收敛到局部最优解。同时，深度神经网络的模型结构、输入数据处理方式、权重初始化方案、参数配置、激活函数选择、权重优化方法等均可能对最终效果有较大影响。另外，深度神经网络的数学基础研究稍显不足。虽然可以通过限制性波尔兹曼机（Restricted Boltzmann Machines，RBMs），或者（DenoisingAutoencoder，DAEs）等产生式建模方法初始化网络模型，以达到减少陷入局部最优的风险，但仍然不是彻底的解决方案，仍然需要在实际使用深度神经网络解决问题的时候，合理的利用海量数据，合理的选择优化方式。上述原因导致需要技巧、经验，基于大量实验来训练出一个效果好的模型。

因此，深度学习成为一个效果很好但门槛极高的方向，如何落地产生实际应用效果成为关注的焦点。

面对机遇和挑战，我们打造了腾讯深度学习平台Mariana。Mariana包括三个框架：深度神经网络（Deep Neural Networks，DNN）的GPU数据并行框架，深度卷积神经网络（Deep ConvolutionalNeural Networks，CNN）的GPU数据并行和模型并行框架，以及DNN CPU集群框架。Mariana通过此三个框架完成并行加速，针对多种应用场景，以解决深度学习训练极慢的问题；Mariana通过模型并行，支持大模型；Mariana提供默认算法的并行实现以减少新算法开发量，简化实验过程。Mariana面向语音识别、图像识别、广告推荐等众多应用领域。

腾讯深度学习平台Mariana以GPU服务器为主，每台服务器配置4或者6块Nvidia Tesla系列高端科学计算用GPU卡。利用每块GPU卡2000多个流处理器的强大计算能力，并实现多GPU卡并行以加速训练。

腾讯深度学习平台Mariana重点研究多GPU卡的并行化技术，完成DNN的数据并行框架，以及CNN的模型并行和数据并行框架。数据并行和模型并行是Google分布式大神Jeff Dean和深度学习大佬Andrew Ng在2012年NIPS会议上发表的DistBelief[3] 论文中针对深度学习的CPU集群框架提出的定义。数据并行指将训练数据划分为多份，每份数据有一个模型实例进行训练，再将多个模型实例产生的梯度合并后更新模型。模型并行指将模型划分为多个分片，每个分片在一台服务器，全部分片协同对一份训练数据进行训练。我们学习并借鉴了这两种并行方式，并成功应用于单机多GPU卡的并行。

DNN的数据并行框架通过同步随机梯度下降进行训练。数据并行训练中，每个GPU卡各自训练，并各自产生一份梯度值，然后进行参数交换。图2. 展示了参数交换过程。每台GPU服务器配置6块GPU卡，其中四块通过树状的PCIe连接，并与另外两块GPU卡通过IOH连接。参数交换过程从逻辑上看，梯度收集阶段将全部梯度值累加起来，然后应用到当前模型以更新参数得到新模型，最后在模型分发阶段将新模型下发给全部GPU卡。采用数据并行后，相对于单卡训练过程，关键的问题是参数交换过程引入额外时间消耗，拖累了并行性能，使得加速比很难提高。我们通过一个精心设计的拓扑完成参数交换，提升整体性能。此外，我们采用近似的自适应学习率算法，使得支持自适应学习率所需交换的数据量降低了一个数量级。

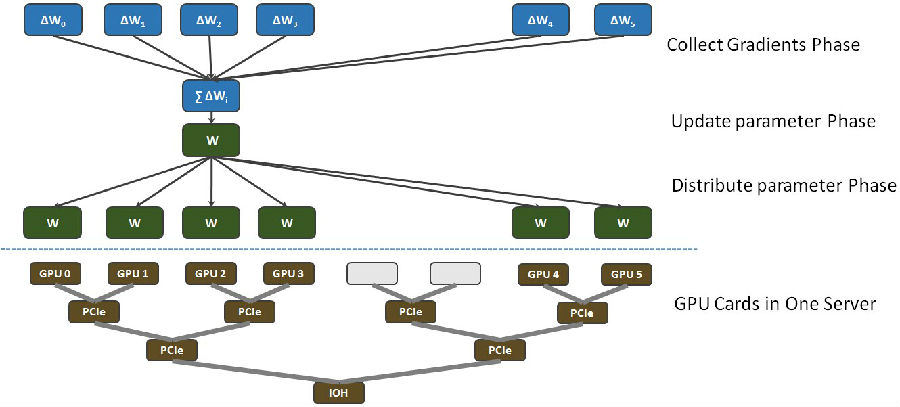


                                      图2：DNN GPU框架数据并行的参数交换过程

DNN的数据并行框架在微信语音识别中得到应用。微信中语音识别功能的入口是语音输入法、语音开放平台以及长按语音消息转文本等。对微信语音识别任务，通过Mariana，识别准确率获得了极大的提升，目前识别能力已经跻身业界一流水平。同时可以满足语音业务海量的训练样本需求，通过缩短模型更新周期，使得微信语音业务可以及时满足各种新业务需求。

卷积神经网络CNN的模型并行和数据并行框架的结构如下图所示：

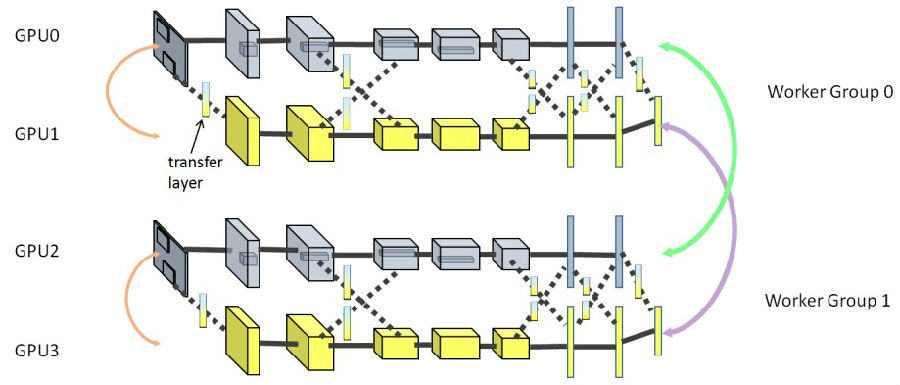


                                  图3：CNN GPU框架的模型并行和数据并行架构

CNN模型并行和数据并行框架对GPU卡分组，组内两个GPU卡做模型并行，组间做数据并行。如上图所示，4个GPU卡分成Worker Group 0和1。组内两卡各持有CNN模型的一部分，称为partition，协作完成单个模型的训练。模型并行中，卡间数据传输通过引入Transfer Layer透明的完成。组间数据并行按同步随机梯度下降进行训练，并采用精巧的拓扑完成参数交换，但注意只有各组内属于同一个partition的数据各自交换，即图中GPU0和GPU2、GPU1和GPU3分别进行参数交换。引入数据并行和模型并行后，从磁盘读取训练数据，训练数据预处理，CNN训练分别占用磁盘、CPU、GPU资源，且均耗时较大。因此，我们引入流水线，使得磁盘、CPU、GPU资源可以同时得到利用，提升整体性能。

CNN数据并行和模型并行框架已在图像识别应用中初见成效。针对Hinton在2012年获得ImageNet竞赛冠军用的网络 [2]，我们取得了两卡模型并行1.71倍加速比，4 GPU卡数据并行加模型并行时比单卡2.52倍的加速比。通过CNN并行框架的模型并行，单个GPU上CNN网络占用的GPU显存从3.99 GB减少到2.15 GB，使得可以训练更大规模的图像分类模型。通过模型并行获得ImageNet 2012数据集87%的top5准确率。CNN并行训练框架在微信图像业务中有众多潜在应用都已尝试接入本框架。同时在广点通广告推荐也开始应用探索。

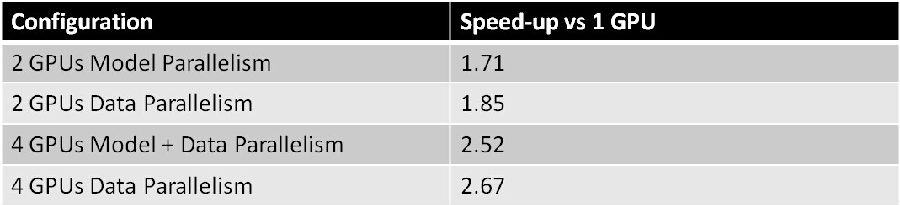


            图4：CNN GPU框架对Hinton的网络[2]在ImageNet 2012的并行加速性能

DNN CPU集群框架实现基于CPU集群的数据并行和模型并行，其总体架构如下图所示：

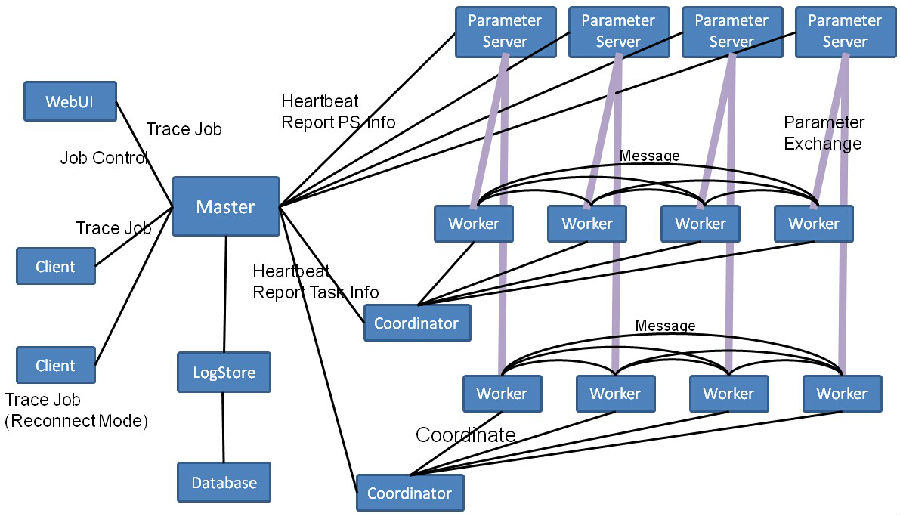


                                           图5：DNN CPU集群框架总体架构

DNN CPU集群框架提供Vertex+Message的API，实现BulkSynchronous Parallel（BSP）模式。每次DNN训练作业作为一个DNN Job，其执行包含多个迭代，用户通过Client工具提交DNN Job。DNN Master负责任务调度，将训练数据分发到不同的Worker Group进行训练，并完成任务的failover等，以支持数据并行。Master将DNN Job的状态变化通过LogStore系统同步到数据库，便于从WebUI展示全部作业状态。此外，Master负责Counter的收集，并在WebUI上可视化展示。每个Worker Group中有1个Coordinator协调全部Worker完成模型并行，而Worker完成具体训练任务。模型通过参数服务器Parameter Server划分，并可靠存储在分布式文件系统中。在微信语音业务中，证明DNN CPU集群可取得与GPU相当的训练结果，而且CPU集群训练和GPU训练框架有互补性。

经过一年多的沉淀，深度学习在腾讯产生了落地的成果。腾讯深度学习平台Mariana逐步成型，形成了3个框架，包括：DNN GPU数据并行框架，CNN GPU模型并行和数据并行框架，以及DNN CPU集群模型并行和数据并行框架。Mariana通过数据并行和模型并行，解决了深度学习耗时冗长的问题，成为深度学习研究的有效助力。Mariana通过模型并行，有效支持了大模型。Mariana完成众多工作增强易用性，简化深度学习实验，大幅节约算法开发时间。Mariana的DNN GPU数据并行框架，针对微信语音识别应用，在单机6 GPU卡配置下获得相比单卡4.6倍的加速，可在数日内完成数十亿高维度训练样本的DNN模型训练，同时模型字错率降低10%以上。Mariana的CNN模型并行和数据并行框架，针对ImageNet图像分类问题，在单机4 GPU卡配置下，获得了相比单卡2.52倍的加速，并支持更大模型，在ImageNet 2012数据集中获得了87%的top5准确率。目前，腾讯深度学习平台Mariana已支持了微信语音识别的语音输入法、语音开放平台、长按语音消息转文本等产品，在微信图像识别中开始应用。此外，在广告推荐及个性化推荐等领域，也正在积极探索和实验中。

**参考文献：**

1.Hinton, G. E. andSalakhutdinov, R. R. (2006) Reducing the dimensionality of data with neuralnetworks. Science, Vol. 313. no.5786, pp. 504 - 507, 28 July 2006.

2. Alex Krizhevsky, IlyaSutskever. and Geoffrey E. Hinton. (2012) ImageNet Classification with DeepConvolutional Neural Networks. in Advancesin Neural Information Processing 25 (NIPS 2012), MIT Press, Cambridge, MA.

3. Jeffrey Dean, Greg S. Corrado,Rajat Monga, et al, and Andrew Y. Ng (2012) Large Scale Distributed DeepNetworks. in Advances in NeuralInformation Processing 25 (NIPS 2012), MIT Press, Cambridge, MA.

# 【深度学习系列2】Mariana DNN多GPU数据并行框架

mary  发表于 2014.8.6     2047浏览    0讨论

 本文是腾讯深度学习系列文章的第二篇，聚焦于腾讯深度学习平台Mariana中深度神经网络DNN的多GPU数据并行框架。

深度神经网络（Deep Neural Networks, 简称DNN）是近年来机器学习领域中的研究热点[1][2]，产生了广泛的应用。DNN具有深层结构、数千万参数需要学习，导致训练非常耗时。GPU有强大的计算能力，适合于加速深度神经网络训练。DNN的单机多GPU数据并行框架是Mariana的一部分，Mariana技术团队实现了数据并行技术加速DNN训练，提供公用算法简化实验过程。对微信语音识别应用，在模型收敛速度和模型性能上都取得了有效提升——相比单GPU 4.6倍加速比，数十亿样本的训练数天收敛，测试集字错率降低约10%。目前基于此框架训练的模型成功上线到微信语音输入法、微信语音开放平台和微信语音消息转文字。

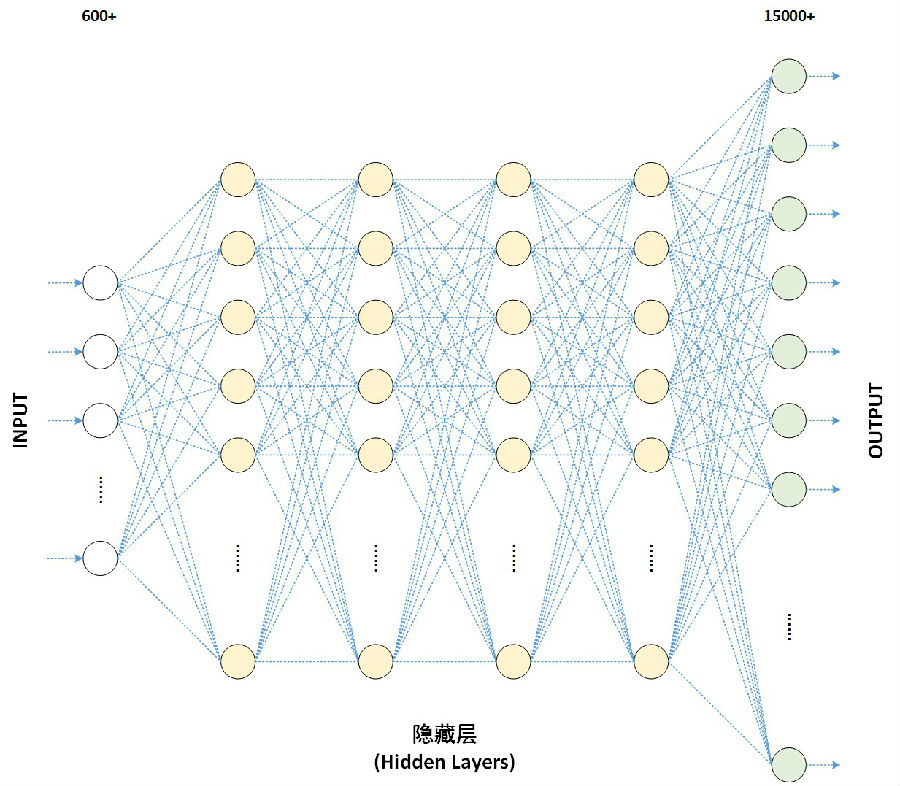
## DNN数据并行导论

### 典型应用分析：语音识别

语音识别是深度神经网络获得成功的一个应用范例。

语音识别应用，简单来说由声学模型建模，语言模型建模以及解码三部分构成。其中声学模型用来模拟发音的概率分布，语言模型用来模拟词语之间的关联关系，而解码阶段就是利用上述两个模型，将声音转化为文本。神经网络具有模拟任何分布的能力，深度神经网络比浅层神经网络表达能力更强，它模拟了人脑的深层结构，能够更准确地“理解”事物的特征。因此相较于其他方法，深度神经网络可以更为准确地模拟声学模型和语言模型。这里将深度神经网络应用于语音识别中的声学模型建模。

图 1揭示了一个深度神经网络表达的具有4个隐藏层的声学模型结构，训练这样的具有深层结构的模型，其优势在于逐层的特征学习过程是由初级到高级的过程：第一层学习初始特征（例如最基本的发声），在后面的层次逐渐学习到越来越高级的特征（例如声音的组合），直到通过模型的输出层来有效地识别声音。

  
            图1深层、稠密、包含4隐藏层的声学模型

### 现有的加速方法

Mariana技术团队在语音识别研究中率先引入了GPU技术用于DNN训练，获得了良好的成果，相比单台CPU服务器达到千倍加速比。随着训练数据集扩充、模型复杂度增加，即使采用GPU加速，在实验过程中也存在着严重的性能不足，往往需要数周时间才能达到模型的收敛，不能满足对于训练大规模网络、开展更多试验的需求。目前服务器上安装多个GPU卡已经非常普遍，在通用计算领域使用多GPU并行加速技术[3]扩展计算密集型应用程序的并行性、提高程序性能也是越来越热门的发展方向。

由于语音业务中DNN模型采用多层全连接的网络结构，Mariana技术团队在单机多GPU模型并行的实践中发现拆分其模型存在较大的额外开销，无论采用普通模型拆分还是流式控制，扩展性有限：相比GPU的计算能力，如果模型参数量不能匹配，模型并行不能有效地利用多个高计算能力的GPU卡，表现为使用2GPU时已有较好的性能提升，但使用更多GPU却无法取得更好效果。

Mariana技术团队考虑到上述问题，在Mariana的DNN多GPU并行训练框架中，选择了数据并行的技术路线，完成了升级版的单机多GPU数据并行版本。

本文描述了多GPU加速深度神经网络训练系统的数据并行实现方法及其性能优化，依托多GPU的强大协同并行计算能力，结合数据并行特点，实现快速高效的深度神经网络训练。

### 框架设计目标

由于训练深层网络使用的训练数据规模庞大、计算开销大，从而训练过程收敛难，训练用时久，通过开发多GPU数据并行版本期望达到下述目标：充分利用单机多GPU计算资源和DNN的数据并行特性，加速模型训练过程；提供默认的深度学习算法实现，以帮助提升语音识别准确率；提供更好的易用性支持，方便进行模型训练。

上述目标完成后，系统可以支持目标模型的快速训练，达到更好的收敛效果。另外，作为Mariana中的DNN数据并行框架，其训练算法具有一定的通用性，可通过简单配置快速适用于其他DNN应用场景中。

### 挑战

在语音识别应用中，深度模型包含数万神经元，采用全连接结构，具有数千万参数需要训练，计算量大；需要海量语音数据进行训练，在避免过拟合的前提下提高模型准确率、提高模型泛化能力。因此，如何充分利用计算资源实施数据并行以加速训练是框架设计首要解决的问题。

数据并行中参数交换容易成为整体性能瓶颈，因为在参数交换时需要将各个Model Replica在本次mini-batch训练中得到的梯度合并后用于更新参数服务器上的模型，之后再将最新模型推送到各个数据并行单元上用于下一轮计算。如何解决参数交换瓶颈以提高性能，成为并行方法设计的重中之重。

自适应学习率算法[4]对于随机梯度下降（Stochastic Gradient Descent, 简称SGD）过程有利于加速收敛并获得更好的结果。但是它需要对模型中每个参数存储helper\_sum并频繁更新对应的学习率，影响性能。因此，如何将其有效地应用到数据并行框架中，是我们所需要解决的另一个重点问题。

## 系统概述

如何数据并行？

数据并行：指对源数据集合中的元素同时（即并行）执行相同操作的情况。在数据并行操作中，将对源数据集合进行分区，以便多个并行处理单元能够同时对不同的子数据集合进行操作。

图2揭示了从单GPU训练到多GPU数据并行训练的变化，主要在于：单GPU每个周期训练1个mini-batch，mini-batch训练结束后即顺带在GPU显存中完成了更新模型参数的操作；在多个GPU同时训练多组mini-batch数据，每个数据并行组（Worker Group）充分利用本组上独占的GPU资源，在mini-batch训练结束时还需要一个将参数从各个GPU上交换并更新的过程，最终使得每个GPU上都持有最新模型副本，继续进行下一个周期训练过程。

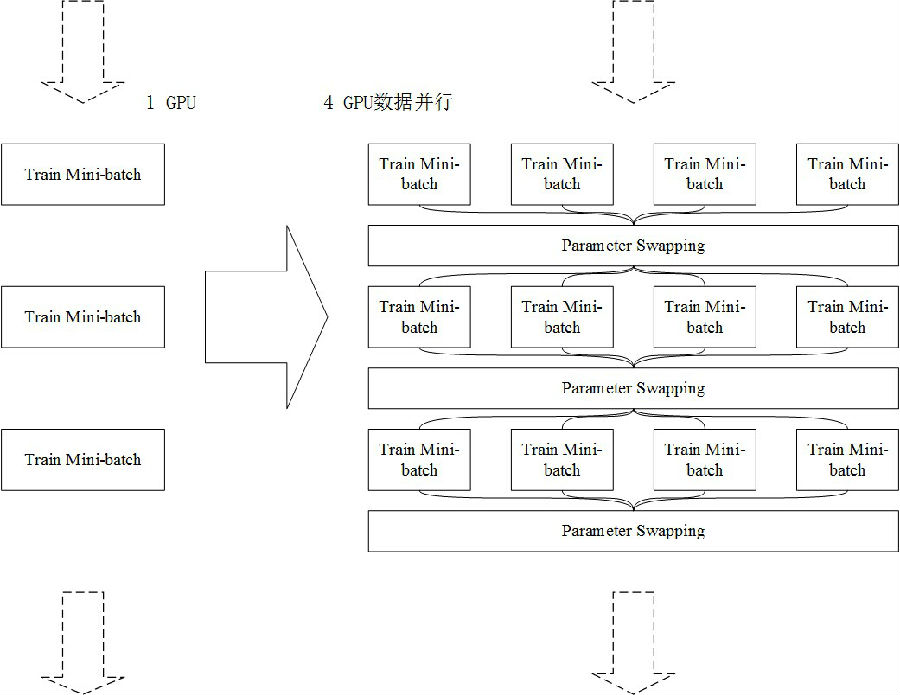


                      图2从单GPU训练到多GPU数据并行训练的概要视图

多GPU并行系统从功能上划分为用于读取和分发数据的Training Data Dispatcher和用于做数据并行训练的GPU Worker Group，如图3所示。训练数据从磁盘文件读取到CPU主存再拷贝到GPU显存，故此设计在各Worker Group计算每batch数据时，由Training Data Dispatcher从文件中读取并分发下一batch数据，以达到用计算时间掩盖I/O时间的设计目标。

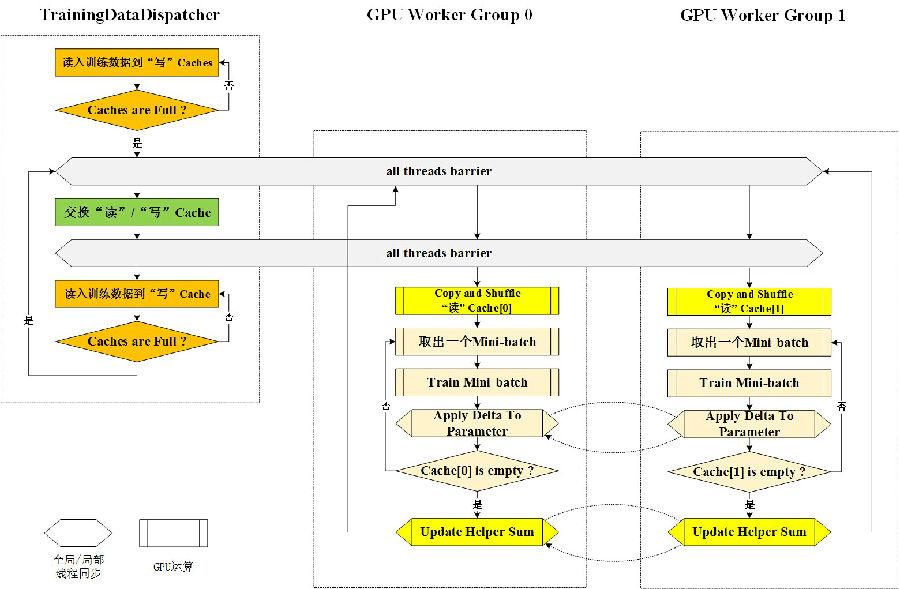


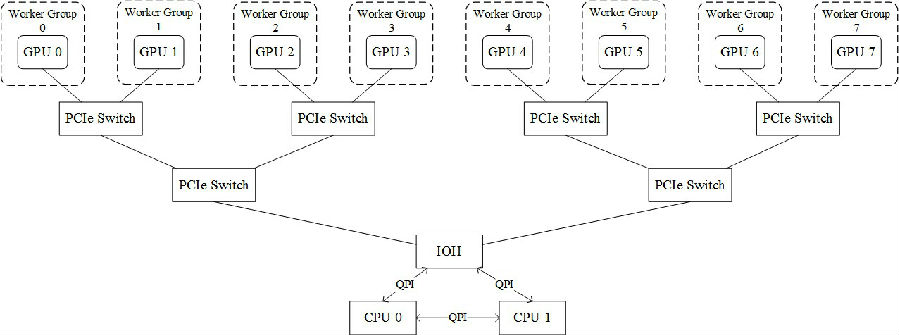
                              图3 2 GPU数据并行系统框架示意

## GPU Worker Group: 数据并行的承载体

数据并行以Worker Group为基本组织形式，调度来源于CPU，计算资源来源于GPU卡。GPU卡通常被看成是加速卡或协处理器卡，必须在基于CPU的主机上下文中被调用来做计算，因此1个CPU线程绑定1个Worker Group中的1张GPU卡能够发挥多GPU的并行效能。

在实际生产环境中，安装多GPU服务器的硬件体系结构如图4所示，示例中揭示了一个8 GPU节点服务器的硬件配置，每两个GPU Slot安装在一个GPU专用PCI槽位上再通过PCIe Switch将GPU Slot 0,1,2,3安装在一颗CPU上，GPU Slot 4,5,6,7安装在另一颗CPU上，两颗CPU通过IOH（Input Output Hub）连接。

在数据并行的场景下，将每颗GPU绑定一个Worker Group，即形成8路数据并行的并行化训练配置。

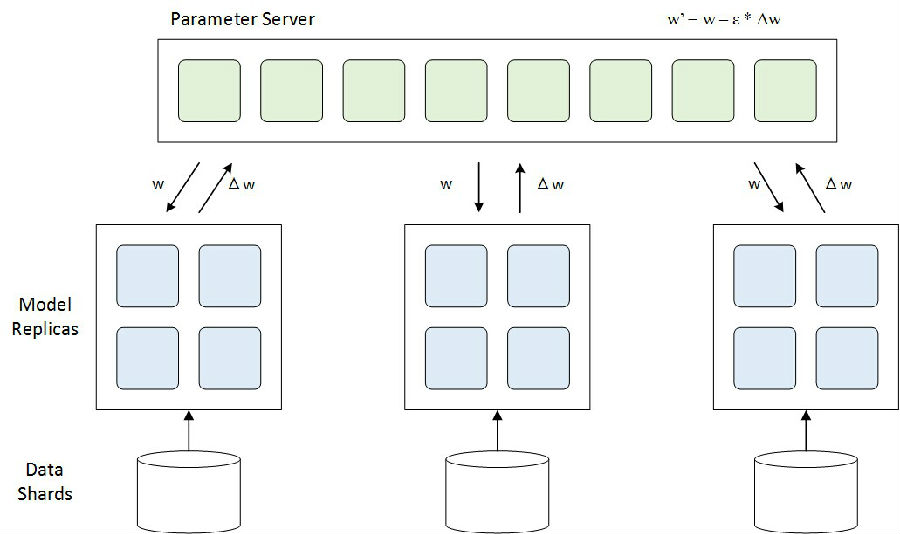


## 数据并行中参数交换的线形拓扑

### 数据并行的关键问题：参数交换是瓶颈

使用基于mini-batch的随机梯度下降算法时，每计算完成1个mini-batch后需要将本次计算得到梯度（图中记为Δw）应用到相应参数（图中记为w）上，朴素的参数更新公式为：

公式中，i是参数更新的次数，weight指代某个参数，ε是学习速率，Δw是本次BP计算出的梯度。

  
                           图5 Model Replicas参数交换过程示意

在数据并行的实现中，影响性能的瓶颈在于多GPU间的参数交换。这是因为按照参数更新公式来看，需要将所有模型replica的梯度交换到参数服务器并更新到相应参数上，而参数服务器的带宽成为瓶颈。

最优化的参数交换解决方案应具有的特征：尽量减少总的通信量；尽量减少所用交换周期数；每个周期用满PCIe总线通信带宽；GPU间数据传输没有等待；可扩展性强，适用于不同GPU数目的场景。

### 常见的解法

解决这一问题，直观的想法是依靠多GPU之间点到点的数据传输，即在每两个GPU之间建立数据传输连接，交换各自的ΔW。但是这样的解决方案提升的性能有限，主要在于任一GPU上计算出的ΔW在PCIe总线上需要多次传输才能更新到所有GPU的参数模型上，过多占用和消耗PCIe总线带宽。而随着GPU数增多，交换时间随之呈平方量级增加，这无疑是不适合的。

另一种常见思路是采用分治法+树形结构，为此可设计树形拓扑：模型2分片，奇/偶Worker Group负责归并模型不同分片，Worker Group间总是两两归并；当最终将2个分片归并结束后，开始反向的分发过程。但在多GPU场景下这种也存在着局限，主要在于每个归并周期后，总有一半GPU不再参与之后的归并过程，闲置了其计算能力和所在节点上的通信带宽。

### 当前最优解：线形拓扑

我们针对多GPU场景设计了参数交换的线形拓扑：

线形拓扑从空间上来看，如图6所示，将存储参数的矩阵、存储梯度的矩阵等量划分了Partition，Partition数量是Worker Group数的一半。令持有某一Partition最终累加结果的Worker Group（一般为Group ID为奇数的GPU），称之为该Partition的Partition Owner。对该Partition的更新参数操作会在该Partition Owner上执行，同时该Partition最新参数的分发过程也由该Partition Owner发起。

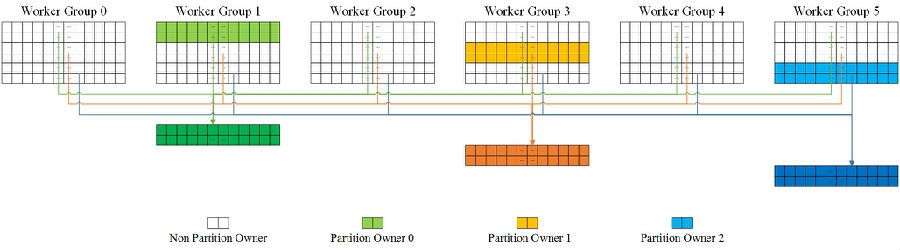


                     图6 Partition拆分及Partition Owner分布示意

线形拓扑从时间上来看，如图7所示，将参数交换过程划分了周期，每个周期内将所有Partition从上游Worker Group向下游Worker Group推送一次，推送过程包括从上游Worker Group拷贝Partition数据和本地Merge两步操作，推送阶段的最后1个周期推送该Partition到Partition Owner上以使其持有参数的最新副本。推送所用周期数为数据并行组数-1。

之后参数分发过程，亦占用与推送相同的周期数分发Partition Owner上的最新副本给所有Worker Group。

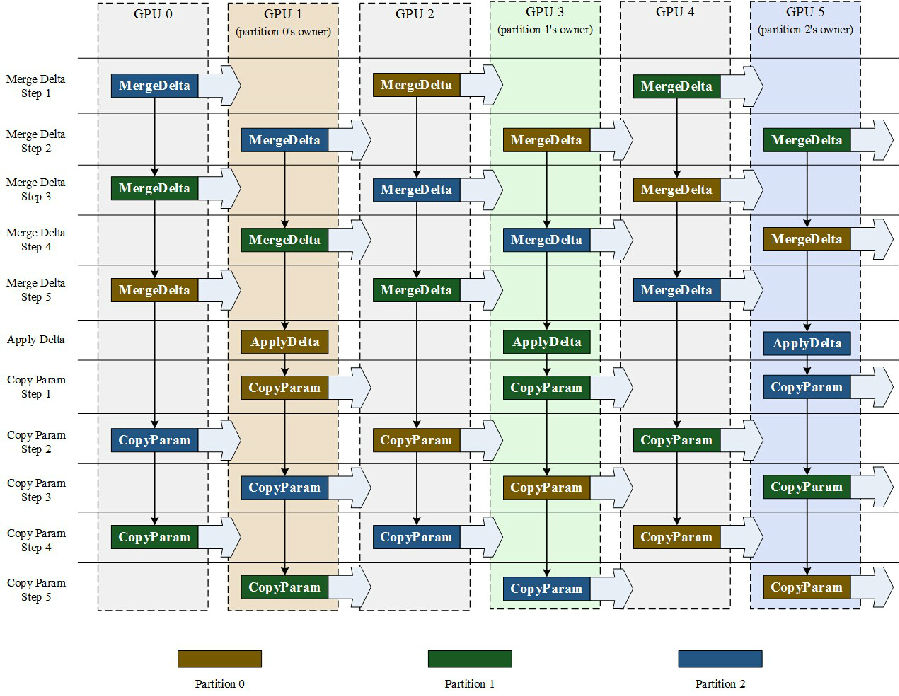


                                      图7线形拓扑参数交换全部过程

我们比较容易给出线形拓扑的性能模型。PCIe总线带宽一定（设为W），取数据并行的Worker Group数目为2N，模型大小为M。在采用点到点全量参数交换的情景下，每两个Worker Group均需要进行对等的数据交换，因此数据传输的最短周期为2(2N-1)次，每次在总线上传输数据量为M，故推导出参数交换用时2(2N-1)(M/W)。在采用线形拓扑的情景下，在每个组号为单数（1,3,5,…）的组上维护着整个模型参数相应的1/N，按照线形拓扑，单次数据交换中只传输了（1/N） M的数据量，参数归并周期为2N-1，参数分发周期也为2N-1，因此推导出参数交换用时为2(2N-1)(M/(N\*W))，仅是点到点用时的1/N（N=1,2,3,…）。可见，数据并行的并发量越大，线形拓扑的性能收益效果越显著。

表1线形拓扑的性能分析



线形拓扑可以很容易地扩展到偶数个数据并行组的参数交换，其收集（推送）用时随数据并行组数增长而缓慢增长，且具有上界——2T0，这说明线形拓扑非常适用于更多GPU做数据并行。

## 多GPU数据并行的自适应学习率（AdaGrad）算法

应用自适应学习率算法，对第i个位置上参数的学习率更新，需要借助helper\_sum，其定义如下：

在数据并行情景下，helper\_sum的跨组更新规则与的数据并行参数交换过程中的推送阶段完全相同，由于梯度是在Partition Owner上最先持有的完全累积结果，可以根据Adagrad参数更新公式Apply到参数上，而这时只用到了Partition Owner上的helper\_sum，因此只需要在Partition Owner上持有其“Own”的Partition的最新helper\_sum即可，其它非Partition Owner在推送之后即需要将helper\_sum清零，以便为下一次计算重新累积所用。由于helper\_sum不需要分发，则交换helper\_sum的过程只需要一次推送阶段即可完成。

结合Adagrad算法实际应用场景：即helper\_sum的累积其实是对每个学习率调整的量化过程，其本身具有一定的近似性。基于性能考量，实际项目中使用一种近似的helper\_sum收集：在每个mini-batch结束后所有partition上做本地累积helper\_sum计算，而在M个mini-batch结束后才将其推送到所属的Partition Owner完成最终累积。这样兼顾了算法效果和训练性能。

## 在语音识别上的应用

### 微信语音识别产品

微信中主要有三款语音识别产品：微信语音输入法、语音开放平台和语音消息转文字。多GPU数据并行框架训练的模型正在由这些产品使用。



微信语音输入法



语音消息转文字



语音开放平台  
图8微信语音识别产品

### 模型训练实验性能

实验环境为一台搭载8核心Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz的服务器，内存为48GB，服务器安装了6块NVIDIA Tesla K20c GPU，单GPU显存大小为4.8GB。

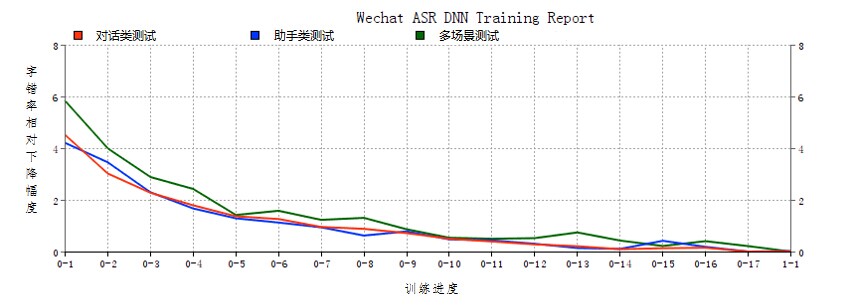
训练同样的语音目标模型，相比于单GPU，使用6 GPU数据并行获得了4.6倍加速比。

### 模型准确率

采用6 GPU数据并行，使用数十亿语音训练样本，在训练目标模型的实验中，数个迭代时取得了在测试集上的最佳模型，取得了10%的字错率降低。同时相比模型并行训练，基于数据并行的Mariana可在数天内完成DNN模型训练。

### 易用性

框架提供良好的易用性支持，主要在于实验者不需要过多干预训练过程即可完成训练，实时获知训练进度和模型准确率等信息。框架主要的易用性支持包括：重要配置参数在运行时可手工改变，可自动衰减；按指定频率输出模型snapshot并进行自动化测试；训练中断点恢复，可从任意模型snapshot继续训练；微信监控帐号自动向实验者推送测试结果。

  
                                    图9微信监控帐号自动推送测试结果

## 结论

本文描述了腾讯深度学习平台Mariana中深度神经网络DNN的多GPU数据并行框架，通过多个Worker Group实现了数据并行，设计线形拓扑解决了参数交换性能瓶颈，实现了数据并行场景下的近似自适应学习率算法。此框架有效支持了微信语音识别，成功提升了微信语音识别的模型训练速度，并且模型字错率降低10%，模型全量上线到微信语音输入法、微信语音开放平台和微信语音消息转文字。

## 参考文献

[1]    Hinton G.E., Osindero S. and Teh Y. (2006) A Fast Learning Algorithm for Deep Belief Nets. Neural Computation, 2006.

[2]    Hinton, G. E. and Salakhutdinov, R. R. (2006) Reducing the Dimensionality of Data with Neural Networks. Science, Vol. 313, No. 5786, pp. 504-507, 28 July 2006.

[3]    David B. Kirk, Wen-mei Hwu. 大规模处理器编程实战(第2版). 清华大学出版社, 2013.

[4]    J. Duchi, E. Hazan and Y. Singer. (2011) Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine learning Research, 12(2011): 2121-2159, 2011.

[5]    Jeffrey Dean, Greg S. Corrado, Rajat Monga, et al, and Andrew Y. Ng. (2012) Large Scale Distributed Deep Networks. In Advances in Neural Information Processing 25 (NIPS 2012), MIT Press, Cambridge, MA.

# 【深度学习系列3】 Mariana CNN并行框架与图像识别

mary  发表于 2014.8.13     2443浏览    1讨论

本文是腾讯深度学习系列文章的第三篇，聚焦于腾讯深度学习平台Mariana中深度卷积神经网络Deep CNNs的多GPU模型并行和数据并行框架。

将深度卷积神经网络（Convolutional Neural Networks, 简称CNNs）用于图像识别在研究领域吸引着越来越多目光。由于卷积神经网络结构非常适合模型并行的训练，因此以模型并行+数据并行的方式来加速Deep CNNs训练，可预期取得较大收获。Deep CNNs的单机多GPU模型并行和数据并行框架是Mariana的一部分，Mariana技术团队实现了模型并行和数据并行技术加速Deep CNNs训练，证实模型拆分对减少单GPU上显存占用有效，并且在加速比指标上得到显著收益，同时可以以较快速度训练更大的深度卷积神经网络，提升模型准确率。

## CNNs模型并行导论

### 典型应用分析：图像识别

图像识别是深度卷积神经网络获得成功的一个典型应用范例。

图1揭示了一个具有5个卷积层和3个全连接层的深度卷积神经网络，该模型可应用于图像分类。

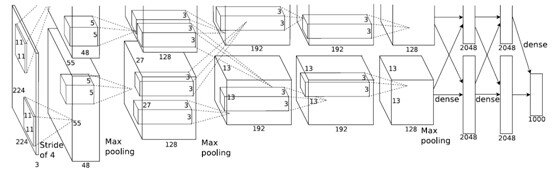


图1 ImageNet深度卷积神经网络模型

使用GPU训练深度卷积神经网络可取得良好的效果[1][2]，自2012年使用Deep CNNs模型在ImageNet图像分类挑战中取得突破性成绩，2013年的最佳分类结果也是由Deep CNNs模型取得。基于此，Mariana技术团队期望引入Deep CNNs来解决或优化图像分类问题和图像特征提取问题，以提升在相应用例场景中的效果。

### 现有系统的问题

在将CNN应用于图像相关领域的算法研究以及CNN训练平台搭建的实践过程中，受限于单个GPU上的显存大小（例如：服务器采购的显卡Tesla K20c可用显存为4.8GB，ImageNet 2012论文[1]中用到的网络占用显存大约3.9GB），在尝试调整参数和网络规模的实验中，往往难以存储下更大规模的深度卷积神经网络模型，使得包含较多参数的网络不能在单GPU上训练，需要通过多GPU模型并行技术，拆分模型到多个GPU上存储和训练来解决。

随着训练数据集扩充、模型复杂度增加，即使采用GPU加速，在实验过程中也存在着严重的性能不足，往往需要十余天时间才能达到模型的收敛，不能满足对于训练大规模网络、开展更多试验的需求。

考虑到上述问题，在Mariana的Deep CNNs多GPU并行训练框架中，通过设计模型拆分方法、模型并行执行引擎和优化访存性能的Transfer Layer，并吸收在数据并行方面设计经验，实现了多GPU加速的模型并行和数据并行版本。

本文描述多GPU加速深度卷积神经网络训练系统的模型并行和数据并行实现方法及其性能优化，依托多GPU的强大协同并行计算能力，结合目标Deep CNNs模型在训练中的并行特点，实现快速高效的深度卷积神经网络训练。

### 框架设计目标

多GPU模型并行+数据并行期望达到下述目标：充分利用Deep CNNs模型的可并行特点，结合SGD（Stochastic Gradient Descent，随机梯度下降）训练的数据并行特性，加速模型训练过程；突破显存大小限制，使得训练超过单GPU显存的模型成为可能，并预期通过训练更复杂的网络来获得更好的模型效果。

上述目标完成后，系统可以更快地训练图1中目标Deep CNNs模型。模型拆分到不同GPU上可减少对单GPU显存占用，适用于训练更深层次、更多参数的卷积神经网络。

### 挑战

在图像识别应用中，深度卷积神经网络模型的卷积层计算量大，全连接层参数多。因此，如何划分计算资源，通过模型并行和数据并行两个数据/计算组织层次上来加速训练是框架设计首要解决的问题。

图像作为输入数据，其数据量庞大，且需要预处理过程，因此在Batch训练时磁盘I/O、数据预处理工作也要消耗一定时间。经典的用计算时间掩盖I/O时间的方法是引入流水线，因此如何设计一套有效的流水线方法来掩盖I/O时间和CPU处理时间，以使得整体耗时只取决于实际GPU训练时间，是一个重要问题。

模型并行是将一个完整Deep CNNs网络的计算拆分到多个GPU上来执行而采取的并行手段，结合并行资源对模型各并行部分进行合理调度以达到模型并行加速效果是实现模型并行的关键步骤。

多GPU系统通过UVA（Unified Virtual Address，统一虚拟地址）技术，允许一颗GPU在kernel计算时访问其他GPU的设备内存（即显存），但由于远程设备存储访问速度远远低于本地存储访问速度，实际性能不佳。因此在跨GPU的邻接层数据访问时，需要关注如何高效利用设备间数据拷贝，使所有计算数据本地化。

## 系统概述

### 如何模型并行？

模型并行是：适当拆分模型到不同的计算单元上利用任务可并行性达到整个模型在计算过程中并行化效果。

如图2所示，揭示了从单GPU训练到多GPU模型并行训练的相异之处，主要在于：在使用单GPU训练的场景下，模型不进行拆分，GPU显存上存储整个模型；模型并行的场景下，将模型拆分到多个GPU上存储，因此在训练过程中每个GPU上实际只负责训练模型的一部分，通过执行引擎的调度在一个WorkerGroup内完成对整个模型的训练。

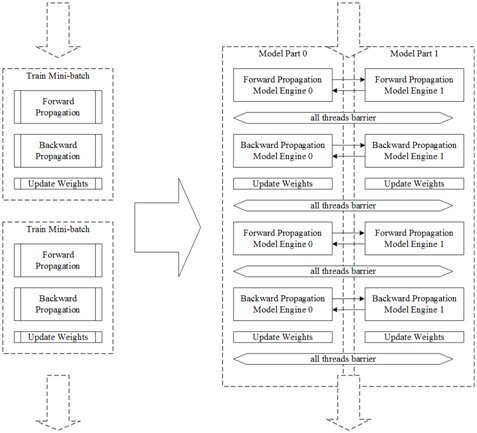


图2从单GPU训练到多GPU模型并行训练的概要视图

多GPU并行系统从功能上划分为用于读取和分发数据的Training Data Dispatcher和用于做模型并行训练的GPU Worker，如图3所示。训练数据从磁盘文件读取到CPU主存再拷贝到GPU显存，故此设计在各Worker计算每batch数据时，由Training Data Dispatcher从文件中读取并分发下一batch数据，以达到用计算时间掩盖I/O时间的设计目标。

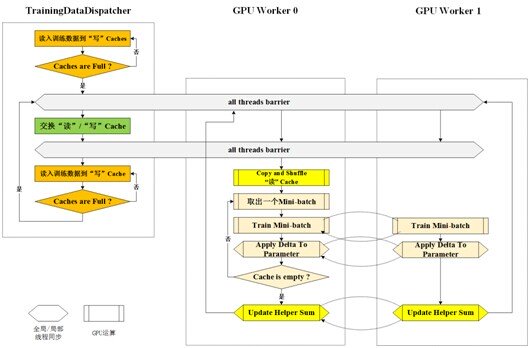


图3 2 GPU模型并行系统框架示意

## 训练数据处理的并行加速

基于mini-batch的训练，现有技术方案在训练深度卷积神经网络时，每次从数据文件中读入和处理1个batch数据，在GPU计算某一batch时由CPU预读取和预处理下一batch。

但是随着训练集图片像素数增大，读取和处理时间随之增加，由于采用多GPU技术加速了单个batch计算时间，数据处理的性能问题随之而来，需要减少数据处理的用时，以使最终加速效果取决于计算用时。

如图4所示，总体看来，在深度卷积神经网络训练过程中始终是在执行一条三阶段并行的流水线：计算本次batch数据——处理下次batch数据——读入再下次batch数据。

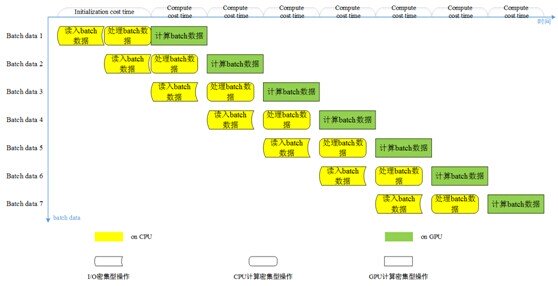
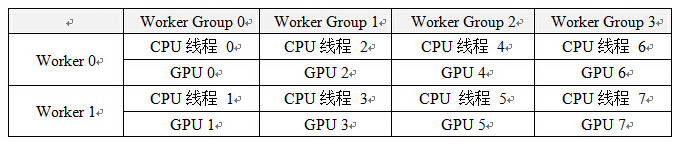


图4数据处理和计算流水线

## GPU Worker: 模型并行的承载体

数据并行以划分Worker Group为基本组织形式，模型并行以在Worker Group内划分Worker为基本组织形式，并行训练的调度资源来源于CPU线程，计算资源来源于GPU卡。由于GPU卡通常意义上被看成是一种加速卡或协处理器卡，必须在基于CPU的主机上下文中被调用来做计算，因此遵循1个CPU线程绑定1张GPU卡能够发挥多GPU共同参与计算时的并行性效能。

表1 4 GPU模型并行+数据并行CPU线程、GPU与Worker Group、Worker绑定关系



在实际生产环境中，安装多GPU服务器的硬件体系结构如图5所示，示例中揭示了一个8 GPU节点服务器的硬件配置，每两个GPU Slot连接在一个GPU专用PCI槽位上再通过PCIe Switch将GPU Slot 0,1,2,3连接在一颗CPU上，GPU Slot 4,5,6,7连接在另一颗CPU上，两颗CPU通过IOH（Input Output Hub）连接。

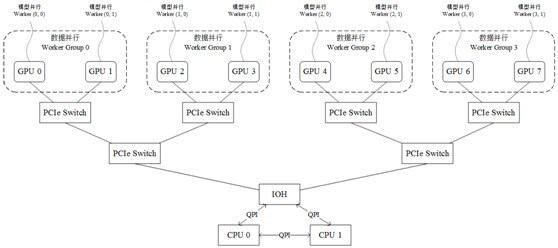


图5硬件体系结构

## CNNs网络的模型划分

### 基本模型划分方法

模型并行的来源是Deep CNNs网络只在特定层（如输入层、全连接层）与其他层有全面的连接，而其他较为独立的直线连接关系即可作为模型的可并行部分。将模型的可并行部分拆分到多个GPU上，同时利用多个GPU的计算能力各执行子模型的计算，可以大大加快模型的单次前向-后向训练时间。

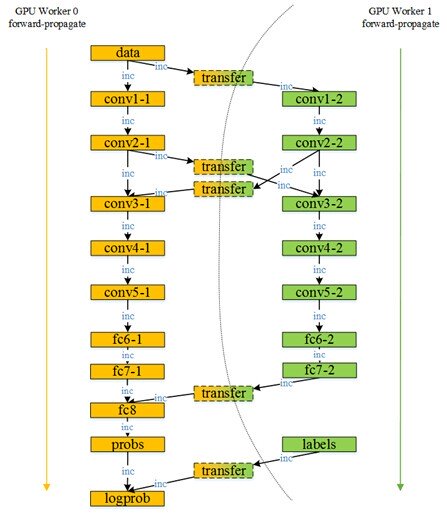


图6模型并行中的模型划分方案示意

Deep CNNs网络的层次模型实际上是一张有向无环图（DAG图），分配到每个模型并行Worker上的层集合，是有向无环图的拓扑排序子集，所有子集组成整个网络的1组模型。

### “十字形”模型划分方法

考虑极端情景：需要训练超大规模Deep CNNs模型，或者使用计算能力相对较强、显存较小（一般在1GB~3GB）的桌面级GeForce系列GPU，则利用模型本身的并行性这种基本的模型划分方法将不再适用。需要将模型再做拆分以保证单个GPU都能存储下对应的子模型。

如图7所示，描述了将模型按“十字形”划分到4 Worker上训练的情景，不仅拆分了模型的可并行部分，也虽然这样的划分在Worker 0和Worker2之间，Worker 1和Worker 3之间达到并行加速效果，却能使得整个模型得以存储在4 GPU上。这种模型划分方法能够适用于训练超大规模网络等特殊模型的需求。

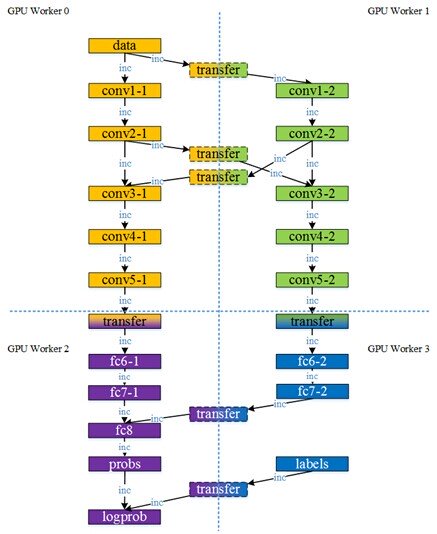


图7 “十字形”模型划分方案示意

## CNNs网络的模型并行工作引擎

每个模型并行Worker上以一个模型并行执行引擎负责调度本Worker上子模型的执行过程。执行引擎控制所有Worker上的子模型完成前向和后向计算，各自对子模型完成参数更新后，到达主线程同步点，开始下一mini-batch训练。

多GPU模型并行和数据并行的Deep CNNs模型replicas及划分结构如图8所示，在使用4 GPU的场景下，划分了2组Worker Group用于数据并行；每个Worker Group内划分2个Worker用于模型并行。

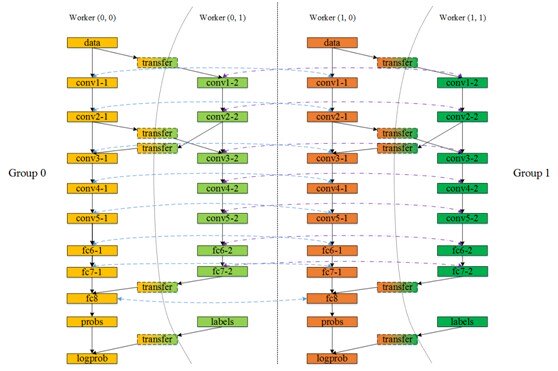


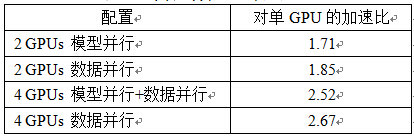
图8对ImageNet网络的模型并行和数据并行划分

## 在图像识别上的应用

### 模型训练实验性能

实验环境为一台搭载8核心Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz的服务器，内存为48GB，服务器安装了4块NVIDIA Tesla K20c GPU，单GPU显存大小为4.8GB。  
训练同样的Deep CNNs模型，相比于单GPU，使用多GPU结合不同并行模式的加速效果如下表所示：

表2不同并行配置的加速比



### 模型收敛性

对于目标Deep CNNs模型，在单GPU训练时（对照实验）显存占用量为3.99GB；使用2 GPU模型并行训练时单个GPU上显存占用量为2.15GB，并且在训练相同迭代时训练集、测试集错误率效果都与对照实验完全相同；抽样比照参数一致性，被抽样的参数（同对照实验相比）也都是一样。  
尝试更改Deep CNNs模型，训练一个更大的网络，增加滤波器数目，减小步长，增加中间卷积层feature map数目，训练时所需显存将达到9GB以上，使用单个Tesla K20c GPU（4.8GB显存）无法开展训练实验；而多GPU模型并行训练实验中该模型的错误率对比图1模型降低2%。

### 效果展示

图9为图像标签识别的示例，通过对两千多类物体的图像进行训练，可实现对常见物体的自动识别。





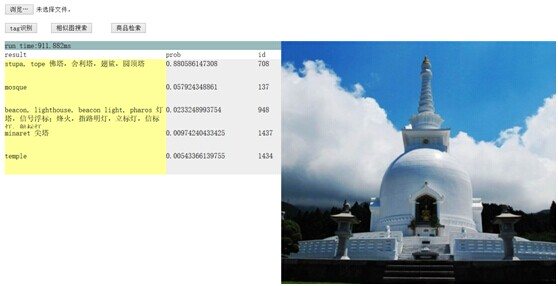


图9应用效果展示

## 结论与展望

本文描述了腾讯深度学习平台Mariana中深度卷积神经网络Deep CNNs的多GPU模型并行和数据并行框架，通过多个Worker Group实现了数据并行，同一Worker Group内多个Worker实现模型并行。框架中实现了三阶段并行流水线掩盖I/O、CPU处理时间；设计并实现了模型并行引擎，提升了模型并行计算执行效率；通过Transfer Layer解决了数据存储访问效率问题。此框架显著提升了深度卷积神经网络训练速度，解决了当前硬件条件下训练大模型的难题。

深度卷积神经网络有着广泛的应用场景：在图像应用方面，Deep CNNs可应用于相似图片检索、图片的自动标注和人脸识别等。在广告图片特征提取方面，考虑Deep CNNs能够很好地学习到图像特征，我们尝试将其用于广告点击率预估（Click-Through Rate Prediction,pCTR）模型中。

## 参考文献

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. (2012) ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing 25 (NIPS 2012), MIT Press, Cambridge, MA.

[2] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Gerald Penn. (2012) Applying Convolutional Neural Networks Concepts to Hybrid NN-HMM Model for Speech Recognition. ICASSP’12, 2012.

[3] Jeffrey Dean, Greg S. Corrado, Rajat Monga, et al, and Andrew Y. Ng. (2012) Large Scale Distributed Deep Networks. In Advances in Neural Information Processing 25 (NIPS 2012), MIT Press, Cambridge, MA.

[4] Omry Yadan, Keith Adams, Yaniv Taigman, Marc’Aurelio Ranzato. (2014) Multi-GPU Training of ConvNets. arXiv: 1312.5853v4 [cs.LG], Feb. 18th, 2014.

# 【深度学习系列4】深度学习及并行化实现概述

mary  发表于 2014.8.18     3503浏览    4讨论

**摘要：** 深度学习可以完成需要高度抽象特征的人工智能任务，如语音识别、图像识别和检索、自然语言理解等。深层模型是包含多个隐藏层的人工神经网络，多层非线性结构使其具备强大的特征表达能力和对复杂任务建模能力。训练深层模型是长期以来的难题，近年来以层次化、逐层初始化为代表的一系列方法的提出给训练深层模型带来了希望，并在多个应用领域获得了成功。深层模型的并行化框架和训练加速方法是深度学习走向实用的重要基石，已有多个针对不同深度模型的开源实现，Google、Facebook、百度、腾讯等公司也实现了各自的并行化框架。深度学习是目前最接近人脑的智能学习方法，深度学习引爆的这场革命，将人工智能带上了一个新的台阶，将对一大批产品和服务产生深远影响。

## 深度学习的革命

人工智能(Artificial Intelligence)，试图理解智能的实质，并制造出能以人类智能相似的方式做出反应的智能机器。如果说机器是人类手的延伸、交通工具是人类腿的延伸，那么人工智能就是人类大脑的延伸，甚至可以帮助人类自我进化，超越自我。人工智能也是计算机领域最前沿和最具神秘色彩的学科，科学家希望制造出代替人类思考的智能机器，艺术家将这一题材写进小说，搬上银幕，引发人们无限的遐想。然而，作为一门严肃的学科，人工智能在过去的半个多世纪中发展却不算顺利。过去的很多努力还是基于某些预设规则的快速搜索和推理，离真正的智能还有相当的距离，或者说距离创造像人类一样具有抽象学习能力的机器还很遥远。

近年来，深度学习（Deep Learning）直接尝试解决抽象认知的难题，并取得了突破性的进展。深度学习引爆的这场革命，将人工智能带上了一个新的台阶，不仅学术意义巨大，而且实用性很强，工业界也开始了大规模的投入，一大批产品将从中获益。

2006年，机器学习泰斗、多伦多大学计算机系教授Geoffery Hinton在Science发表文章[1]，提出基于深度信念网络（Deep Belief Networks, DBN）可使用非监督的逐层贪心训练算法，为训练深度神经网络带来了希望。

2012年，Hinton又带领学生在目前最大的图像数据库ImageNet上，对分类问题取得了惊人的结果[3]，将Top5错误率由26%大幅降低至15%。

2012年，由人工智能和机器学习顶级学者Andrew Ng和分布式系统顶级专家Jeff Dean领衔的梦幻阵容，开始打造Google Brain项目，用包含16000个CPU核的并行计算平台训练超过10亿个神经元的深度神经网络，在语音识别和图像识别等领域取得了突破性的进展[4]。该系统通过分析YouTube上选取的视频，采用无监督的方式训练深度神经网络，可将图像自动聚类。在系统中输入“cat”后，结果在没有外界干涉的条件下，识别出了猫脸。

2012年，微软首席研究官Rick Rashid在21世纪的计算大会上演示了一套自动同声传译系统[5]，将他的英文演讲实时转换成与他音色相近、字正腔圆的中文演讲。同声传译需要经历语音识别、机器翻译、语音合成三个步骤。该系统一气呵成，流畅的效果赢得了一致认可，深度学习则是这一系统中的关键技术。

2013年，Google收购了一家叫DNN Research的神经网络初创公司，这家公司只有三个人，Geoffrey Hinton和他的两个学生。这次收购并不涉及任何产品和服务，只是希望Hinton可以将深度学习打造为支持Google未来的核心技术。同年，纽约大学教授，深度学习专家Yann LeCun加盟Facebook，出任人工智能实验室主任[6]，负责深度学习的研发工作，利用深度学习探寻用户图片等信息中蕴含的海量信息，希望在未来能给用户提供更智能化的产品使用体验。

2013年，百度成立了百度研究院及下属的深度学习研究所（IDL），将深度学习应用于语音识别和图像识别、检索，以及广告CTR预估（Click-Through-Rate Prediction，pCTR），其中图片检索达到了国际领先水平。2014年又将Andrew Ng招致麾下，Andrew Ng是斯坦福大学人工智能实验室主任，入选过《时代》杂志年度全球最有影响力100人，是16位科技界的代表之一。

如果说Hinton 2006年发表在《Science》杂志上的论文[1]只是在学术界掀起了对深度学习的研究热潮，那么近年来各大巨头公司争相跟进，将顶级人才从学术界争抢到工业界，则标志着深度学习真正进入了实用阶段，将对一系列产品和服务产生深远影响，成为它们背后强大的技术引擎。

目前，深度学习在几个主要领域都获得了突破性的进展：在语音识别领域，深度学习用深层模型替换声学模型中的混合高斯模型（Gaussian Mixture Model, GMM），获得了相对30%左右的错误率降低；在图像识别领域，通过构造深度卷积神经网络（CNN）[3]，将Top5错误率由26%大幅降低至15%，又通过加大加深网络结构，进一步降低到11%；在自然语言处理领域，深度学习基本获得了与其他方法水平相当的结果，但可以免去繁琐的特征提取步骤。可以说到目前为止，深度学习是最接近人类大脑的智能学习方法。

## 深层模型的基本结构

深度学习采用的模型为深层神经网络（Deep Neural Networks，DNN）模型，即包含多个隐藏层（Hidden Layer，也称隐含层）的神经网络（Neural Networks，NN）。深度学习利用模型中的隐藏层，通过特征组合的方式，逐层将原始输入转化为浅层特征，中层特征，高层特征直至最终的任务目标。

深度学习源于人工神经网络的研究，先来回顾一下人工神经网络。一个神经元如下图所示[15]：

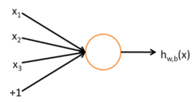


图1 神经元结构

这个神经元接受三个输入x1，x2，x3，神经元输出为

h(W,b) (x)=f(∑(i=1)^3▒〖W\_i x\_i+b〗)，

其中W1, W2, W3和b为神经元的参数，f(z)称为激活函数，一种典型的激活函数为Sigmoid函数，即f(z)=1/(1+e^(-z) )其图像为

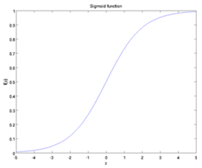


图2 Sigmoid函数图像

神经网络则是多个神经元组成的网络，一个简单的神经网络如下图所示

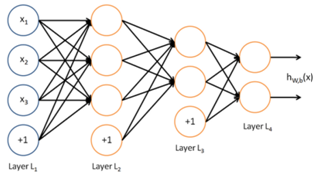


图3 一个简单的神经网络

使用圆圈来表示神经网络的输入，标上“+1”的圆圈称为偏置节点，也就是截距项。神经网络最左边的一层叫做输入层（本例中，有3个输入单元，偏置单元不计）；最右的一层叫做输出层（本例中，输出层有2个节点）；中间的节点叫做隐藏层（本例中，有2个隐藏层，分别包含3个和2个神经元，偏置单元同样不计），因为不能在训练样本集中观测到它们的值。神经元网络中的每一条连线对应一个连接参数，连线个数对应网络的参数个数（本例共有４×３ ＋ ４×２ ＋３×２＝２６个参数）。求解这个的神经网络，需要（x(i)，y(i)）的样本集，其中x(i)是3维向量，y(i)是2维向量。

上图算是一个浅层的神经网络，下图是一个用于语音识别的深层神经网络。具有1个输入层，4个隐藏层和1个输出层，相邻两层的神经元全部连接。

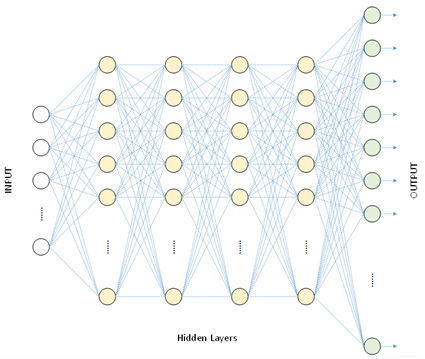


图4 一种典型的深层神经网络模型

## 选择深层模型的原因

为什么要构造包含这么多隐藏层的深层网络结构呢？背后有一些理论依据：

### 天然层次化的特征

对于很多训练任务来说，特征具有天然的层次结构。以语音、图像、文本为例，层次结构大概如下表所示。

表1 几种任务领域的特征层次结构



以图像识别为例，图像的原始输入是像素，相邻像素组成线条，多个线条组成纹理，进一步形成图案，图案构成了物体的局部，直至整个物体的样子。不难发现，可以找到原始输入和浅层特征之间的联系，再通过中层特征，一步一步获得和高层特征的联系。想要从原始输入直接跨越到高层特征，无疑是困难的。

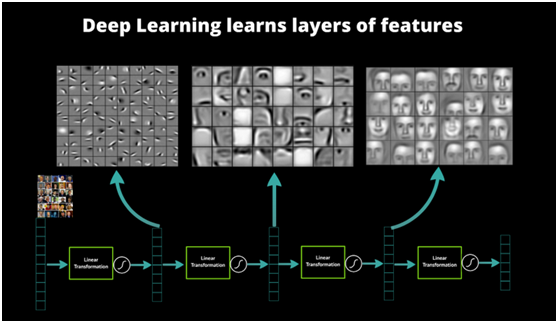


图5 人脸识别系统的多层结构和特征表示 [7]

### 仿生学依据

人工神经网络本身就是对人类神经系统的模拟，这种模拟具有仿生学的依据。1981年，David Hubel 和Torsten Wiesel发现可视皮层是分层的[14]。人类的视觉系统包含了不同的视觉神经元，这些神经元与瞳孔所受的刺激（系统输入）之间存在着某种对应关系（神经元之间的连接参数），即受到某种刺激后（对于给定的输入），某些神经元就会活跃（被激活）。这证实了人类神经系统和大脑的工作其实是不断将低级抽象传导为高级抽象的过程，高层特征是低层特征的组合，越到高层特征就越抽象。

### 特征的层次可表示性

特征的层次可表示性也得到了证实。1995年前后，Bruno Olshausen和David Field[8]收集了很多黑白风景照，从这些照片中找到了400个16×16的基本碎片，然后从照片中再找到其他一些同样大小的碎片，希望将其他碎片表示为这400个基本碎片的线性组合，并使误差尽可能小，使用的碎片尽可能少。表示完成后，再固定其他碎片，选择更合适的基本碎片组合优化近似结果。反复迭代后，得到了可以表示其他碎片的最佳的基本碎片组合。他们发现，这些基本碎片组合都是不同物体不同方向的边缘线。

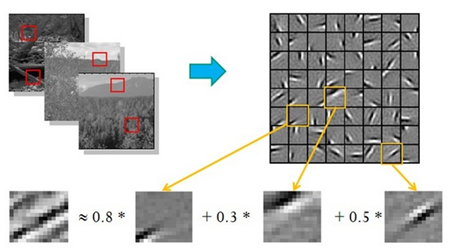


图6 初级图像特征的提取和表示（Sparse Coding）（原图由Andrew Ng提供）

这说明可以通过有效的特征提取，将像素抽象成更高级的特征。类似的结果也适用于语音特征。

## 从浅层模型到深层模型

前文谈到了深层模型的结构和它的优势。事实上，深层模型具有强大的表达能力，并可以像人类一样有效提取高级特征，并不是新的发现。那么为什么深层模型直到最近几年才开始得到广泛的关注和应用呢？还是从传统的机器学习方法和浅层学习谈起。

### 浅层模型及训练方法

反向传播算法（Back Propagation，BP算法）[11]是一种神经网络的梯度计算方法。反向传播算法先定义模型在训练样本上的代价函数，再求代价函数对于每个参数的梯度。反向传播算法巧妙的利用了下层神经元的梯度可由上层神经元的残差导出的规律，求解的过程也正如算法的名字那样，自上而下反向逐层计算，直至获得所有参数的梯度。反向传播算法可以帮助训练基于统计的机器学习模型，从大量的训练样本中挖掘出统计规律，进而可对未标注的数据进行预测。这种基于统计的学习方法比起传统的基于规则的方法具备很多优越性[16]。

上世纪八九十年代，人们提出了一系列机器学习模型，应用最为广泛的包括支持向量机（Support Vector Machine，SVM）[12]和逻辑回归（Logistic Regression，LR）[13]，这两种模型分别可以看作包含1个隐藏层和没有隐藏层的浅层模型。训练时可以利用反向传播算法计算梯度，再用梯度下降方法在参数空间中寻找最优解。浅层模型往往具有凸代价函数，理论分析相对简单，训练方法也容易掌握，取得了很多成功的应用。

### 深层模型的训练难度

浅层模型的局限性在于有限参数和计算单元，对复杂函数的表示能力有限，针对复杂分类问题其泛化能力受到一定的制约。深层模型恰恰可以克服浅层模型的这一弱点，然而应用反向传播和梯度下降来训练深层模型，就面临几个突出的问题[10]：1.梯局部最优。与浅层模型的代价函数不同，深层模型的每个神经元都是非线性变换，代价函数是高度非凸函数，采用梯度下降的方法容易陷入局部最优。

1.梯局部最优。与浅层模型的代价函数不同，深层模型的每个神经元都是非线性变换，代价函数是高度非凸函数，采用梯度下降的方法容易陷入局部最优。

2.梯度弥散。使用反向传播算法传播梯度的时候，随着传播深度的增加，梯度的幅度会急剧减小，会导致浅层神经元的权重更新非常缓慢，不能有效学习。这样一来，深层模型也就变成了前几层相对固定，只能改变最后几层的浅层模型。

3.数据获取。深层模型的表达能力强大，模型的参数也相应增加。对于训练如此多参数的模型，小训练数据集是不能实现的，需要海量的有标记的数据，否则只能导致严重的过拟合（Over fitting）。

### 深层模型的训练方法

尽管挑战很大，Hinton教授并没有放弃努力，他30年来一直从事相关研究，终于有了突破性的进展。2006年，他在《Science》上发表了一篇文章[1]，掀起了深度学习在学术界和工业界的浪潮。这篇文章的两个主要观点是：

1.多隐藏层的人工神经网络具有优异的特征学习能力，学习到的特征对数据有更本质的刻画，从而有利于可视化或分类。

2.深度神经网络在训练上的难度，可以通过“逐层初始化”（Layer-wise Pre-training）来有效克服，文中给出了无监督的逐层初始化方法。

优异的特征刻画能力前文已经提到，不再累述，下面重点解释一下“逐层初始化”的方法。

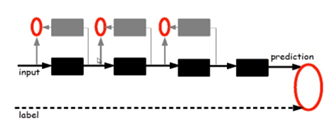


图7 逐层初始化的方法（原图由Marc’Aurelio Ranzato提供）

给定原始输入后，先要训练模型的第一层，即图中左侧的黑色框。黑色框可以看作是一个编码器，将原始输入编码为第一层的初级特征，可以将编码器看作模型的一种“认知”。为了验证这些特征确实是输入的一种抽象表示，且没有丢失太多信息，需要引入一个对应的解码器，即图中左侧的灰色框，可以看作模型的“生成”。为了让认知和生成达成一致，就要求原始输入通过编码再解码，可以大致还原为原始输入。因此将原始输入与其编码再解码之后的误差定义为代价函数，同时训练编码器和解码器。训练收敛后，编码器就是我们要的第一层模型，而解码器则不再需要了。这时我们得到了原始数据的第一层抽象。固定第一层模型，原始输入就映射成第一层抽象，将其当作输入，如法炮制，可以继续训练出第二层模型，再根据前两层模型训练出第三层模型，以此类推，直至训练出最高层模型。

逐层初始化完成后，就可以用有标签的数据，采用反向传播算法对模型进行整体有监督的训练了。这一步可看作对多层模型整体的精细调整。由于深层模型具有很多局部最优解，模型初始化的位置将很大程度上决定最终模型的质量。“逐层初始化”的步骤就是让模型处于一个较为接近全局最优的位置，从而获得更好的效果。

### 浅层模型和深层模型的对比

表2 浅层模型和深层模型的对比



浅层模型有一个重要的特点，需要依靠人工经验来抽取样本的特征，模型的输入是这些已经选取好的特征，模型只用来负责分类和预测。在浅层模型中，最重要的往往不是模型的优劣，而是特征的选取的优劣。因此大多数人力都投入到特征的开发和筛选中来，不但需要对任务问题领域有深刻的理解，还要花费大量时间反复实验摸索，这也限制了浅层模型的效果。

事实上，逐层初始化深层模型也可以看作是特征学习的过程，通过隐藏层对原始输入的一步一步抽象表示，来学习原始输入的数据结构，找到更有用的特征，从而最终提高分类问题的准确性。在得到有效特征之后，模型整体训练也可以水到渠成。

## 深层模型的层次组件

深层模型是包含多个隐藏层的神经网络，每一层的具体结构又是怎样的呢？本节介绍一些常见的深层模型基本层次组件。

### 自编码器（Auto-Encoder）

一种常见的深层模型是由自编码器（Auto-Encoder）构造的[15]。自编码器可以利用一组无标签的训练数据{x(1), x(2), … }（其中x(i)是一个n维向量）进行无监督的模型训练。它采用反向传播算法，让目标值接近输入值。下图是一个自编码器的示例：

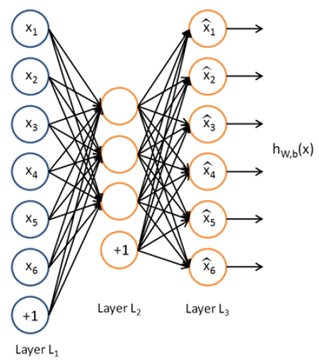


图8 自编码器

自编码器尝试训练一个恒等函数，让输出接近等于输入值，恒等函数看似没有学习的意义，但考虑到隐藏层神经元的数目（本例中为3个）小于输入向量的维数（本例中为6维），事实上隐藏层就变成了输入数据的一种压缩的表示，或说是抽象的简化表示。如果网络的输入是完全随机的，将高维向量压缩成低维向量会难以实现。但训练数据往往隐含着特定的结构，自编码器就会学到这些数据的相关性，从而得到有效的压缩表示。实际训练后，如果代价函数越小，就说明输入和输出越接近，也就说明这个编码器越靠谱。当然，自编码器训练完成后，实际使用时只需要它的前一层，即编码部分，解码部分就没用了。

稀疏自编码器（Sparse Auto-Encoder）是自编码器的一个变体，它在自编码器的基础上加入正则化（Regularity）。正则化是在代价函数中加入抑制项，希望隐藏层节点的平均激活值接近于0，有了正则化的约束，输入数据可以用少数隐藏节点表达。之所以采用稀疏自编码器，是因为稀疏的表达往往比稠密的表达更有效，人脑神经系统也是稀疏连接，每个神经元只与少数神经元连接。

降噪自编码器是另一种自编码器的变体。通过在训练数据中加入噪声，可训练出对输入信号更加鲁棒的表达，从而提升模型的泛化能力，可以更好地应对实际预测时夹杂在数据中的噪声。

得到自编码器后，我们还想进一步了解自编码器到底学到了什么。例如，在10×10的图像上训练一个稀疏自编码器，然后对于每个隐藏神经元，找到什么样的图像可以让隐藏神经元获得最大程度的激励，即这个隐藏神经元学习到了什么样的特征。将100个隐藏神经元的特征都找出来，得到了如下100幅图像：

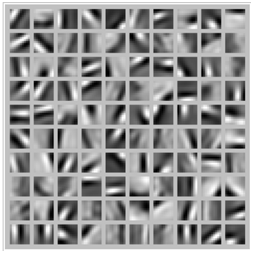


图9 自编码器的隐藏神经元 [15]

可以看出，这100幅图像具备了从不同方向检测物体边缘的能力。显然，这样的能力对后续的图像识别很有帮助。

### 受限玻尔兹曼机（Restricted Boltzmann Machine，RBM）

受限玻尔兹曼机（Restricted Boltzmann Machine， RBM）是一个二部图，一层是输入层（v），另一层是隐藏层（h），假设所有节点都是随机二值变量节点，只能取值0或1，同时假设全概率分布p(v, h)满足Boltzmann分布。

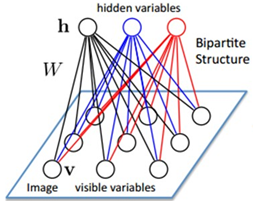


图10受限玻尔兹曼机（RBM）

由于同层节点之间没有连接，因此已知输入层的情况下，隐藏层的各节点是条件独立的；反之，已知隐藏层的情况下，输入层各节点也是条件独立的。同时，可以根据Boltzmann分布，当输入v时通过p(h|v)生成隐藏层，得到隐藏层之后再通过p(v|h)生成输入层。相信很多读者已经猜到了，可以按照训练其他网络类似的思路，通过调整参数，希望通过输入v生成的h，再生成的v’与v尽可能接近，则说明隐藏层h是输入层v的另外一种表示。这样就可以作为深层模型的基本层次组件了。全部用RBM形成的深层模型为深度玻尔兹曼机（Deep Boltzmann Machine，DBM）。如果将靠近输入层的部分替换为贝叶斯信念网络，即有向图模型，而在远离输入层的部分仍然使用RBM，则称为深度信念网络（Deep Belief Networks，DBN）。

### 卷积神经网络（Convolutional Neural Networks, CNN）

以上介绍的编码器都是全连通网络，可以完成10×10的图像识别，如手写体数字识别问题。然而对于更大的图像，如100×100的图像，如果要学习100个特征，则需要1,000,000个参数，计算时间会大大增加。解决这种尺寸图像识别的有效方法是利用图像的局部性，构造一个部分联通的网络。一种最常见的网络是卷积神经网络（Convolutional Neural Networks，CNN）[19][20]，它利用图像固有的特性，即图像局部的统计特性与其他局部是一样的。因此从某个局部学习来的特征同样适用于另外的局部，对于这个图像上的所有位置，都能使用同样的特征。

具体地说，假设有一幅100×100的图像，要从中学习一个10×10的局部图像特征的神经元，如果采用全连接的方式，100×100维的输入到这个神经元需要有10000个连接权重参数。而采用卷积核的方式，只有10×10=100个参数权重，卷积核可以看作一个10×10的小窗口，在图像上上下左右移动，走遍图像中每个10×10的位置（共有91×91个位置）。每移动到一个位置，则将该位置的输入与卷积核对应位置的参数相乘再累加，得到一个输出值（输出值是91×91的图像）。卷积核的特点是连接数虽然很多，有91×91×10×10个连接，但是参数只有10×10=100个，参数数目大大减小，训练也变得容易了，并且不容易产生过拟合。当然，一个神经元只能提取一个特征，要提取多个特征就要多个卷积核。

下图揭示了对一幅8×8维图像使用卷积方法提取特征的示意过程。其中使用了3×3的卷积核，走遍图像中每个3×3的位置后，最终得到6×6维的输出图像：

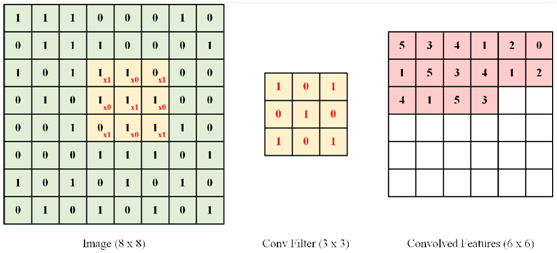


图11 8×8图像的卷积过程示意

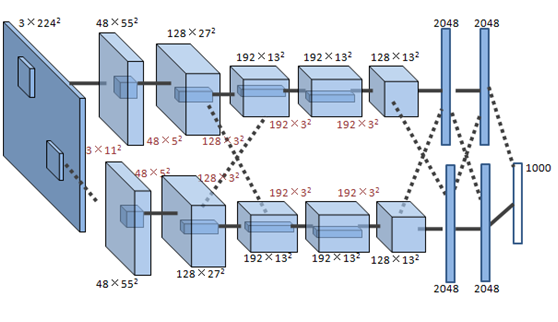


图12 用户图像分类的卷积神经网络

如图所示是Hinton的研究小组在ImageNet竞赛中使用的卷积神经网络[3]，共有5个卷积层，每层分别有96，256，384，384和256个卷积核，每层卷积核的大小分别为11×11，5×5，3×3，3×3和3×3。网络的最后两层是全连接层。

## 深度学习的训练加速

深层模型训练需要各种技巧，例如网络结构的选取，神经元个数的设定，权重参数的初始化，学习率的调整，Mini-batch的控制等等。即便对这些技巧十分精通，实践中也要多次训练，反复摸索尝试。此外，深层模型参数多，计算量大，训练数据的规模也更大，需要消耗很多计算资源。如果可以让训练加速，就可以在同样的时间内多尝试几个新主意，多调试几组参数，工作效率会明显提升，对于大规模的训练数据和模型来说，更可以将难以完成的任务变成可能。这一节就谈谈深层模型的训练加速方法。

### GPU加速

矢量化编程是提高算法速度的一种有效方法。为了提升特定数值运算操作（如矩阵相乘、矩阵相加、矩阵-向量乘法等）的速度，数值计算和并行计算的研究人员已经努力了几十年。矢量化编程强调单一指令并行操作多条相似数据，形成单指令流多数据流（SIMD）的编程泛型。深层模型的算法，如BP，Auto-Encoder，CNN等，都可以写成矢量化的形式。然而，在单个CPU上执行时，矢量运算会被展开成循环的形式，本质上还是串行执行。

GPU（Graphic Process Units，图形处理器）的众核体系结构包含几千个流处理器，可将矢量运算并行化执行，大幅缩短计算时间。随着NVIDIA、AMD等公司不断推进其GPU的大规模并行架构支持，面向通用计算的GPU（General-Purposed GPU, GPGPU）已成为加速可并行应用程序的重要手段。得益于GPU众核（many-core）体系结构，程序在GPU系统上的运行速度相较于单核CPU往往提升几十倍乃至上千倍。目前GPU已经发展到了较为成熟的阶段，受益最大的是科学计算领域，典型的成功案例包括多体问题（N-Body Problem）、蛋白质分子建模、医学成像分析、金融计算、密码计算等。

利用GPU来训练深度神经网络，可以充分发挥其数以千计计算核心的高效并行计算能力，在使用海量训练数据的场景下，所耗费的时间大幅缩短，占用的服务器也更少。如果对针对适当的深度神经网络进行合理优化，一块GPU卡可相当于数十甚至上百台CPU服务器的计算能力，因此GPU已经成为业界在深度学习模型训练方面的首选解决方案。

### 数据并行

数据并行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。

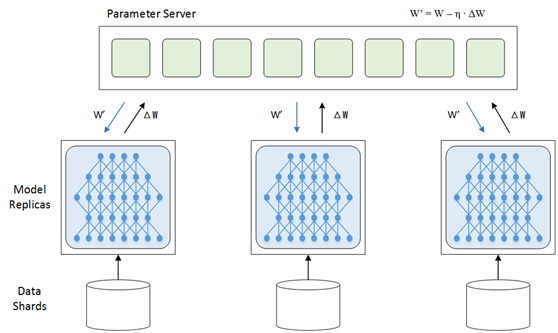


图13 数据并行的基本架构 [17]

要完成数据并行需要做参数交换，通常由一个参数服务器（Parameter Server）来帮助完成。在训练的过程中，多个训练过程相互独立，训练的结果，即模型的变化量ΔW需要汇报给参数服务器，由参数服务器负责更新为最新的模型W’ = W – η ∙ ΔW，然后再将最新的模型W’分发给训练程序，以便从新的起点开始训练。

数据并行有同步模式和异步模式之分。同步模式中，所有训练程序同时训练一个批次的训练数据，完成后经过同步，再同时交换参数。参数交换完成后所有的训练程序就有了共同的新模型作为起点，再训练下一个批次。而异步模式中，训练程序完成一个批次的训练数据，立即和参数服务器交换参数，不考虑其他训练程序的状态。异步模式中一个训练程序的最新结果不会立刻体现在其他训练程序中，直到他们进行下次参数交换。

参数服务器只是一个逻辑上的概念，不一定部署为独立的一台服务器。有时候它会附属在某一个训练程序上，有时也会将参数服务器按照模型划分为不同的分片，分别部署。

### 模型并行

模型并行将模型拆分成几个分片，由几个训练单元分别持有，共同协作完成训练。当一个神经元的输入来自另一个训练单元上的神经元的输出时，产生通信开销。

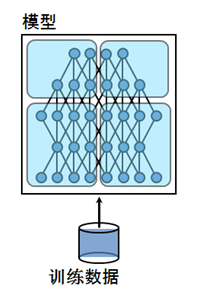


图14 模型并行的基本架构 [17]

多数情况下，模型并行带来的通信开销和同步消耗超过数据并行，因此加速比也不及数据并行。但对于单机内存无法容纳的大模型来说，模型并行是一个很好的选择。令人遗憾的是，数据并行和模型并行都不能无限扩展。数据并行的训练程序太多时，不得不减小学习率，以保证训练过程的平稳；模型并行的分片太多时，神经元输出值的交换量会急剧增加，效率大幅下降。因此，同时进行模型并行和数据并行也是一种常见的方案。如下图所示，4个GPU分为两组，GPU0，1为一组模型并行，GPU2，3为另一组，每组模型并行在计算过程中交换输出值和残差。两组GPU之间形成数据并行，Mini-batch结束后交换模型权重，考虑到模型的蓝色部分由GPU0和GPU2持有，而黄色部分由GPU1和GPU3持有，因此只有同色的GPU之间需要交换权重。

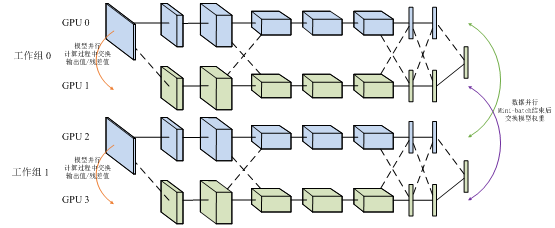


图15 4GPU卡的数据并行和模型并行混合架构

### 计算集群

搭建CPU集群用于深度神经网络模型训练也是业界常用的解决方案，其优势在于利用大规模分布式计算集群的强大计算能力，利用模型可分布式存储、参数可异步通信的特点，达到快速训练深层模型的目的。

CPU集群方案的基本架构包含用于执行训练任务的Worker、用于分布式存储分发模型的参数服务器（Parameter Server）和用于协调整体任务的主控程序（Master）。CPU集群方案适合训练GPU内存难以容纳的大模型，以及稀疏连接神经网络。Andrew Ng和Jeff Dean在Google用1000台CPU服务器，完成了模型并行和Downpour SGD数据并行的深度神经网络训练[17]。

结合GPU计算和集群计算技术，构建GPU集群正在成为加速大规模深度神经网络训练的有效解决方案。GPU集群搭建在CPU-GPU系统之上，采用万兆网卡或Infiniband等更加快速的网络通信设施，以及树形拓扑等逻辑网络拓扑结构。在发挥出单节点较高计算能力的基础上，再充分挖掘集群中多台服务器的协同计算能力，进一步加速大规模训练任务。

## 深度学习的软件工具及平台

目前，在深度学习系统实现方面，已有诸多较为成熟的软件工具和平台。

### 开源软件

在开源社区，主要有以下较为成熟的软件工具：

Kaldi是一个基于C++和CUDA的语音识别工具集[22][23]，提供给语音识别的研究人员使用。Kaldi中既实现了用单个GPU加速的深度神经网络SGD训练，也实现了CPU多线程加速的深度神经网络SGD训练。

Cuda-convnet基于C++/CUDA编写，采用反向传播算法的深度卷积神经网络实现[24][25]。2012年cuda-convnet发布，可支持单个GPU上的训练，基于其训练的深度卷积神经网络模型在ImageNet LSVRC-2012对图像按1000个类目分类，取得Top 5分类15%错误率的结果[3]；2014年发布的版本可以支持多GPU上的数据并行和模型并行训练[26]。

Caffe提供了在CPU以及GPU上的快速卷积神经网络实现，同时提供训练算法，使用NVIDIA K40或Titan GPU可以1天完成多于40,000,000张图片的训练[27][28]。

Theano提供了在深度学习数学计算方面的Python库，它整合了NumPy矩阵计算库，可以运行在GPU上，并提供良好的算法上的扩展性[29][30]。

OverFeat是由纽约大学CILVR实验室开发的基于卷积神经网络系统，主要应用场景为图像识别和图像特征提取[31]。

Torch7是一个为机器学习算法提供广泛支持的科学计算框架，其中的神经网络工具包（Package）实现了均方标准差代价函数、非线性激活函数和梯度下降训练神经网络的算法等基础模块，可以方便地配置出目标多层神经网络开展训练实验[32]。

### 工业界平台

在工业界，Google、Facebook、百度、腾讯等公司都实现了自己的软件框架：

Google的DistBelief系统是CPU集群实现的数据并行和模型并行框架，集群内使用上万CPU core来训练多达10亿参数的深度神经网络模型。DistBelief应用的主要算法有Downpour SGD和L-BFGS，支持的目标应用有语音识别和2.1万类目的图像分类[17]。

Google的COTS HPC系统是GPU实现的数据并行和模型并行框架，GPU服务器间使用了Infiniband连接，并由MPI控制通信。COTS可以用3台GPU服务器在数天内完成对10亿参数的深度神经网络训练[33]。

Facebook实现了多GPU训练深度卷积神经网络的并行框架，结合数据并行和模型并行的方式来训练CNN模型，使用4张NVIDIA Titan GPU可在数天内训练ImageNet的1000分类网络[34]。

百度搭建了Paddle（Parallel Asynchonous Distributed Deep Learning）多机GPU训练平台[35]。将数据分布到不同机器，通过Parameter Server协调各机器训练。Paddle支持数据并行和模型并行。

腾讯深度学习平台（Mariana）是为加速深度学习模型训练而开发的并行化平台，包括深度神经网络的多GPU数据并行框架，深度卷积神经网络的多GPU模型并行和数据并行框架，以及深度神经网络的CPU集群框架。Mariana基于特定应用的训练场景，设计定制化的并行化训练平台，支持了语音识别、图像识别，并积极探索在广告推荐中的应用[36]。

## 总结

近年来人工智能领域掀起了深度学习的浪潮，从学术界到工业界都热情高涨。深度学习尝试解决人工智能中抽象认知的难题，从理论分析和应用方面都获得了很大的成功。可以说深度学习是目前最接近人脑的智能学习方法。

深度学习可通过学习一种深层非线性网络结构，实现复杂函数逼近，并展现了强大的学习数据集本质和高度抽象化特征的能力。逐层初始化等训练方法显著提升了深层模型的可学习型。与传统的浅层模型相比，深层模型经过了若干层非线性变换，带给模型强大的表达能力，从而有条件为更复杂的任务建模。与人工特征工程相比，自动学习特征，更能挖掘出数据中丰富的内在信息，并具备更强的可扩展性。深度学习顺应了大数据的趋势，有了充足的训练样本，复杂的深层模型可以充分发挥其潜力，挖掘出海量数据中蕴含的丰富信息。强有力的基础设施和定制化的并行计算框架，让以往不可想象的训练任务加速完成，为深度学习走向实用奠定了坚实的基础。已有Kaldi，Cuda-convnet，Caffe等多个针对不同深度模型的开源实现，Google、Facebook、百度、腾讯等公司也实现了各自的并行化框架。

深度学习引爆的这场革命，将人工智能带上了一个新的台阶，不仅学术意义巨大，而且实用性很强，深度学习将成为一大批产品和服务背后强大的技术引擎。

## 参考文献

[1] Geoffery E. Hinton, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006 Jul 28;313(5786):504-7.

[2] Geoffrey E. Hinton, Simon Osindero, Yee-Whye Teh, A fast learning algorithm for deep belief nets Neural Compute, 18(7), 1527-54 (2006)

[3] ImageNet Classification with Deep Convolutional Neural Networks, Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, NIPS 2012.

[4] Q.V. Le, M.A. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, A.Y. Ng. Building high-level features using large scale unsupervised learning. ICML, 2012.

[5] Rick Rashid, Speech Recognition Breakthrough for the Spoken, Translated Wordhttp://www.youtube.com/watch?v=Nu-nlQqFCKg

[6] NYU “Deep Learning” Professor LeCun Will Lead Facebook’s New Artificial Intelligence Lab. <http://techcrunch.com/2013/12/09/facebook-artificial-intelligence-lab-lecun/>

[7] A Primer on Deep Learning  
<http://www.datarobot.com/blog/a-primer-on-deep-learning/>

[8] Bruno A. Olshausen & David J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature. Vol 381. 13 June, 1996<http://www.cs.ubc.ca/~little/cpsc425/olshausen_field_nature_1996.pdf>

[9] Andrew Ng. Machine Learning and AI via Brain simulations,  
<https://forum.stanford.edu/events/2011slides/plenary/2011plenaryNg.pdf>

[10] Deep Networks Overview<http://ufldl.stanford.edu/wiki/index.php/Deep_Networks:_Overview>

[11] Back propagation algorithm<http://ufldl.stanford.edu/wiki/index.php/Backpropagation_Algorithm>

[12] Support Vector Machine <http://en.wikipedia.org/wiki/Support_vector_machine>

[13] Logistic Regression <http://en.wikipedia.org/wiki/Logistic_regression>

[14] The Nobel Prize in Physiology or Medicine 1981.http://www.nobelprize.org/nobel\_prizes/medicine/laureates/1981/

[15] Stanford deep learning tutorial  
<http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial>

[16] 余凯，深度学习-机器学习的新浪潮，Technical News程序天下事<http://blog.csdn.net/datoubo/article/details/8577366>

[17] Dean, J., Corrado, G.S., Monga, R., et al, Ng, A. Y. Large Scale Distributed Deep Networks. In Proceedings of the Neural Information Processing Systems (NIPS’12) (Lake Tahoe, Nevada, United States, December 3–6, 2012). Curran Associates, Inc, 57 Morehouse Lane, Red Hook, NY, 2013, 1223-1232.

[18] Geoffrey Hinton：UCLTutorial on: Deep Belief Nets

[19] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press, 1995

[20] Introduction to Convolutional neural network<http://en.wikipedia.org/wiki/Convolutional_neural_network>

[21] Krizhevsky, Alex. “ImageNet Classification with Deep Convolutional Neural Networks”. Retrieved 17 November 2013.

[22] Kaldi project <http://kaldi.sourceforge.net/>

[23] Povey, D., Ghoshal, A. Boulianne, G., et al, Vesely, K. Kaldi. The Kaldi Speech Recognition Toolkit. in Proceedings of IEEE 2011 Workshop on Automatic Speech Recognition and Understanding(ASRU 2011) (Hilton Waikoloa Village, Big Island, Hawaii, US, December 11-15, 2011). IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

[24] cuda-convent <https://code.google.com/p/cuda-convnet/>

[25] Krizhevsky, A., Sutskever, I., and Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Neural Information Processing Systems (NIPS’12) (Lake Tahoe, Nevada, United States, December 3–6, 2012). Curran Associates, Inc, 57 Morehouse Lane, Red Hook, NY, 2013, 1097-1106.

[26] Krizhevsky, A. Parallelizing Convolutional Neural Networks. in tutorial of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014). (Columbus, Ohio, USA, June 23-28, 2014). 2014.

[27] caffe <http://caffe.berkeleyvision.org/>

[28] Jia, Y. Q. Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding. http://caffe.berkeleyvision.org (2013).

[29] Theano <https://github.com/Theano/Theano>

[30] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio. Theano: A CPU and GPU Math Expression Compiler. Proceedings of the Python for Scientific Computing Conference (SciPy) 2010. June 30 - July 3, Austin, TX.

[31] Overfeat <http://cilvr.nyu.edu/doku.php?id=code:start>

[32] Torch7 [http://torch.ch](http://torch.ch/)

[33] Coates, A., Huval, B., Wang, T., Wu, D. J., Ng, A. Y. Deep learning with COTS HPC systems. In Proceedings of the 30th International Conference on Machine Learning (ICML’13) (Atlanta, Georgia, USA, June 16–21, 2013). JMLR: W&CP volume 28(3), 2013, 1337-1345.

[34] Yadan, O., Adams, K., Taigman, Y., Ranzato, M. A. Multi-GPU Training of ConvNets. arXiv:1312.5853v4 [cs.LG] (February 2014)

[35] Kaiyu, Large-scale Deep Learning at Baidu, ACM International Conference on Information and Knowledge Management (CIKM 2013)

[36] aaronzou, Mariana深度学习在腾讯的平台化和应用实践

[37] “Convolutional Neural Networks (LeNet) - DeepLearning 0.1 documentation”. DeepLearning 0.1. LISA Lab. Retrieved 31 August 2013.

[38] Bengio, Learning Deep Architectures for AI，<http://www.iro.umontreal.ca/~bengioy/papers/ftml_book.pdf>；

[39] Deep Learning <http://deeplearning.net/>

[40] Deep Learning <http://www.cs.nyu.edu/~yann/research/deep/>

[41] Introduction to Deep Learning. <http://en.wikipedia.org/wiki/Deep_learning>

[42] Google的猫脸识别:人工智能的新突破<http://www.36kr.com/p/122132.html>

[43] Andrew Ng’s talk video: <http://techtalks.tv/talks/machine-learning-and-ai-via-brain-simulations/57862/>

[44] Invited talk “A Tutorial on Deep Learning” by Dr. Kai Yu  
<http://vipl.ict.ac.cn/News/academic-report-tutorial-deep-learning-dr-kai-yu>