

生产实习报告

**专业班级：**计算机科学与技术1506班

**学 号：** U201514652

**姓 名：** 陈雨浩

**指导教师：** 肖 亮

**实习单位：** 华中科技大学计算机学院

**实习时间：** 2018.07 - 2018.08

**计算机科学与技术学院**

**目 录**

[1 实习概要 2](#_Toc522971788)

[1.1 实习目的与要求 2](#_Toc522971789)

[1.2 实习内容 3](#_Toc522971790)

[1.2.1 调查和记录 3](#_Toc522971791)

[1.2.2 生产实践 4](#_Toc522971792)

[1.3 实习单位与实习岗位简介 4](#_Toc522971793)

[1.3.1 实习单位 4](#_Toc522971794)

[1.3.2 实习指导老师 5](#_Toc522971795)

[1.3.3 实习岗位 6](#_Toc522971796)

[2 实习报告 7](#_Toc522971797)

[2.1 实习项目概述 7](#_Toc522971798)

[2.1.1 项目背景 7](#_Toc522971799)

[2.1.2 深度学习简介 7](#_Toc522971800)

[2.1.3 CUDA简介 9](#_Toc522971801)

[2.1.4 项目概述 10](#_Toc522971802)

[2.2 Caffe模型解析 10](#_Toc522971803)

[2.2.1 Blob的存储与交换 10](#_Toc522971804)

[2.2.2 Layer的计算与连接 12](#_Toc522971805)

[2.2.3 Net的定义与操作 13](#_Toc522971806)

[2.2.4 模型格式 14](#_Toc522971807)

[2.3 前向传播与反向传播 15](#_Toc522971808)

[2.3.1 前向传播 15](#_Toc522971809)

[2.3.2 反向传播 16](#_Toc522971810)

[2.3.3 Caffe中前传和反传的实现 16](#_Toc522971811)

[2.4 损失 17](#_Toc522971812)

[2.5 Solver 18](#_Toc522971813)

[2.5.1 Solver简介 18](#_Toc522971814)

[2.5.2 Solver方法 18](#_Toc522971815)

[2.6 Layer Cataloge 21](#_Toc522971816)

[2.6.1 视觉层Vision Layers 21](#_Toc522971817)

[2.6.2 损失层Loss Layers 25](#_Toc522971818)

[2.6.3 激活层Activation Layers 26](#_Toc522971819)

[2.6.4 数据层Data Layers 28](#_Toc522971820)

[2.6.5 普通层Common Layers 29](#_Toc522971821)

[3 实习工作总结 31](#_Toc522971822)

[4 实习问题解答 32](#_Toc522971823)

# 1 实习概要

## 1.1 实习目的与要求

生产实习是高等工科院校教学计划中实践教学的重要环节之一，是理论与 实际相结合的有效方式。

通过生产实习环节，学生可以利用所学的理论知识去分析实习中所看到的实 际生产技术，从而将科学的理论知识加以验证、深化、巩固和充实。生产实习还 能培养学生进行调查、研究、分析和解决实际问题的能力，拓展学生的知识面， 既体会到学习书本知识的必要性，又提高解决实际工程问题的能力。此外，生产 实习也是学生接触社会、了解社会的重要途径，为后续专业课程的学习、课程设计和毕业设计打下坚实的基础。

通过生产实习，还可以拓宽学生的知识面，增强感性认识，把所学知识条理化、系统化，学习一些从书本上学不到的专业知识，并获得本专业国内、外科技发展现状的最新信息，激发学生向实践学习和探索的积极性，为今后的学习和将从事的技术工作打下坚实的基础，是培养学生工程技术素养、实现理论联系实际的重要措施。通过生产实习，学生把所学的理论知识与生产实际相结合，能够重点培养学生分析问题和解决问题的能力。

在生产实习中，可根据学生特点和实习单位的具体实习要求确定实习内容。

生产实习的总体要求如下：

[1] 了解和学习生产单位的管理和经营方式。

[2] 重点学习和掌握产品开发和研制过程、使用方法、采用的技术和实际操 作能力。

[3] 学习工人、技术人员和干部的认真工作和敬业精神。

[4] 在计算机系统生产单位或计算机研究单位实习，要求学生在实习中对计 算机整体各部分组成、结构、功能、测试、维修等方面有较完整的了解，对某些方面有深入的了解。

[5] 在计算机应用部门实习，要求学生在生产实习中了解应用计算机进行管 理或控制的全过程，并参与软件或硬件应用系统的开发。

[6] 了解企事业、工厂、研究所或公司对计算机专业人才的要求；了解在这些部门工作的管理人员、工程技术人员和普通工人所在工作岗位的职责和工作程序。

[7] 切实认真记好生产实习日记、写好实习报告并参加生产实习考核。

[8] 工程知识：能够将数学、自然科学、工程基础和专业知识用于解决复杂工程问题。

[9] 问题分析：能够应用数学、自然科学和工程科学的基本原理，识别、表达、并通过文献研究分析复杂工程问题，以获得有效结论。

[10] 设计/开发解决方案：能够设计针对复杂工程问题的解决方案，设计满足特定需求的系统、单元（部件）或工艺流程，并能够在设计环节中体现创新意识，考虑社会、健康、安全、法律、文化以及环境等因素。

[11] 研究：能够基于科学原理并采用科学方法对复杂工程问题进行研究，包括设计实验、分析与解释数据、并通过信息综合得到合理有效的结论。

[12] 终身学习：具有自主学习和终身学习的意识，有不断学习和适应发展的能力。

## 1.2 实习内容

生产实习内容包括两个方面：

①调查和记录

②生产实践。

### 1.2.1 调查和记录

1、调查和记载实习单位的管理方式和经营情况：

* 单位的组织结构
* 技术部门结构及人员配备情况
* 生产的产品或销售规模、产值、产量及新产品的研制情况
* 实习单位的各种规章制度，包括规章制度名称和重要规定

2、记录产品研制、开发的方式及过程：

* 计算机硬件、软件的组成、结构和性能、测试及维修等情况，用学过的表示方法加以描述、绘制和记载
* 记载产品生产和装配的工序、工艺及各个环节上的重要处理技术
* 产品开发或研制的技术特点及技术性能的先进性

3、收集整理产品开发的技术资料：

* 硬部件的线路图及技术参数
* 各种软件设计文档
* 产品性能指标及参数
* 产品使用说明书

以上内容应严格按照生产实习单位的工作要求有选择地记载，具体内容可包括各种资料的类型、格式、图样及全部或部分内容，注意生产实习单位保密工作要求，所记载内容均写在“生产实习日记”中。

### 1.2.2 生产实践

每个学生应根据所在实习单位的具体条件完成下列一至二项工作并做详细记载。

1.了解计算机、计算机网络和计算机新技术的实际应用情况；

2.剖析实习单位引进或研制的最新软件环境，软件工具及支撑软件的结构和采用的技术；

3.参加新产品某一部分的开发试制工作，解决 l～2 个实际技术问题；

4.掌握软件开发工具及多媒体软件的应用；

5.参与系统总体调试工作；

6.独立承担模块的编程和调试。

## 1.3 实习单位与实习岗位简介

### 1.3.1 实习单位

华中科技大学“服务计算技术与系统教育部重点实验室”暨“集群与网格计算湖北省重点实验室”依托于计算机系统结构国家重点学科和计算机软件与理论湖北省重点学科，建有湖北省大数据技术与系统工程实验室，拥有自由开放的学术氛围和国际前沿的研究方向。目前主要的研究领域包括：系统软件与体系结构、云计算与移动计算、网络空间安全、大数据等。

实验室具备实力雄厚的师资力量、充满活力的科研梯队以及良好的硬件设施环境。现有教授11人，副教授17人，具有博士学位者29人，其中长江学者特聘教授1人，973计划项目首席科学家2人次、国家杰出青年基金获得者1人、“新世纪百千万人才工程”国家级入选1人、湖北省高端人才引领培养计划入选1人、全国百优1人、教育部“长江学者”青年项目获得者1人、中组部“青年拔尖人才支持计划” 入选2人、国家自然科学基金优秀青年基金获得者3人、“新世纪优秀人才支持计划”入选3人、“楚天学子”入选2人、湖北省杰出青年基金获得者3人、湖北省十佳师德标兵1人、武汉市“青年科技晨光计划”入选4人、微软亚洲研究院“铸星计划”3人。目前在读全日制博士、硕士研究生200余人。实验室承建了中国教育科研网格ChinaGrid主结点、中国国家网格CNGrid（武汉）结点、985科技创新平台；拥有3000平方米实验基地，主要实验设备资产总值达9千多万元。实验室发展与建设将直接为高层次人才的培养提供良好的基础设施与外部条件。

实验室承担了70余项重大科研项目，包括国家973项目、教育部重大专项、国家科技支撑计划项目、国家科技重大专项、国家杰出青年基金项目、国家自然科学基金重大/重点项目、国家863重大/重点项目、国家发改委CNGI项目、科技部国际合作项目等。实验室现为科技部重点领域创新团队、教育部“长江学者和创新团队发展计划”创新团队牵头单位、湖北省自然科学基金创新团队。

实验室出版专著和教材18余本，发表国内外学术期刊论文及国际学术会议论文近800篇，获得国际发明专利6项、获得国家发明专利201项，在审国家发明专利127项，获得国家软件著作版权162项；获得国家科技进步二等奖2项、国家技术发明二等奖1项、国家自然科学四等奖1项、省部级科技进步/技术发明一等奖4项、二等奖1项、湖北省教学成果一等奖1项。

实验室坚持开放与联合，与美国、德国、澳大利亚、日本、英国、法国、加拿大、匈牙利、挪威等国家和香港、台湾地区的大学，以及Intel、惠普、微软、IBM、AMD、法国电信、曙光、UT斯达康、华为、中国移动、腾讯、天涯在线、阿里巴巴、海尔、西门子等国内外知名IT企业保持着密切合作，先后建立了华中科技大学-惠普高性能计算联合实验室、Intel“安腾”应用研发中心、UT斯达康IPTV联合实验室、Intel“多核技术”实验室、华中科技大学-天涯在线联合实验室、阿里巴巴联合实验室、华中科技大学—腾讯联合实验室。此外，实验室成功举办了近30余次具有重要影响的国际/国内学术会议。

实验室本着“明德厚学、求是创新”的华中大精神，坚持“育才明德、开放竞争”的原则，不断开拓进取，致力成为国内一流、国际知名的研究开发基地和人才培养基地。

### 1.3.2 实习指导老师

蒋文斌导师，博士、副教授。主要从事深度学习、并行分布式计算、媒体计算等领域的研究。主持了包括支撑计划、973课题子项、自科基金在内的多个国家项目。已在PPoPP、FGCS、NOSSDAV、TCSVT等重要学术会议和期刊上发表学术论文近50篇。获得发明专利12项。获得湖北省教学成果一等奖一项(2013)。先后在日本会津大学(2005)、美国加州大学洛杉矶分校(2014)、新加坡国立大学(2017)访学。担任了多个国际学术会议的程序委员会、宣传主席等职务。现为IEEE Computer Society会员、ACM会员、CCF普适计算专委会委员。

### 1.3.3 实习岗位

在蒋文斌老师的指导安排下，我被分配到其研究生张杨松和刘湃团队内，主要的工作是掌握基本的深度学习知识，看懂Caffe项目源码，有余力可尝试调试优化。

# 2 实习报告

## 2.1 实习项目概述

### 2.1.1 项目背景

Caffe（Convolutional Architecture for Fast Feature Embedding），是一种常用的深度学习框架，主要应用在视频、图像处理方面的应用上，具有表达力强、速度快和模块化的特点，由伯克利视觉学习中心（BVLC）和社区贡献者开发。

Caffe采用了纯粹的C++/CUDA架构，支持命令行、Python和MATLAB接口。Caffe的清晰性表现在其网络结构与参数都独立于代码，用户只需以遵循一定简单格式的普通文本就可以定义自己的神经网络，并根据自己的需要进行调整。Caffe的高效性则体现在其对CUDA的支持，GPU 运算能极大地提高图像处理的速度，同时Caffe提供了在CPU模式和GPU模式之间的无缝切换。

Caffe的特点：

1、上手快：模型和优化以纯文本的模式定义，而不是以代码模式。Caffe给出了模型的定义、最优化设置以及预训练的权重，方便立即上手。

2、速度快：能够运行最棒的模型与海量的数据。Caffe与cuDNN结合使用，测试AlexNet模型，在K40上处理每张图片只需要1.17ms。

3、模块化：具有灵活性和扩展性，方便扩展到新的任务和设置上。可以使用Caffe提供的各层类型来定义自己的模型。

4、开放性：公开的代码和参考模型用于再现。

5、社区好：可以通过开源社区和Github参与讨论和开发。

### 2.1.2 深度学习简介

深度学习的概念源于人工神经网络的研究。含多隐层的多层感知机就是一种深度学习结构。深度学习通过组合低层特征形成更加抽象的高层表示属性类别或特征，以发现数据的分布式特征表示。

深度学习的概念由Hinton等人于2006年提出。基于深度置信网络(DBN)提出非监督贪心逐层训练算法，为解决深层结构相关的优化难题带来希望，随后提出多层自动编码器深层结构。此外Lecun等人提出的卷积神经网络是第一个真正多层结构学习算法，它利用空间相对关系减少参数数目以提高训练性能。

深度学习是机器学习中一种基于对数据进行表征学习的方法。观测值（例如一幅图像）可以使用多种方式来表示，如每个像素强度值的向量，或者更抽象地表示成一系列边、特定形状的区域等。而使用某些特定的表示方法更容易从实例中学习任务（例如人脸识别或面部表情识别）。深度学习的好处是用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。

深度学习是机器学习研究中的一个新的领域，其动机在于建立、模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释数据，例如图像、声音和文本。

同机器学习方法一样，深度机器学习方法也有监督学习与无监督学习之分。不同的学习框架下建立的学习模型很是不同。例如卷积神经网络（Convolutional neural networks，简称CNNs）就是一种深度的监督学习下的机器学习模型，而深度置信网（Deep Belief Nets，简称DBNs）就是一种无监督学习下的机器学习模型。

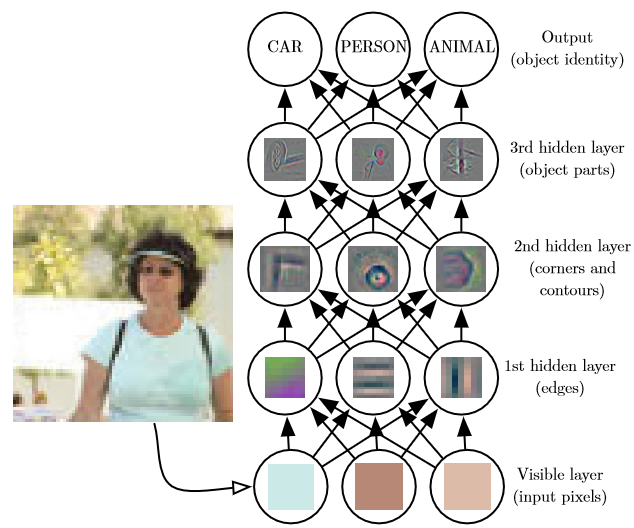


图2-1 深度学习模型的示意图

计算机难以理解原始感观输入数据的含义，如表示为像素值集合的图像。将一组像素映射到对象标识的函数非常复杂。如果直接处理，学习或评估此映射似乎是不可能的。深度学习将所需的复杂映射分解为一系列嵌套的简单映射（每个由模型的不同层描述）来解决这一难题。如图2-1所示，输入展示在可见层（visible layer），这样命名的原因是因为它包含我们能观察到的变量。然后是一系列从图像中提取越来越多抽象特征的隐藏层（hidden layer）。因为它们的值不在数据中给出，所以将这些层称为“隐藏”；模型必须确定哪些概念有利于解释观察数据中的关系。这里的图像是每个隐藏单元表示的特征的可视化。给定像素，第一层可以轻易地通过比较相邻像素的亮度来识别边缘。有了第一隐藏层描述的边缘，第二隐藏层可以容易地搜索可识别为角和扩展轮廓的边集合。给定第二隐藏层中关于角和轮廓的图像描述，第三隐藏层可以找到轮廓和角的特定集合来检测特定对象的整个部分。最后，根据图像描述中包含的对象部分，可以识别图像中存在的对象。

### 2.1.3 CUDA简介

CUDA(Compute Unified Device Architecture)，是显卡厂商NVIDIA推出的运算平台。CUDA™是一种由NVIDIA推出的通用并行计算架构，该架构使GPU能够解决复杂的计算问题，它包含了CUDA指令集架构（ISA）以及GPU内部的并行计算引擎。开发人员现在可以使用C语言来为CUDA™架构编写程序，C语言是应用最广泛的一种高级编程语言，所编写出的程序可以在支持CUDA™的处理器上以超高性能运行。CUDA3.0已经开始支持C++和FORTRAN。

随着显卡的发展，GPU越来越强大，而且GPU为显示图像做了优化，在计算上已经超越了通用的CPU。如此强大的芯片如果只是作为显卡就太浪费了，因此NVIDIA推出CUDA，让显卡可以用于图像计算以外的目的。

目前支持CUDA的GPU销量已逾1亿，数以千计的软件开发人员正在使用免费的CUDA软件开发工具来解决各种专业以及家用应用程序中的问题。这些应用程序从视频与音频处理和物理效果模拟到石油天然气勘探、产品设计、医学成像以及科学研究，涵盖了各个领域。

CUDA 的核心有三个重要抽象概念： [线程组](http://baike.baidu.com/view/6427102.htm)层次结构、共享存储器、屏蔽同步（ barrier synchronization），可轻松将其作为C语言的最小扩展级公开给程序员。CUDA 软件堆栈由几层组成，一个硬件驱动程序，一个应用程序编程接口(API)和它的Runtime，还有两个高级的通用数学库（CUFFT 和CUBLAS）。硬件被设计成支持轻量级的驱动和Runtime 层面，因而提高了性能。

CUDA目前支持Linux和Windows操作系统。进行CUDA开发需要依次安装驱动、toolkit、SDK三个软件。在安装目录/C/src目录下有很多的例程可以进行学习。

NVIDIA进军高性能计算领域，推出了Tesla&CUDA高性能计算系列解决方案，CUDA技术，一种基于NVIDIA图形处理器（GPU）上全新的并行计算体系架构，让科学家、工程师和其他专业技术人员能够解决以前无法解决的问题。作为一个专用高性能GPU计算解决方案，NVIDIA把超级计算能够带给任何工作站或服务器，以及标准、基于CPU的服务器集群。

CUDA是用于GPU计算的开发环境，它是一个全新的软硬件架构，可以将GPU视为一个并行数据计算的设备，对所进行的计算进行分配和管理。在CUDA的架构中，这些计算不再像过去所谓的GPGPU架构那样必须将计算映射到图形API（OpenGL和Direct 3D）中，因此对于开发者来说，CUDA的开发门槛大大降低了。CUDA的GPU编程语言基于标准的C语言，因此任何有C语言基础的用户都很容易地开发CUDA的应用程序。

由于GPU的特点是处理密集型数据和并行数据计算，因此CUDA非常适合需要大规模并行计算的领域。目前CUDA除了可以用C语言开发，也已经提供FORTRAN的应用接口，未来可以预计CUDA会支持C++、Java、Python等各类语言。可广泛的应用在图形动画、科学计算、地质、生物、物理模拟等领域。

2008年NVIDIA推出CUDA SDK2.0版本，大幅提升了CUDA的使用范围，使得CUDA技术愈发成熟。

### 2.1.4 项目概述

实习的目标是掌握基本的深度学习知识，阅读Caffe项目源码，理解Caffe的结构和工作流程，在此基础上进行调试优化。

Caffe基于自己的模型架构，通过逐层定义（layer-by-layer）的方式定义一个网络（Nets）。网络从数据输入层到损失层自下而上地定义整个模型。Caffe使用 Blob结构来存储、交换和处理网络中正向和反向迭代时的数据和导数信息：Blob 是Caffe的标准数组结构，它提供了一个统一的内存接口；Layer是Caffe模型和计算的基本单元；Net是一系列layers和其连接的集合。Blob详细描述了信息是如何在layer和net中存储和交换的。

## 2.2 Caffe模型解析

### 2.2.1 Blob的存储与交换

Blob是Caffe中处理和传递实际数据的数据封装包，并且在CPU与GPU之间具有同步处理能力。从数学意义上讲，Blob是按C语言风格连续存储的N维数组。

Caffe基于Blob存储和交换数据。为了便于优化，Blob提供统一的内存接口来存储某种类型的数据，例如批量图像数据、模型参数以及用来进行优化的导数。

Blob可根据CPU主机到GPU设备的同步需要，屏蔽 CPU/GPU混合运算在计算上的开销。主机和设备上的内存按需求分配，以提高内存的使用效率。

对于批量图像数据来说，Blob常规的维数为图像数量N \*通道数K \*图像高度H \*图像宽度W。Blob 按行为主进行存储，所以一个4维Blob中，坐标为(n, k, h, w)的值的物理位置为（（n \* K + k) \* H + h) \* W + w，这也使得最后面/最右边的维度更新最快。 Number/N是每个批次处理的数据量。批量处理信息有利于提高设备处理和交换的数据的吞吐率。Channel/K是特征维度，例如对RGB图像来说，K=3。

参数Blob的维度是根据层的类型和配置而变化的。一个卷积层中若有96个空间维度为11 x 11、输入为3通道的滤波器，那么其blob维度是96 x 3 x 11 x 11。对于一个输入是1024维（输入通道数），输出是1000维（输出通道数）的内积层/全连接层，参数 blob 维度是 1000 x 1024。

对于Blob中的数据，我们关心的是values（值）和gradients（梯度），所以一个Blob单元存储了两块数据——data和diff。前者是我们在网络中传送的普通数据，后者是通过网络计算得到的梯度。由于数据既可存储在CPU上，也可存储在GPU上，因而有两种数据访问方式： 静态方式，不改变数值；动态方式，改变数值：



这样设计的原因是Blob 使用了一个SyncedMem类来同步CPU和GPU上的数值，以隐藏同步的细节和最小化传送数据。一个经验准则是，如果不想改变数值，就一直使用常量调用，而且绝不要在自定义类中存储指针。每次操作Blob 时，调用相应的函数来获取它的指针，因为SyncedMem需要用这种方式来确定何时需要复制数据。

使用GPU时，Caffe中CPU代码先从磁盘中加载数据到Blob，同时请求分配一个GPU设备核（device kernel）以使用GPU进行计算，再将计算好的Blob 数据送入下一层，这样既实现了高效运算，又忽略了底层细节。只要所有layers 均有GPU实现，这种情况下所有的中间数据和梯度都会保留在GPU上。

以下示例确定了Blob何时会复制数据：

// 假定数据在CPU上进行初始化，我们有一个blob

const Dtype\* foo;

Dtype\* bar;

foo = blob.gpu\_data(); // 数据从CPU复制到GPU

foo = blob.cpu\_data(); // 没有数据复制，两者都有最新的内容

bar = blob.mutable\_gpu\_data(); // 没有数据复制

// ... 一些操作 ...

bar = blob.mutable\_gpu\_data(); // 仍在GPU，没有数据复制

foo = blob.cpu\_data(); // 由于GPU修改了数值，数据从GPU复制到CPU

foo = blob.gpu\_data(); //没有数据复制，两者都有最新的内容

bar = blob.mutable\_cpu\_data(); // 依旧没有数据复制

bar = blob.mutable\_gpu\_data(); //数据从CPU复制到GPU

bar = blob.mutable\_cpu\_data(); //数据从GPU复制到 CPU

### 2.2.2 Layer的计算与连接

Layer是Caffe模型的本质内容和执行计算的基本单元。Layer可以进行很多运算，如：convolve（卷积）、pool（池化）、inner product（内积），rectified-linear 和sigmoid等非线性运算，元素级的数据变换，normalize（归一化），load data（数据加载），softmax和hinge等损失计算。以卷积层运算为例，其结构如图2-2所示。

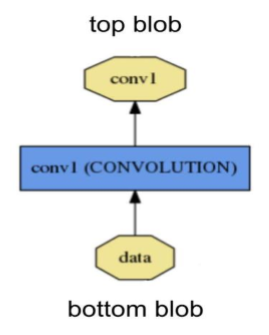


图2-2 卷积层结构

一个layer通过bottom（底部）连接层接收数据，通过top（顶部）连接层输出数据。每一个layer都定义了3种重要的运算：setup（初始化设置）、forward（前向传播）、backward（反向传播）。

Setup：在模型初始化时重置layers及其相互之间的连接。

Forward：从bottom层中接收数据，进行计算后将输出送入到top层中。

Backward：给定相对于top层输出的梯度，计算其相对于输入的梯度，并传递到bottom层。一个有参数的 layer 需要计算相对于各个参数的梯度值并存储在内部。

其中，Forward和Backward函数分别有CPU和GPU两种实现方式。如果没有实现GPU版本，那么layer将转向作为备用选项的CPU方式。总的来说，Layer承担了网络的两个核心操作：forward pass（前向传播）——接收输入并计算输出；backward pass（反向传播）——接收关于输出的梯度，计算相对于参数和输入的梯度并反向传播给在它前面的层。由此组成了每个layer的前向和反向通道。

由于Caffe网络的组合性和其代码的模块化，自定义layer是很容易的。只要定义好layer的setup（初始化设置）、forward（前向通道）和 backward（反向通道），就可将layer纳入到网络中。

### 2.2.3 Net的定义与操作

通过合成和自动微分，网络同时定义了一个函数和其对应的梯度。通过合成各层的输出来计算这个函数，执行给定的任务，并通过合成各层的后向传播过程来计算来自损失函数的梯度，从而学习任务。Caffe 模型是端到端的机器学习引擎。

准确的说，Net是由一系列层组成的有向无环（DAG）计算图，Caffe保留了计算图中所有的中间值以确保前向和反向迭代的准确性。一个典型的Net开始于 data layer——从磁盘中加载数据，终止于loss layer——计算如分类和重构这些任务的目标函数。

Net由一系列层和它们之间的相互连接构成，用的是一种文本建模语言。一个简单的逻辑回归分类器的结构和定义如图2-3所示。

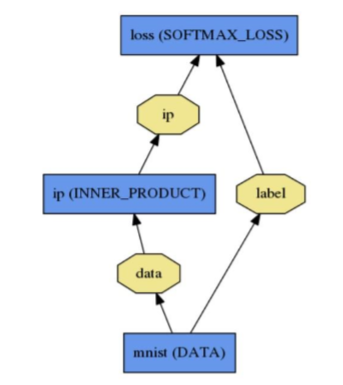


图2-3 简单逻辑回归分类器的结构

其定义如下所示：

name: "LogReg"

layer {

name: "mnist"

type: "Data"

top: "data"

top: "label"

data\_param {

source: "input\_leveldb"

batch\_size: 64

}

}

layer {

name: "ip"

type: "InnerProduct"

bottom: "data"

top: "ip"

inner\_product\_param {

num\_output: 2

}

}

layer {

name: "loss"

type: "SoftmaxWithLoss"

bottom: "ip"

bottom: "label"

top: "loss"

}

在Caffe中，Net::Init()函数进行模型的初始化。初始化主要实现两个操作：创建blobs和layers以搭建整个网络DAG图，以及调用layers的SetUp()函数。初始化时也会做另一些记录，例如确认整个网络结构的正确与否等。另外，初始化期间，Net会打印其初始化日志到INFO信息中。

### 2.2.4 模型格式

Caffe的模型是利用文本protocol buffer（prototxt）语言定义的，学习好的模型会被序列化地存储在二进制protocol buffer (binaryproto) .caffemodel文件中。 模型格式用protobuf语言定义在caffe.proto文件中。

Caffe使用Google Protocol Buffer有以下优势：按序排列时二进制字符串尺寸最小，高效序列化，易读的文本格式与二进制版本兼容，可用多种语言实现高效的接口，尤其是C++和Python。这些优势造就了Caffe模型的灵活性与扩展性。

## 2.3 前向传播与反向传播

前向传播与反向传播是一个网络最重要的计算过程，其示意图如图2-4所示。

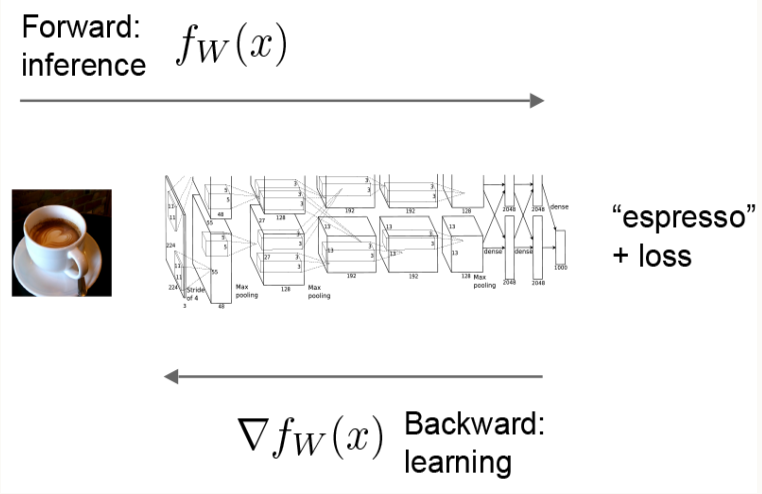


图2-4 前传和反传示意图

### 2.3.1 前向传播

前向传播（forward）过程为给定的待推断的输入计算输出。在前传过程中，Caffe组合每一层的计算以得到整个模型的计算“函数”。本过程自底向上进行。以最简单的逻辑分类回归器为例，如图2-5所示，数据x通过一个内积层得到g(x)，然后通过softmax层得到h(g(x))，通过softmax loss得到fw(x)。

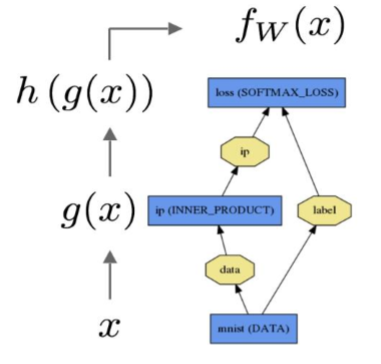


图2-5 前向传播示意图

### 2.3.2 反向传播

反向传播（backward）过程根据损失来计算梯度从而进行学习。在反传过程中，Caffe通过自动求导并反向组合每一层的梯度来计算整个网络的梯度。这就是反传过程的本质，本过程自顶向下进行。

反向传播过程以损失开始。以最简单的逻辑分类回归器为例，如图2-6所示，根据输出计算梯度，再根据链式准则，逐层计算出模型其余部分的梯度。有参数的层，例如INNER\_PRODUCT层，会在反传过程中根据参数计算梯度。

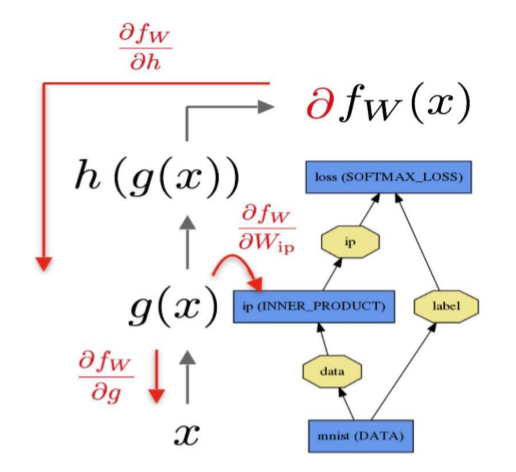


图2-6 反向传播示意图

### 2.3.3 Caffe中前传和反传的实现

Caffe中前传和反传的实现方法：

Net::Forward()和Net::Backward()方法实现网络的前传和后传，而 Layer::Forward()和Layer::Backward()计算每一层的前传后传。

每一层都有forward\_{cpu, gpu}()和backward\_{cpu, gpu}方法来适应不同的计算模式。由于条件限制或者为了使用便利，一个层可能仅实现了CPU或者 GPU模式。

Solver优化一个模型，首先通过调用前传来获得输出和损失，然后调用反传产生模型的梯度，将梯度与权值更新后相结合来最小化损失。Solver、网络和层之间的分工使得Caffe可以模块化并且开源。

## 2.4 损失

与大多数的机器学习模型一样，在Caffe中，学习是由一个损失函数（通常也被称为误差、代价或者目标函数）驱动的。一个损失函数通过将参数集（即当前的网络权值）映射到一个可以标识这些参数“不良程度”的标量值来学习目标。因此，学习的目的是找到一个网络权重的集合，使得损失函数最小。

损失函数在统计学中是一种衡量损失和误差程度的函数，它一般包括损失项(loss term)和正则项(regularization term)，其计算公式为。比较常见的损失项有平方损失，其常用在回归问题；对数损失，其常用在分类问题。加入正则项的目的是减小权重的幅度，防止过度拟合，常用的有L1范数正则化（L1-regularization）和L2范数正则化（L2-regularization）。

在Caffe中，损失是通过网络的前向计算得到的。每一层由一系列的输入blobs (bottom)，然后产生一系列的输出blobs (top)。这些层的某些输出可以用来作为损失函数。

对于含有多个损失层的网络（例如，一个网络使用一个softMaxWithLoss输入分类并使用EuclideanLoss层进行重构），损失权值可以被用来指定它们之间的相对重要性。

按照惯例，有着Loss后缀的Caffe层对损失函数有贡献，其他层被假定仅仅用于中间计算。然而，通过在层定义中添加一个loss\_weight:<float>字段到由该层的top blob，任何层都可以作为一个loss。对于带后缀Loss的层来说，其对于该层的第一个top blob含有一个隐式的loss\_weight:1；其他层对应于所有top blob 有一个隐式的loss\_weight:0。

然而，任何可以反向传播的层，可允许给予一个非0的loss\_weight，例如，如果需要，对网络的某些中间层所产生的激活进行正则化。对于具有相关非0损失的非单输出，损失函数可以通过对所有blob求和来进行简单地计算。

在Caffe中最终的损失函数可以通过对整个网络中所有的权值损失进行求和计算获得，正如以下的伪代码：

loss := 0

for layer in layers:

for top, loss\_weight in layer.tops, layer.loss\_weights:

loss += loss\_weight \* sum(top)

## 2.5 Solver

### 2.5.1 Solver简介

Solver通过协调Net的前向推断计算和反向梯度计算（forward inference and backward gradients），来对参数进行更新，从而达到减小损失（loss）的目的。Caffe 模型的学习被分为两个部分：由Solver进行优化、更新参数，由Net计算出损失（loss）和梯度（gradient）。

Caffe支持的solvers包括：

Stochastic Gradient Descent (type: "SGD")，随机梯度下降

AdaDelta (type: "AdaDelta")

Adaptive Gradient (type: "AdaGrad")，自适应梯度

Adam (type: "Adam")

Nesterov’s Accelerated Gradient (type: "NAG")

RMSprop (type: "RMSProp")

Solver的作用包括：

用于优化过程的记录、创建训练网络（用于学习）和测试网络（用于评估）；

通过forward和backward过程来迭代地优化和更新参数；

周期性地用测试网络评估模型性能；

在优化过程中记录模型和solver状态的快照（snapshot）。

在每一次迭代过程中：

调用Net的前向过程计算出输出和loss；

调用Net的后向过程计算出梯度（loss对每层的权重w和偏置b求导）；

根据2.5.2所讲的Solver方法，利用梯度更新参数；

根据学习率（learning rate）、历史数据和求解方法更新Solver的状态,使权重从初始化状态逐步更新到最终的学习到的状态。Solver的运行模式有 CPU/GPU两种模式。

### 2.5.2 Solver方法

Solver方法用于最小化损失（loss）值。给定一个数据集D，优化的目标是 D中所有数据损失的均值（即平均损失）取得最小值，其计算公式如式(1)。

()

其中 是数据中项的损失，是正则项，权重为。当数据量D很大时，在每一次迭代中，我们采用数据集的一个随机子集来近似代替，其数据量远小于整个数据集（𝑁 ≪ |𝐷|），计算公式如式(2)。

()

在前向过程（forward）中计算（即loss），在反向过程（backward）中计算（即梯度gradient）。根据误差梯度、正则项的梯度以及其他方法的特定项来计算参数更新量∆𝑊。

**（1） SGD**

随机梯度下降（Stochastic gradient descent, type:“SGD”）利用负梯度𝛻𝐿(𝑊)和上一次权重的更新值的线性组合来更新权重W。学习率（learning rate）α是负梯度的权重，动量（momentum）μ是上一次更新值的权重。

如公式（3）、（4）所示，根据上一次计算的更新值和当前权重来计算本次的更新值和权重：

()

()

学习的超参数α和μ需要一定的调整才能达到最好的效果。设定学习率α 和动量μ的经验法则：将学习速率初始化为α ≈ 0.01 = ，然后在训练中当 loss达到稳定时，将α除以一个常数（例如10），将这个过程重复多次。对于动量一般设置为μ = 0.9，μ使权重 weight的更新更为平缓，使学习过程更为稳定、快速。

**(2) AdaDelta**

AdaDelta方法是一种“鲁棒的学习率方法”，同SGD一样是一种基于梯度的优化方法。其更新方程如式（5）、（6）、（7）所示：

()

()

()

**(3) AdaGrad**

自适应梯度下降方法（Adaptive gradient）跟随机梯度下降（Stochastic gradient）一样是基于梯度的优化方法。给定之前更新的信息，对于𝑡′ ∈ {1,2,…,𝑡}，权重W中每个元素i的更新如式（8）所示：

()

实际操作中，对于权重𝑊 ∈ ，自适应梯度（AdaGrad）的实现只需要𝑂(𝑑)额外空间存储历史梯度信息，而不是𝑂(𝑑𝑡)的存储空间来存储每一个单独的历史纪录。

**(4) Adam**

Adam也是一种基于梯度的优化方法。它包含一对自适应时刻估计变量 (𝑚𝑡,𝜇𝑡)，可以看做是AdaGrad的一种泛化形式。其更新方程如式（9）、（10）、（11）所示。

()

()

()

Caffe中使用momemtum,momentum2,delta分别代表、和。

**(5) NAG**

Nesterov提出的加速梯度下降（Nesterov’s accelerated gradient）是凸优化的一种最优算法，其收敛速度可以达到，而不是。尽管在使用Caffe 训练深度神经网络时很难满足收敛条件（例如，由于非平滑或非凸），但实际中NAG对于某些特定结构的深度学习模型仍是一个非常有效的方法。

其权重weight更新参数与随机梯度下降非常相似，如式（12）、（13）所示：

()

()

与SGD的不同之处在于梯度𝛻𝐿(𝑊)项中取值不同：在NAG中，我们取当前权重和动量之和的梯度；在SGD中，只是简单的计算当前权重的动量。

**(6) RMSProp**

Tieleman在Coursera课程中提到的RMSprop方法同样是一种基于梯度的优化方法（同SGD类似）。更新方程如式（14）、（15）所示：

()

()

如果梯度更新值产生振动，则让梯度减小（乘以1−𝛿），否则增加δ。δ默认值是0.02。

## 2.6 Layer Cataloge

为了创建一个Caffe模型，我们需要在一个protocol buffer(prototxt)文件中定义模型的结构。在Caffe中，层和相应的参数都定义在caffe.proto文件里。

### 2.6.1 视觉层Vision Layers

视觉层的输入与输出均为图像。一个典型的图像通常为单通道的灰度图或三通道的RBG彩色图。图像还有一个广义的概念，明显特性来自于空间结构:高和宽通常均大于1，而通道数不限，类似结构的数据均可理解为图像。这种结构可以帮助Caffe的层决定如何处理输入数据，具体来说，大多数视觉层通常是在输入数据的某块区域执行特定操作来产生对应的输出。相反的，其它类型的层通常会忽略空间结构而把输入图像看作是一个维度为chw的“单个大向量”。

**(1) 卷积层Convolution Layer**

卷积层使用一系列可训练的卷积核对输入图像进行卷积操作，每组卷积核生成输出图像中的一个特征图。在Caffe中，卷积操作做了优化，变成了一个矩阵相乘的操作，其中有两个比较主要的函数是im2col以及col2im。传统卷积与矩阵相乘版本的对比如图2-7所示。

传统卷积过程如图2-8所示。第一列是一个7\*7\*3的特征图；第二列是第一个卷积核，大小为3\*3\*3；第三列是第二个卷积核，大小为3\*3\*3；第四列得到了两个输出特征图，第一个是由原特征图和第一个卷积核卷积得到的，第二个是由原特征图和第二个卷积核卷积得到的。第二列的卷积核的第一个3\*3模板在第一个特征图上进行加权迭代，得到值为-3；第二个3\*3模块在第二个特征图对应位置，进行同样的操作，得到值为4；第三个得到值为0；最后加上偏置1，即：-3+4+0+1=2。按这样的计算规则，卷积核在原特征图上进行滑动遍历，最终得到了第四列第一个输出特征图。第三列的卷积核计算同上，得到了第四列第二个输出特征图。第一个卷积核进行了3\*3\*3次乘法，两个卷积核共进行了2\*3\*3\*3次乘法。第一个卷积核有3\*3\*3个参数，加上一个偏置，共3\*3\*3+1=28个参数。所以，两个卷积核共28\*2=56个参数。

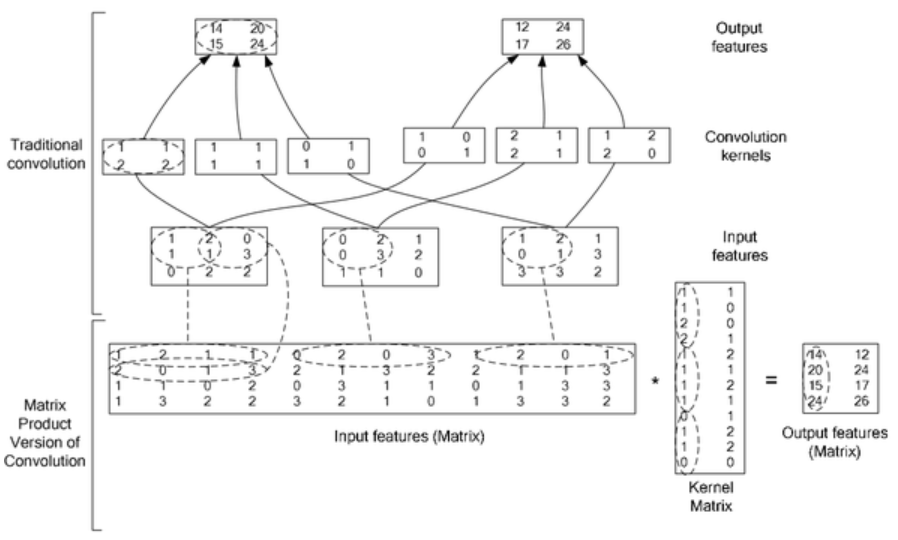


图2-7 传统卷积与矩阵相乘版本对比

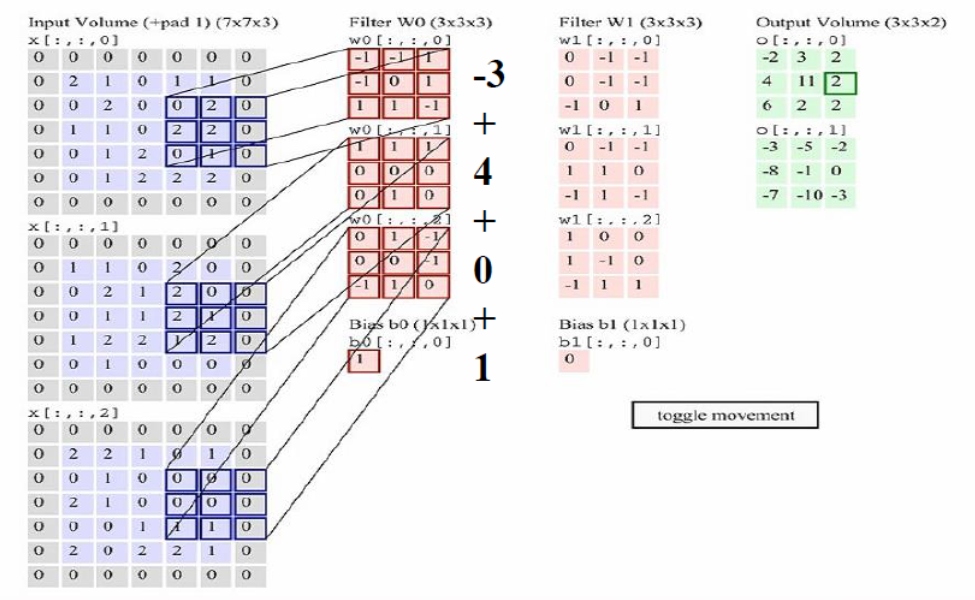


图2-8 传统卷积过程

在Caffe中，卷积时先将输入特征图和卷积核都转化为矩阵，计算完成后再还原。如图2-9所示，将输入特征图中，卷积部分的那个区域中数据(原特征图中)，转变为一列，如果channels为1，则列长为k\_height\_\*k\_width，否则为in\_channels\*k\_height\_\*k\_width。当卷积核移动一次，对应的区域数据又形成了新的一列，卷积核一共移动out\_height\_\*out\_width次，这样输入特征图矩阵的宽度就为out\_height\_\*out\_width。所以，就形成了一个(in\_channels\*k\_height\_\*k\_width)\*(out\_height\_\*out\_width)的矩阵。同上，得到一个(out\_channles)\*(in\_channels\*k\_height\_\*k\_width)的卷积核矩阵。用卷积核矩阵\*输入特征图矩阵，得到(out\_channles)\*(out\_height\_\*out\_width)大小的输出特征图矩阵，还原后得到out\_channles个out\_height\_\*out\_width大小的特征图。

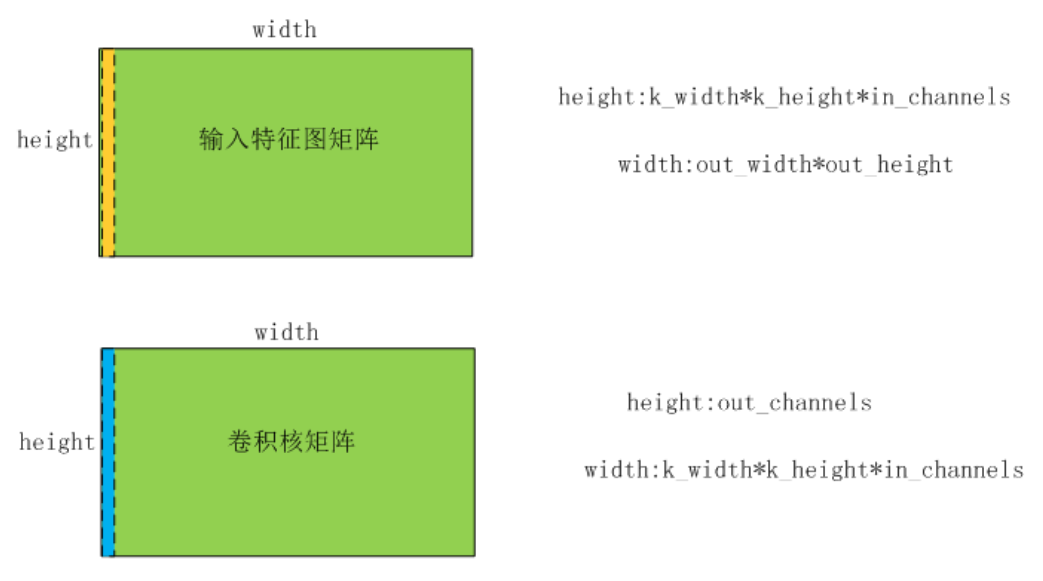


图2-9 Caffe中卷积

在Caffe中，num\_output (c\_o)指定卷积核的数量；kernel\_size (或kernel\_h和kernel\_w)指定卷积核的高度和宽度；bias\_term [default true]指定是否给卷积输出添加偏置项；pad (或pad\_h和pad\_w) [default 0]指定在输入图像周围补0的像素个数；stride (或stride\_h和stride\_w) [default 1]指定卷积核在输入图像上滑动的步长；group (g) [default 1]指定分组卷积操作的组数，默认为1即不分组。如果 g >1,我们可以将卷积核的连接限制为输入数据的一个子集。具体地说, 输入图像和输出图像在通道维度上分别被分成g个组, 输出图像的第 i组只与输入图像第i组连接（即输入图像的第i组与相应的卷积核卷积得到第i组输出）。

输入为n \* c\_i \* h\_i \* w\_i，输出为n \* c\_o \* h\_o \* w\_o，式中h\_o = (h\_i + 2 \* pad\_h - kernel\_h) / stride\_h +1，w\_o 计算方式类似。

Caffe中的卷积层使用一系列可训练的卷积核对输入图像进行卷积操作，每组卷积核生成输出图像中的一个特征图。

**(2) 池化层Pooling Layer**

池化层一般在网络中连接在卷积层之后，做采样操作，目的是为了进一步缩小feature map，同时也能增大神经元的视野。在Caffe中，池化层属于视觉层的一部分，池化层的相关操作比较少，在Caffe的自带模式下只有最大值池化（Max pooling）和均值池化（Mean poooling）两种。其中Max pooling和Mean poooling两种池化模式分别如图2-10、2-11所示。

在Caffe中，kernel\_size(或kernel\_h和kernel\_w)指定池化窗口的高度和宽度；pad (或pad\_h和pad\_w) [default 0]指定在输入图像周围补0的像素个数；stride (或stride\_h和stride\_w) [default 1]指定池化窗口在输入数据上滑动的步长。输入为n \* c \* h\_i \* w\_i，输出为n \* c \* h\_o \* w\_o, 式中 h\_o 和 w\_o 的计算方法与卷积层相同。

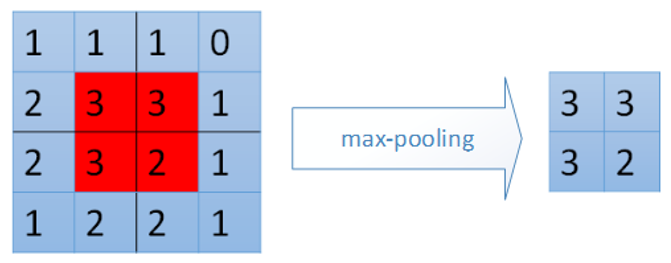


图2-10 Max pooling

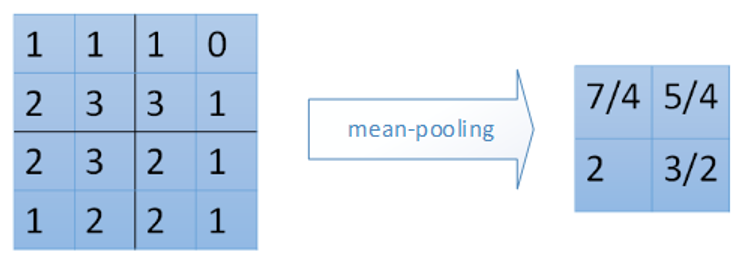


图2-11 Mean pooling

**(3) 局部响应值归一化层LRN Layer**

局部响应值归一化层通过对输入数据的局部归一操作执行了一种“侧抑制”的机制。在ACROSS\_CHANNELS（在通道之间进行规范化）模式下，局部区域沿着临近通道延伸（而非在特征图的平面内），而没有空间扩展，即局部区域的形状为 local\_size x 1 x 1（local\_size指参与求和的通道数）。在 WITHIN\_CHANNEL（在通道内进行规范化）模式下，局部区域在各自通道内部的图像平面上延伸，即局部区域的形状为1 x local\_size x local\_size（local\_size指参与求和的方形区域的边长）。每个输入值除以以实现归一化，式中，是尺度参数，是指数参数，n是局部区域的大小，在以当前输入值为中心的区域内计算加和（如有需要，需在边缘补零）。

**(4) im2col Layer**

Im2col是一个辅助操作，用来实现图像到“列向量”的转换。Im2col 用在Caffe早期卷积的矩阵乘法中，即将所有图像块组成一个矩阵。在 Caffe中，执行卷积操作时，将图像中与卷积核作用的图像块写成列向量，然后将这些列向量按行的方向依次排开组成一个二维的矩阵，同时，每组卷积核写成一个行向量，多个输出通道对应的多组卷积核按列的方向依次排开形成一个二维矩阵，这样，卷积操作就转化为矩阵乘法操作。

### 2.6.2 损失层Loss Layers

Loss设置了一个损失函数用来比较网络的输出和目标值，通过最小化损失来驱动网络的训练。网络的损失通过前向操作计算，网络参数相对于损失函数的梯度则通过反向操作计算。

**(1) Softmax损失**

softmax损失层一般用于计算多类分类问题的损失，在概念上等同于softmax 层后跟随一个多变量Logistic回归损失层(multinomial logistic loss)，但能提供数值上更稳定的梯度。

Softmax的损失函数使用的是对数损失函数，其中k为该样本的label（即该样本对应的正确输出，比如我们要识别的图片是数字7，则k=7，选择softmax的第7个输出值来计算loss）。一般我们进行训练时一批图片有多张，比如batch size = 16，则。

由于我们的输入为，则。若loss对每个输入z求导，则有。

**(2) 平方和/欧式损失Sum-of-Squares / Euclidean**

平方和损失的损失函数如式（16）所示：

()

欧式距离损失函数如式（17）所示，其中，表示样本标签，表示神经网络的实际输出。

()

### 2.6.3 激活层Activation Layers

在激活层中，对输入数据进行激活操作（实际上就是一种函数变换），是逐元素进行运算的。输入一个底层blob，输出一个尺寸相同的顶层blob。在运算过程中，没有改变数据的大小，即输入和输出的数据大小是相等的。输入数据尺寸为 n \* c \* h \* w，输出数据尺寸也为 n \* c \* h \* w。Caffe中的激活函数有Sigmoid、tanh、relu等。

**(1) Sigmoid函数**

Sigmoid函数表达式如式（18）所示，它将输入值映射到[0,1]区间内，其函数图像如图2-12所示。

()

其反向传播公式如式（19）所示，

()

但Sigmoid函数有一个十分致命的缺点，就是它的导数值很小，sigmoid函数导数图像如图2-13所示。由图2-13可知，sigmoid函数导数最大值只有1/4，而且在输入很大或者很小的时候，其导数趋近于0。这直接导致的结果是在反向传播中，梯度会衰减的十分迅速，导致传递到前边层的梯度很小甚至消失，训练会变得十分困难。

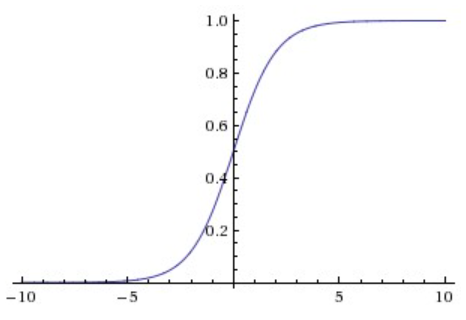


图2-12 Sigmoid函数图像

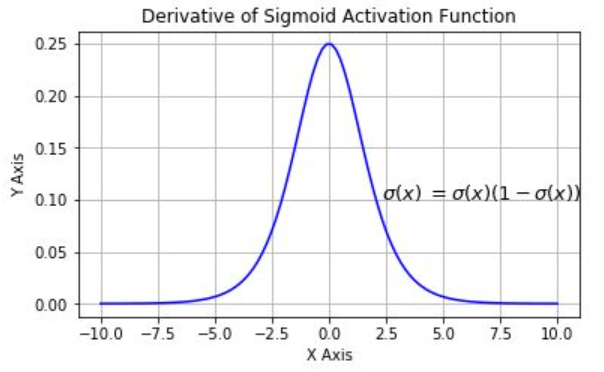


图2-13 Sigmoid函数导数图像

**(2) ReLU函数**

ReLU函数为现在使用比较广泛的激活函数，其表达式如式（20）所示：

()

给定一个输入数据，当x >= 0时，ReLU层的输出为x，当 x<0时，输出为0。其函数图像如图2-14所示。其反向传播公式如式（21）所示。

()

ReLU函数主要有以下2点优点：

ReLU函数的计算十分简单，前向计算时只需将输入值和一个阈值（这里为0）进行比较，即可得到输出值。在反向传播时，ReLU函数的导数为，较之其它激活函数更为简单。

由于ReLU函数的导数为，所以反向传播时梯度要么为0，要么不变，因此梯度的衰减很小，即使网络层数很深，前边层的收敛速度也不会很慢。

ReLU函数也有很明显的缺点，就是在训练的时候，网络很脆弱，很容易出现很多神经元值为0，从而再也训练不动。一般我们将学习率设置为较小值来避免这种情况的发生。

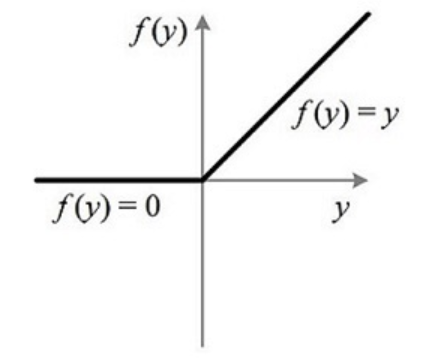


图2-14 ReLU函数图像

**(3) tanh函数**

tanh函数表达式如式（22）所示：

()

其输出值的取值范围为[-1,1]，函数图像如图2-15所示。

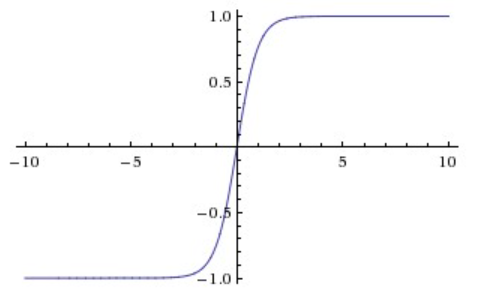


图2-15 tanh函数图像

### 2.6.4 数据层Data Layers

数据通过数据层进入caffe网络：数据层处于网络的最底层，数据可以从高效率的数据库中读取（如LevelDB或LMDB），也可以直接从内存中读取，若对读写效率要求不高也可以从硬盘上的HDFT文件或者普通的图片文件读取。

当从数据库中读取数据时，由source指定数据库文件的路径；batch\_size指定网络单次输入数据的数量；rand\_skip指定跳过开头的 rand\_skip \* rand(0,1)个数据，通常在异步随机梯度下降法里使用；backend [default LEVELDB]指定选择使用LevelDB还是LMDB。

当从内存中读取数据时，由batch\_size, channels, height；width指定从内存中读取的输入数据块的尺寸。

当从HDFT文件中读取数据时，由source指定文件路径；batch\_size指定网络单次输入数据的数量。

当从图片文件中读取数据时，由source: text指定文件的路径名，该text文件的每一行存储一张图片的路径名和对应的标签；batch\_size指定打包成batch 的图片数量；new\_height和new\_width指定输入的图片将会被调整成的尺寸。

### 2.6.5 普通层Common Layers

**(1) 内积/全连接 InnerProduct Layer**

全连接层的每一个结点都与上一层的所有结点相连，用来把前边提取到的特征综合起来。由于其全相连的特性，一般全连接层的参数也是最多的。

正向传导时输出数据等于输入数据乘上权重，如果有偏置项就再加上偏置项。正向传导与反向传导的公式分别如式（23）、（24）所示：

()

()

以如图2-16的简单网络为例说明推导过程如下：

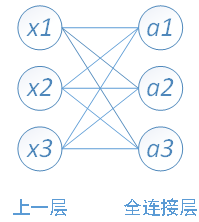


图2-16 示例

其中x1、x2、x3为全连接层的输入，a1、a2、a3为输出，根据推导公式，有

()

改写成矩阵形式如式（26）所示：

+ ()

**(2) 变形 Reshape Layer**

Reshape层在不改变数据的情况下改变输入blob的维度，处理过程只在输入 blob上进行，没有进行数据的拷贝。输出数据的维度通过参数ReshapeParam设定，可以使用正数直接指定输出blob的相应维度，也可以使用两个特殊的值来设定维度：

0表示从层的底层blob中直接取相应的维度，即不改变对应的维度。例如，在参数中设置第一个维度为0，底层blob在第一个维度上是2，则输出的顶层blob的第一个维度也是2。

-1表示用其它的维度计算该维度的值。对应的维度通过保blob中数据的总个数不变来计算。因此，在reshape操作中最多设定一个-1。

**(3) 连结 Concat Layer**

连结层的输入为n\_i \* c\_i \* h \* w，其对于第i个输入blob，i的取值为{1,2,...,K}。当axis =0时，表示沿着样本个数的维度串联，输出为(n\_1 + n\_2 + ... + n\_K) \* c\_1 \* h \* w（所有输入blob在通道上的维度c\_i 需相同）。当axis =1时，表示沿着通道维度串联，输出为n\_1 \* (c\_1 + c\_2 + ... + c\_K) \* h \* w（所有输入blob在样本个数上的维度n\_i需相同）。

# 3 实习工作总结

实习涉及到了一个优秀的深度学习框架Caffe，它的结构清晰，整体框架具有灵活性和扩展性，代码可读性高。对我来说，在从新手入门深度学习的过程中，能够以Caffe作为敲门砖，让我收获颇丰。在这一个多月的实习生活中，我不仅学习到了深度学习的基本知识和cuda的相关知识，也在阅读Caffe代码的过程中体验到了编程的艺术与代码之美。

实习与上课的最大区别是不再有人主动教授你知识，所有的相关知识都需要自己通过各种方式主动去获取。没有了老师的考核与督促，这就需要自己有更高的自律性和自制力，能够合理安排时间，不断丰富自己的知识领域。只有这样，才能在实习期间有所收获，不枉匆匆流逝的时光。

刚开始阅读Caffe代码时，觉得十分头疼，因为它不仅涉及许多深度学习的相关知识和编程技巧，而且代码量很大，因此很担心自己能否完成实习任务。但正如古语所云：书山有路勤为径，学海无涯苦作舟。通过网上各位高手写的Caffe学习指南的指导，老师和实验室学长的点拨，以及自己主动“充电”学习相关知识，一步步慢慢迈入深度学习的大门，让自己对深度学习有了一个整体初步的了解。

总而言之，这一个多月的实习对我的帮助是非常大的。它不仅让我学习到了一些新鲜有趣的科学知识，丰富了自己的知识视野，也锻炼了自己的自制力和自律性，这是成功不可或缺的能力。同时，在此过程中我也收获了一些平时在课堂上收获不到的东西，领略了学术研究的严谨与认真。这些都将是我人生中一次难得的经验与财富。

# 4 实习问题解答

问题1：

在Caffe中GPU的加速原理。

回答：

这里默认GPU加速是指NVIDIA的cuda加速。CPU是中央处理单元，GPU是图形处理单元。GPU由上千个流处理器(core)作为运算器，执行采用单指令多线程(SIMT)模式。相比于单核CPU（向量机）流水线式的串行操作，虽然GPU单个处理器计算能力很弱，但是通过大量线程进行同时计算，在数据量很大时会带来较为可观的加速效果。

具体到Caffe，利用GPU加速主要是在conv（卷积）过程上。卷积过程同理可以像向量加法一样通过cuda实现并行化。具体的方法很多，不过最好的还是利用FFT（fast Fourier transform，快速傅里叶变换）进行快速卷积。NVIDIA提供了cufft库实现FFT，复数乘法则可以使用cublas库里的对应的level3的cublasCgemm函数。

GPU加速的基本准则就是“人多力量大”。卷积操作说到底主要问题就是计算量大，但是却可以比较有效的拆分成并行问题。以一个层的filter操作为例，假设某一层有n个filter，每一个需要对上一层输入过来的map进行卷积操作。那么，这个卷积操作并不需要按照线性的流程去做，每个滤波器互相之间并不影响，因此可以同时做，然后生成了n张新的谱之后再继续接下来的操作。既然可以并行，那么同一时间处理单元越多，理论上速度优势就会越大。所以，处理问题就变得很简单粗暴，就像NV那样，暴力增加显卡单元数（当然，显卡的架构、内部数据的传输速率、算法的优化等等也都很重要）。

GPU主要是针对图形显示及渲染等技术的出众，而其中的根本是因为处理矩阵算法能力的强大，刚好Caffe网络中中涉及大量的卷积，也就是矩阵乘法等，所以在这方面具有优势。

问题2：

Caffe的优化方法。

回答：

优化Caffe的方法有优化代价函数：交叉熵代价函数一般来说都比二次方程代价函数效果要好，因为其梯度下降的学习速率是差别越大，学得越快；优化神经网络层数以及每一层的节点数选择。Yoshua Bengio在Quora上说，对于NN的layer的数量，只需要增加layer的数量然后看test error是否增加。隐含层每一层的节点数都一样的话，一般效果会比节点数慢慢变小或变大要表现的好。