浮游动物识别分类——深度学习

Dai Jialun　　Wu Bin

July 5, 2015

1. 问题

深度学习使用的数据库基本都是 ImageNet，因此使用 ImageNet 与 ZooScan 的训练样本进行比较。

小数据 ZooScan 的训练样本有 13 类（包含负样本），总共有 9460 张图片。ImageNet 的 ILSVRC 2013 的训练样本有 200 类，总共 395909 张图片。

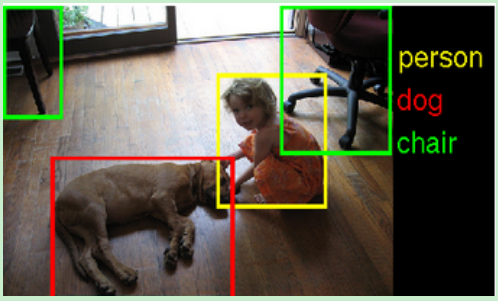
分类 将浮游动物分为不同种类，且种类数目较少，只有 10 种。

数据的标注 ZooScan 的训练图像中只有单个物体，即 single-label，且背景为空白。ImageNet 的训练图像基本都用多个不同物体的标注，即 multiple-label。

Comparative scale

		PASCAL VOC 2012	ILSVRC 2013
Number of object classes		20	**200**
Training	Num images	5717	395909
	Num objects	13609	345854
Validation	Num images	5823	20121
	Num objects	13841	55502
Testing	Num images	10991	40152
	Num objects	---	---

(a)



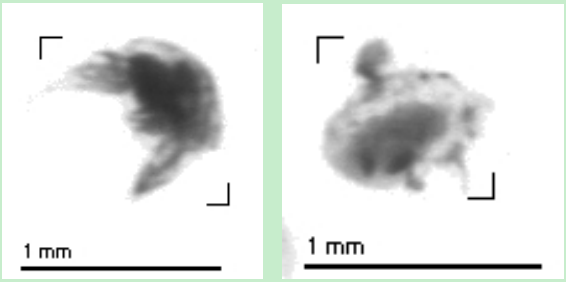
(b)

Figure 1: ILSVRC 2013

2. 可用方案

2.1 开源软件

Caffe Caffe 是一个清晰而高效的深度学习框架，由表达式、速度和模块化组成，其作者是博士毕业于 UC Berkeley 的贾扬清。它具有有开放性架构、扩展代码、速度与社区化的优点。Caffe 是纯粹的 C++/CUDA 架构，支持命令行、Python 和 MATLAB 接口，而且可以在 CPU 和 GPU 直接无缝切换。



(a) (b)

Figure 2: Zooplankton

cxxnet 一个机器学习项目，其具有轻量而齐全的框架、强大统一的并行计算接口和易于扩展的代码结构。cxxnet 平衡了 python 为主的编程效率和以 c++ 为核心的为代表的追逐性能，使得我们在不损失效率的前提下可以通过模板编程技术允许开发者编写和 matlab/numpy 类似的代码，并且在编译时自动展开成优化的 kernel。另外一些特性：CuDNN 的支持、及时更新的最新技术、Caffe 模型转换和方便的语言接口。

Cuda-convnet 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。Cuda-convnet 是 Alex Krizhevsky 公开的一套 CNN 代码，运行于 Linux 系统上，使用 GPU 做运算；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。(据说 Alex 的这套代码还是不太友好，按理说应该给出一个 IO 的标准，供后来人测试其他数据集的。)

Theano 提供了在深度学习数学计算方面的 Python 库，让你可以有效定义、优化和评价包含多维数组的数学公式。它整合了 NumPy 矩阵计算库、可以运行在 GPU 上、提供良好的算法上的扩展、在优化方面进行加速和稳定与动态 C 代码的生成等。

OverFeat 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。主要使用 C++ 编写的库来运行 OverFeat 卷积网络，可用数据增强来实现提高分类结果。

Torch7 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。

2.2 网络模型

在图像识别与分类上，主要使用卷积神经网络（Convolutional Neural Network, CNN）的网络模型来实现的。

AlexNet AlexNet[1] 是 Hinton 与其学生为了回应别人对于 deep learning 的质疑而将 deep learning 用于 ImageNet 的 ILSVRC 2012 上，结果优于当时最优的水平。该模型训练了一个深度卷积神经网络来完成分类和检测任务。AlexNet 包含 5 层卷积层和 3 层全连接层。

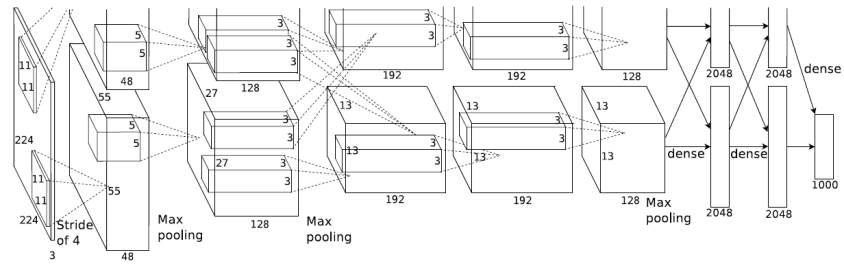


Figure 3: AlexNet

OverFeat OverFeat[3] 由 CILVR Lab，New York University 开发的，基于卷积网络的图像识别和特征提取系统。OverFeat 的选举网络是由 ImageNet 1K 数据集训练的，参加了 ILSVRC 2013 分类、定位与检测竞赛。在 Overfeat 主要展示了多尺度和扫描窗口可以有效应用在卷积网络上。而且，不同的任务可以通过一个共享的学习网络来实现。

Layer	1	2	3	4	5	6	7	Output 8
Stage	conv + max	conv + max	conv	conv	conv + max	full	full	full
# channels	96	256	384	384	256	4096	4096	1000
Filter size	7x7	7x7	3x3	3x3	3x3	-	-	-
Conv. stride	2x2	1x1	1x1	1x1	1x1	-	-	-
Pooling size	3x3	2x2	-	-	3x3	-	-	-
Pooling stride	3x3	2x2	-	-	3x3	-	-	-
Spatial input size	221x221	72x72	33x33	21x21	15x15	5x5	1x1	1x1

Figure 4: overfeat

NIN NIN（Network In Network）[2] 是 Learning and Vision Research Group，National University of Singapore 所使用的深度学习框架。他们团队通过在深度网络中建立更复杂的微型神经网络来提取数据。NIN 网络包含了三个多元卷积层和一个全局平局 pooling 层。另外，他们团队提出“adaptive non-parametric rectification” 方法来提高准确率。

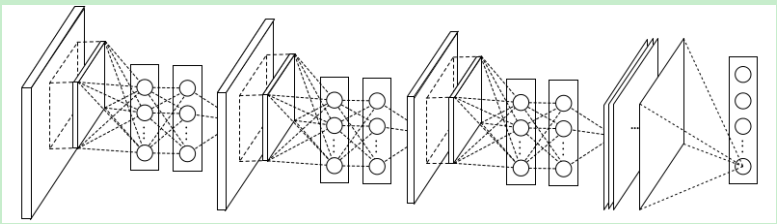


Figure 5: NIN

GoogLeNet GoogLeNet[5] 是 Google 团队在 ILSVRC 2014 上，使用的深度学习网络模型，来完成分类和检测的任务。在 GoogLeNet 模型中，提出了 Inception 模型，即采用不同尺度的卷积核来处理问题。另外，该网络最大的特点就是提升了计算资源的利用率，即在网络所需计算不变的前提下，提升了网络的宽度和深度。最后，使用了基于 Hebbian Principle 和多尺寸处理来提高准确率。GoogLeNet 包含 pooling 层在内，总共是 27 层的模型。

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 6: GoogLeNet

VGGNet VGGNet[6] 是 Visual Geometry Group 在 ILSVRC 2014 上，使用的深度学习网络模型。VGGNet 与 AlexNet 的网络模型很相似，同为 5 组卷积层，3 个全连接层。在 VGGNet 中，前 5 组的卷积层各有不同配置，即模型中有 A~E，卷积层数从 8 到 16 递增。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 7: VGGNet

PCANet PCANet 是一个基于 CNN 的简化 Deep Learning 模型。经典的 CNN 存在的问题是参数训练时间过长且需要特别的调参技巧。PCANet 模型是一种训练过程简单，且能适应不同任务、不同数据类型的网络模型。卷积核是直接通过 PCA 计算得到的，而不是像 CNN 一样通过反馈迭代得到的。

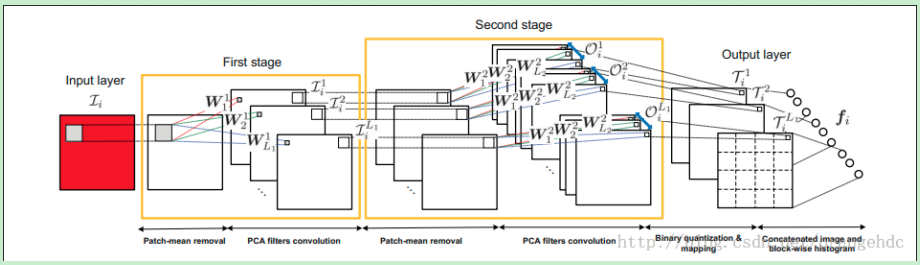


Figure 8: PCANet

Deep Fisher Net Deep Fisher Net[4] 是 Visual Geometry Group, University of Oxford 在 ILSVRC 2013 分类上使用的模型，结合了两种深度框架：the deep Fisher vector network 和 the deep convolutional network。

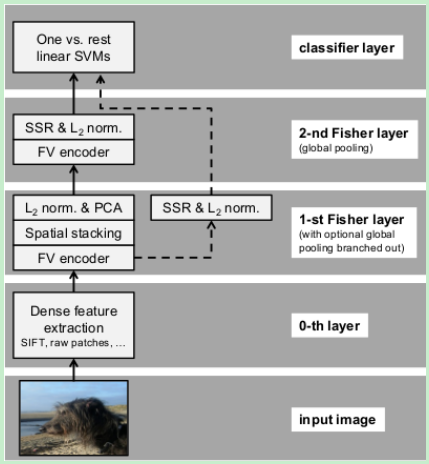


Figure 9: Deep Fisher Net

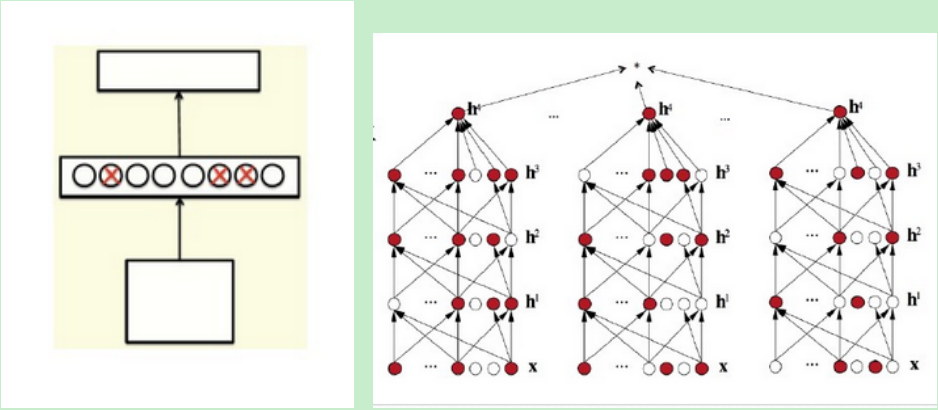
MP-CNN

3. 优化技巧

3.1 Dropout

应用在全连接层，是一种引入噪声机制，避免过拟合的有效正规化方法。适用于样本数目数 ≪ 模型参数。对于每一个隐层的 output，以 50% 的概略将它们设置为 0，不再对与 forward 或 backward

过程起作用，即迫使每个神经元不依赖某些特定神经元独立工作，从而学到更有用的特征。



(a) (b)

Figure 10: ILSVRC 2013

3.2 Pooling

基于图像的局部信息与全局信息，可认为一个图像区域有用的特征极有可能在另一个区域同样适用。而且为了描述大的图像，可以对不同位置的特征进行聚合统计。

在卷积神经网络中，经常在卷积层后加入池化操作（pooling），通过池化来降低卷积层输出的特征向量维数，即用更高层的抽象表示图像特征，实现信息的融合。常见的池化操作分为为平均池化 mean pooling 和最大池化 max pooling。

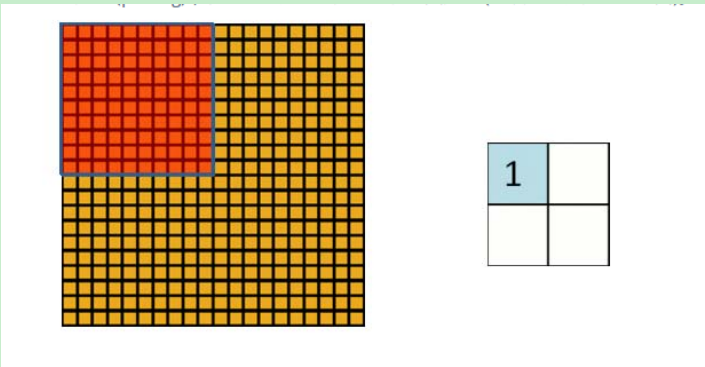


Figure 11: Pooling

3.3 PCA

PCA（Principal Component Analysis），主成分分析，不仅仅是对高维数据进行降维，更重要的是经过降维去除了噪声，发现了数据中的模式。PCA 把原先的 n 个特征用数目更少的 m 个特征取代，

新特征是旧特征的线性组合，这些线性组合最大化样本方差，尽量使新的 m 个特征互不相关。从旧特征到新特征的映射捕获数据中的固有变异性。对每个 RGB 图像进行一个 PCA 变换，完成了去噪功能与保证图像多样性。可认为，加入了一个随机尺度因子，每一轮重新生成一个尺度因子，保证了同一副图像中在显著特征上有一定范围的变换，降低 overfitting。

PCA 过程：

1. 特征中心化。即每一维的数据都减去该维的均值。

2. 计算数据的协方差矩阵

3. 计算协方差矩阵 C 的特征值和特征向量

4. 选取大的特征值对应的特征向量，得到新的数据集。

3.4 Image transformation

通过对于图像的变换实现了对于数据集合的扩大，在位置的维度上丰富了训练数据。降低了 overfitting 和网络结构设计的复杂程度。

3.5 LPN

Local Response Normalization，局部响应归一化。完成“临近抑制”操作，对局部输入区域进行归一化。在通道间归一化模式中，局部区域范围在相邻通道间，没有空间扩展。在通道内归一化模式中，局部区域在空间上扩展，只针对独立通道进行。

3.6 Pre-training

“逐层初始化”（Layer-wise Pre-training）。采用的逐层贪婪训练方法来训练网络的参数，即先训练网络的第一个隐含层，解得到较好的值作为第一层的初始参数。然后接着训练第二层，第三层。最后用这些训练好的网络参数值作为整体网络参数的初始值。前几层的网络层次基本都用无监督的方法容易获得，只有最后一个输出层需要有监督的数据。

3.7 优化参数算法

梯度下降（Gradient Descent，GD）是最小化风险函数、损失函数的一种常用方法，随机梯度下降和批量梯度下降时两种迭代求解思路，二者在一定程度上都可以优化参数。

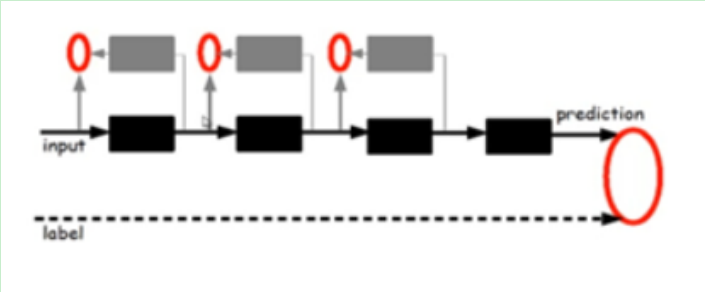


Figure 12: pre-training

- BSD(Batch Gradient Descent)：批量梯度下降。
  也叫最速梯度下降法，它的求解思路如下：

  1. 将 J(theta) 对 theta 求偏导，得到每个 theta 对应的的梯度：

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

  2. 由于是要最小化风险函数，所以按每个参数 theta 的梯度负方向，来更新每个 theta:

$$\theta_j = \theta_j + \frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

  3. 从上面公式可以注意到，它得到的是一个全局最优解，但是每迭代一步，都要用到训练集所有的数据，如果 m 很大，那么这种方法的迭代速度会非常慢。

- SGD(Stochastic Gradient Descent)：随机梯度下降。
  SGD 是 BSD 的变种，使用 SGD，将进行 N 次迭代，直到目标函数收敛，或者达某个既定的收敛界限。每次迭代都将对 m 个样本进行计算，计算量大，因此引入随机梯度下降法，他的求解思路如下：

  1. 上面的风险函数可以写成如下这种形式，损失函数对应的是训练集中每个样本的粒度，而上面批量梯度下降对应的是所有的训练样本：

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(y^i - h_\theta(x^i))^2 = \frac{1}{m}\sum_{i=1}^{m}\cos t(\theta,(x^i,y^i))$$

$$\cos t(\theta,(x^i,y^i)) = \frac{1}{2}(y^i - h_\theta(x^i))^2$$

2. 每个样本的损失函数，对 theta 求偏导得到对应梯度，来更新 theta:

$$\theta_j = \theta_j + (y^i - h_\theta(x^i))^2 x_j^i$$

3. 随机梯度下降是通过每个样本来迭代更新一次，如果样本量很大的情况（例如几十万），那么可能只用其中几万条或者几千条的样本，就已经将 theta 迭代到最优解了，对比上面的批量梯度下降，迭代一次需要用到十几万训练样本，一次迭代不可能最优，如果迭代 10 次的话就需要遍历训练样本 10 次。但是，SGD 伴随的一个问题是噪音较 BGD 要多，使得 SGD 并不是每次迭代都向着整体最优化方向。

对二者的总结:

- 批量梯度下降—最小化所有训练样本的损失函数，使得最终求解的是全局的最优解，即求解的参数是使得风险函数最小。
- 随机梯度下降—最小化每条样本的损失函数，虽然不是每次迭代得到的损失函数都向着全局最优方向，但是大的整体的方向是向全局最优解的，最终的结果往往是在全局最优解附近。

3.8 Batch-Normalization

批量归一化的算法是 Google 在论文《Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift》中提出来的，根据这个算法，如果我们对每一层的数据都进行处理，训练时间和 overfit 程度可以降低一些。文中使用了类似 z-score 的归一化方式：每一维度减去自身均值，再除以自身标准差，由于使用的是随机梯度下降法，这些均值和方差也只能在当前迭代的 batch 中计算，故作者给这个算法命名为 Batch Normalization。

Batch Normalization 的加速作用体现在两个方面：一是归一化了每层和每维度的 scale，所以可以整体使用一个较高的学习率，而不必像以前那样迁就小 scale 的维度；二是归一化后使得更多的权重分界面落在了数据中，降低了 overfit 的可能性，因此一些防止 overfit 但会降低速度的方法，例如 dropout 和权重衰减就可以不使用或者降低其权重。

3.9 PReLU

参数化修正线性单元 PReLU, Parametric Rectified Linear Unit)，是来自 MRSA 何恺明等研究员在论文《Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification》中提出的。神经网络已经出现几十年了，大家都默认使用 Sigmoid/Tanh 函数作为非线性单元，直到

近几年大家才意识到 ReLU 更好（更适合优化，精度也更好）。修正神经元 (rectifier neuron) 是近期将深度神经网络应用于计算机视觉挑战时取得成功的关键要素之一。这篇文章中提出了一种新的修正线性单元（ReLU, Rectified Linear Unit），并将其称为参数化修正线性单元（PReLU, Parametric Rectified Linear Unit）。

它的激活函数被定义为：

$$f(y_i) = max(0, y_i) + a_i min(0, y_i)$$

其中 a_i 是一个学习参数。下图是 ReLU 和 PReLU 的示意图:

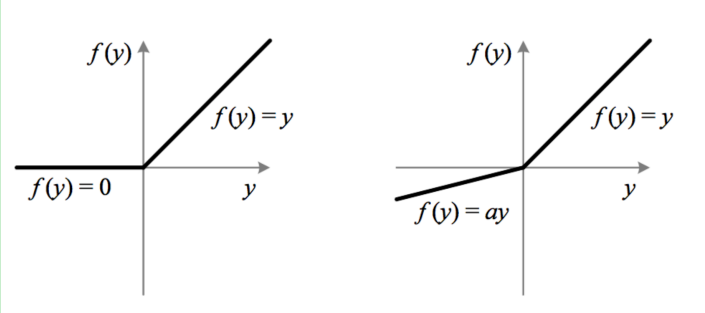


Figure 13: ReLU 和 PReLU 的示意图

该激活函数不仅可自适应获取修正参数，还可提高计算精度，且所需额外计算成本几乎可以忽略不计。

4. 解决方案

4.1 Caffe

Caffe（Convolutional Architecture for Fast Feature Embedding）是纯粹的 C++/CUDA 架构，支持命令行、Python 和 MATLAB 接口，且在 CPU 和 GPU 直接无缝切换。

4.1.1 Caffe 优点：

- 上手快：模型与相应优化都是以文本形式而非代码形式给出。
- 速度快：能够运行较好的模型与海量的数据。
- 模块化：方便扩展到新的任务和设置上。
- 开放性：公开的代码和参考模型用于再现。
- 社区好：可以通过 BSD-2 参与开发与讨论。

4.1.2 代码层次

Blob 基础的数据结构，保存学习到的参数以及网络传输过程中产生数据的类。

主要是对 protocol buffer 所定义的数据结构的继承，Caffe 也因此可以在尽可能小的内存占用下获得很高的效率。在更高一级的 Layer 中 Blob 用下面的形式表示学习到的参数：

- vector<sharedptr<Blob<Dtype> > > blob
- vector<Blob<Dtype>*> & bottom
- vector<Blob<Dtype>*> *top

Layer 网络的基本单元，由此派生出了各种层类。修改这部分的人主要是研究特征表达方向的。

用某种 Layer 来表示卷积操作，非线性变换，Pooling，权值连接等操作。具体分为 5 大类 Layer：

- **NeuronLayer** 定义于 neuron_layers.hpp 中，其派生类主要是元素级别的运算，运算均为同址计算。
- **LossLayer** 定义于 loss_layers.hpp 中，其派生类会产生 loss，只有这些层能够产生 loss。
- **数据层** 定义于 data_layer.hpp 中，作为网络的最底层，主要实现数据格式的转换。
- **特征表达层** 定义于 vision_layers.hpp，实现特征表达功能，例如卷积操作，Pooling 操作等。
- **网络连接层和激活函数（待定）** 定义于 common_layers.hpp，Caffe 提供了单个层与多个层的连接，并在这个头文件中声明。这里还包括了常用的全连接层 InnerProductLayer 类。

在 Layer 内部，数据主要有两种传递方式，**正向传导（Forward）**和**反向传导（Backward）**。Caffe 中所有的 Layer 都要用这两种方法传递数据。

Net 网络的搭建，将 Layer 所派生出层类组合成网络。

Net 用容器的形式将多个 Layer 有序地放在一起，其自身实现的功能主要是对逐层 Layer 进行初始化，以及提供 Update() 的接口（更新网络参数），本身不能对参数进行有效地学习。

- vector<shared_ptr<Layer<Dtype> > > layers _
- vector<Blob<Dtype>*> & Forward()
- void Net<Dtype>::Backward()

Solver Net 的求解，修改这部分人主要会是研究 DL 求解方向的。

这个类中包含一个 Net 的指针，主要是实现了训练模型参数所采用的优化算法，它所派生的类就可以对整个网络进行训练了。

- shared_ptr<Net<Dtype> > net_
- virtual void ComputeUpdateValue() = 0

4.2 CNN

Deep Learning 是全部深度学习算法的总称，CNN 是深度学习算法在图像处理领域的一个应用。

4.2.1 CNN 优点：

- 权值共享网络结构，降低网络模型的复杂度，减少了权值的数量。
- 图像直接作为网络的输入，避免复杂的特征提取和数据重建。
- 对平移、比例缩放、倾斜或者其他形式的变形具有高度不变性。

4.2.2 CNN 组成：

- 局部感知
- 参数共享
- 多卷积核
- Down-pooling
- 多层卷积

4.2.3 CNN 网络配置文件：

- Imagenet_solver.prototxt （包含全局参数的配置文件）
- Imagenet.prototxt （包含训练网络的配置文件）
- Imagenet_val.prototxt （包含测试网络的配置文件）

4.3 AlexNet

4.3.1 网络结构

AlexNet 网络结构是 CNN 类型的 DeepLearning 网络模型在图像分类上的应用，如图 Figure 14。

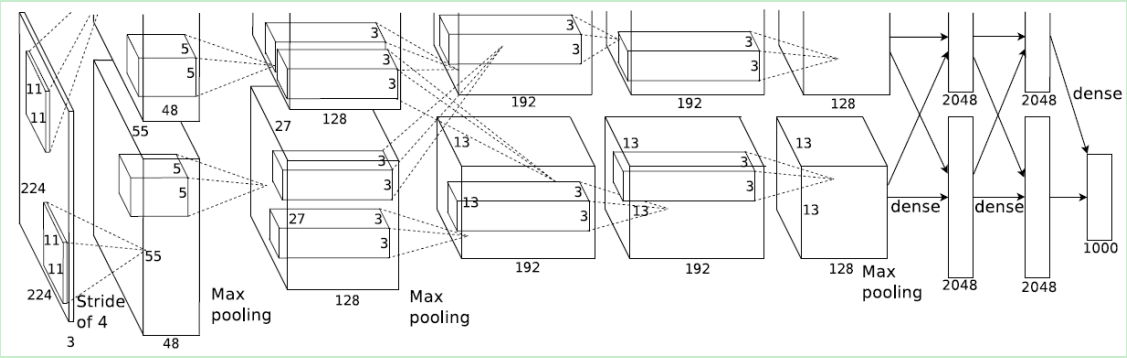


Figure 14: AlexNet

AlexNet 网络模型训练了一个深度卷积神经网络，来将 ILSVRC 2010 中 1.2M 的高分辨率图像数据分为 1000 类。测试结果，Top-1 和 Top-5 的错误率分别为 37.5% 和 17%，优于当时最优的水平。后来作者修改了该模型参与了 ILSVRC 2012 竞赛，以 Top-5 错误率 15.3% 遥遥领先亚军的 26.2%。

AlexNet 网络模型用 5 个卷积层和 3 个全连接层，其中包含了 60M 参数和 650K 神经元。为使训练更快，AlexNet 采用非饱和神经元，包括了大量不常见和新的特征来提升性能，减少训练时间。并利用了两个高效的 GPU 进行卷积运算。而这个层深度很重要，因为移除任何一个卷积层，将会导致更差的结果。网络大小主要受限于 GPU 的内存和训练时间。实验证明，本网络在有两个 GTX 580 3GB GPU 的机器上训练了 5~6 天。实验结果显示，如果 GPU 更快或数据集更大，实验结果将会更好。

为了简化实验，模型没有使用任何无监督学习。否则，在更大规模的网络和更长时间训练的情况下，AlexNet 所得结果就能得到提高。另外，保持 AlexNet 网络模型的结构稳定，因为每移走一层中间的层，Top-1 的错误率将会增高 2%。

4.3.2 逐层功能实现：

- 输入 224×224 的图片，3 通道
- 第一层卷积 + pooling：11×11 的卷积核 96 个，步长为 4，每个 GPU 各有 48 个。max-pooling 的核为 2×2。如图 Figure 15。
- 第二层卷积 + pooling：5×5 的卷积核 256 个，每个 GPU 各有 128 个。max-pooling 的核为 2×2。如图 Figure 16。

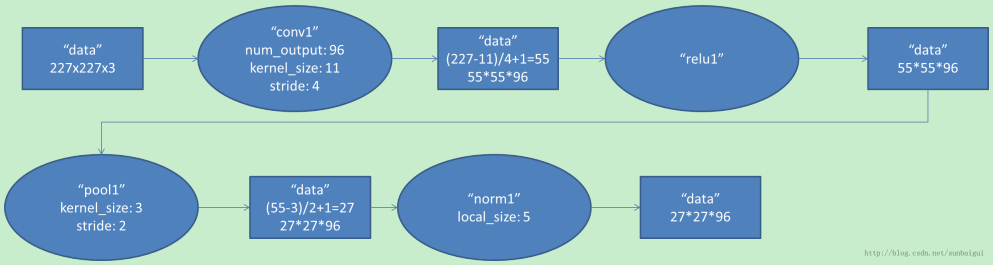


Figure 15: Conv1

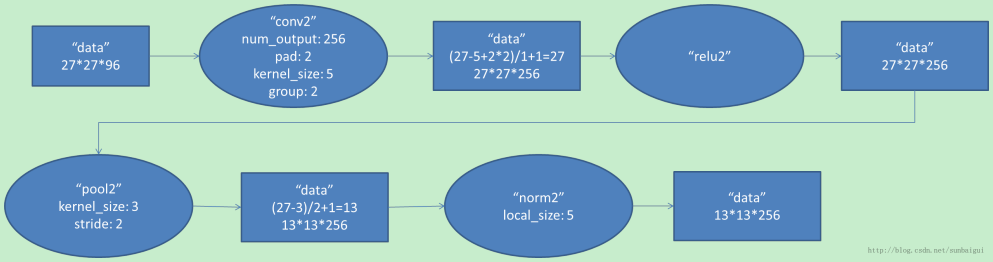


Figure 16: Conv2

- 第三层卷积：与上一层是全连接。3×3 的卷积核 384 个，每个 GPU 各有 192 个。如图 Figure 17。

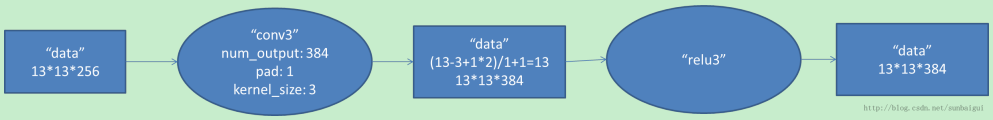


Figure 17: Conv3

- 第四层卷积：3×3 的卷积核 384 个，每个 GPU 各有 192 个。如图 Figure 18。

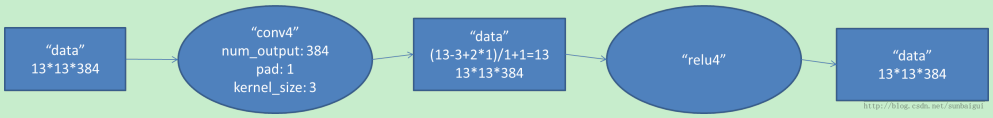


Figure 18: Conv4

- 第五层卷积 + pooling：3×3 的卷积核 256 个，每个 GPU 各有 128 个。max-pooling：2 ×2 的核。如图 Figure 19。

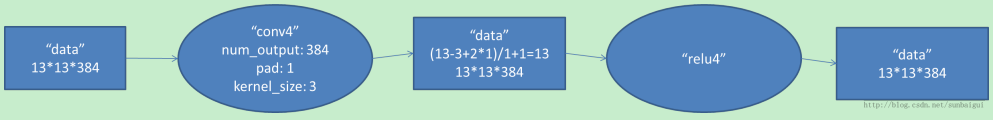


Figure 19: Conv5

- 第六层全连接：4096 维，将第五层的输出连接成为一个一维向量，作为该层的输入。如图 Figure 20。

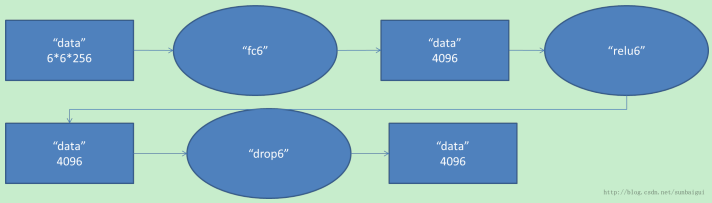


Figure 20: Fc6

- 第七层全连接：4096 维，将第六层的输出连接成为一个一维向量。如图 Figure 21。

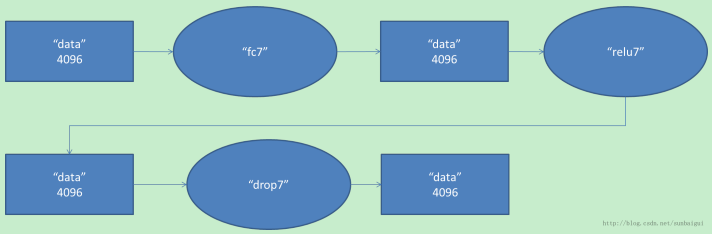


Figure 21: Fc7

- Softmax 层：输出为 1000，输出的每一维都是图片属于该类别的概率。如图 Figure 22。

Figure 22: Fc8

4.3.3 注意事项

- 降低过拟合:
  - 数据增强：通过提取图片的 5 个 224*224 切片（中间和四角）和它们的水平翻转来做出预测，预测结果为十次预测的平均值。第二种数据增强的方式为改变训练图像 RGB 通道的强度，对 RGB 空间做 PCA，然后对主成分做高斯扰动。结果让错误率又下降了 1%。
  - Dropout：将某些层隐藏，按照 50% 的概率输出 0。这些隐藏的神经元不会参加 CNN 的 forward 过程，也不会参加 back propagation 过程。测试中，在前两个全连接层使用了该方法，使所有的神经元，但其输出都乘以 0.5。如果没有 dropout 策略，AlexNet 将会产生严

重的过拟合。

- 架构上可改进方面：
  - ReLU 非线性特征
  - 在多 GPU 上训练
  - 局部响应标准化
  - 重叠采样

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. 12 2013.

[3] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[4] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep fisher networks for large-scale image classification. In *Advances in neural information processing systems*, pages 163–171, 2013.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.