浮游动物识别分类——深度学习

Dai Jialun Wu Bin

June 27, 2015

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。

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。

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

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。

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。

2.2 CNN 组成：

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

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）

3. AlexNet

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 1。

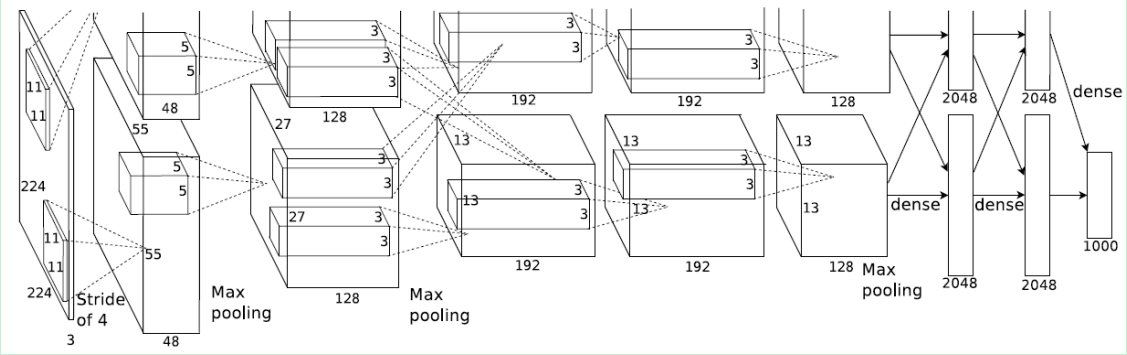

Figure 1: AlexNet

- 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 2。

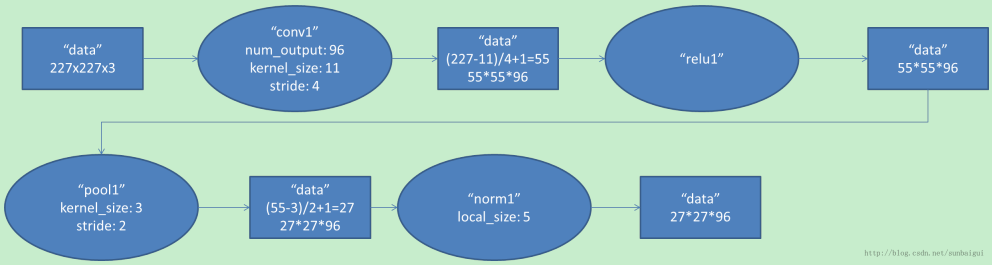

Figure 2: Conv1

- 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 3。

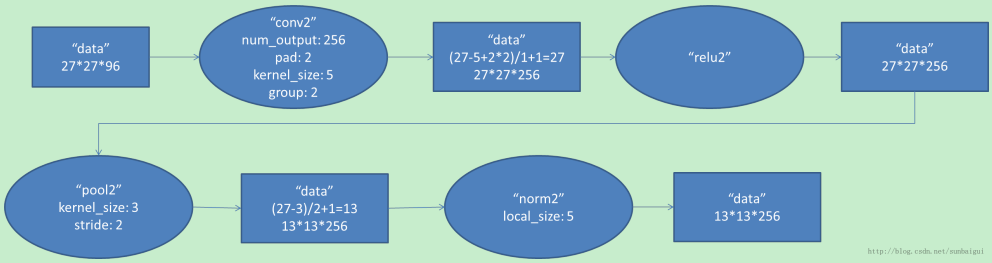

Figure 3: Conv2

- 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 4。

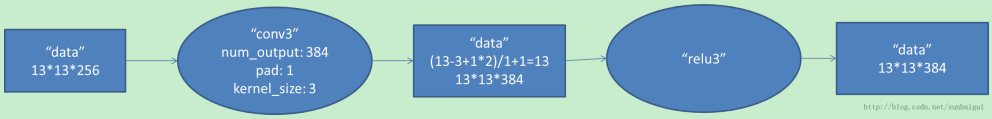

Figure 4: Conv3

- 第四层卷积：3×3 的卷积核 384 个，每个 GPU 各有 192 个。如图 Figure 5。

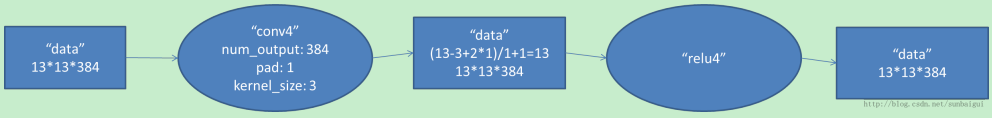

Figure 5: Conv4

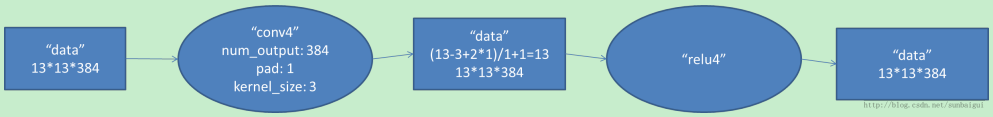

Figure 6: Conv5

- 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 6。

- 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 7。

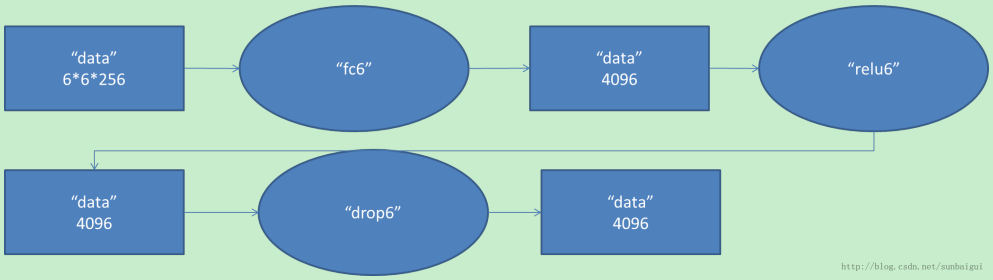

Figure 7: Fc6

- 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 8。

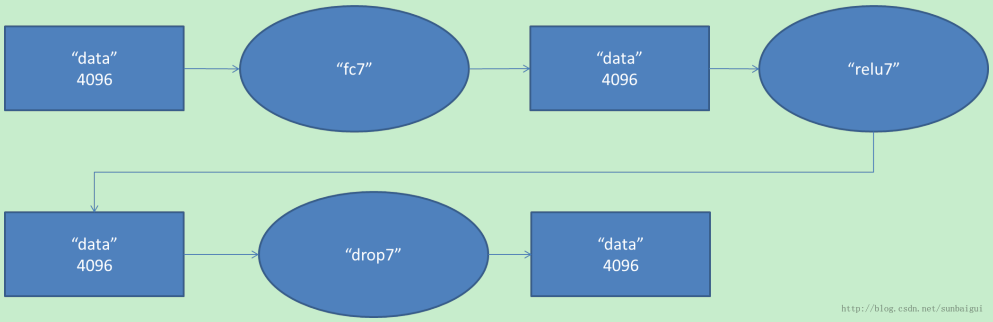

Figure 8: Fc7

- 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 9。

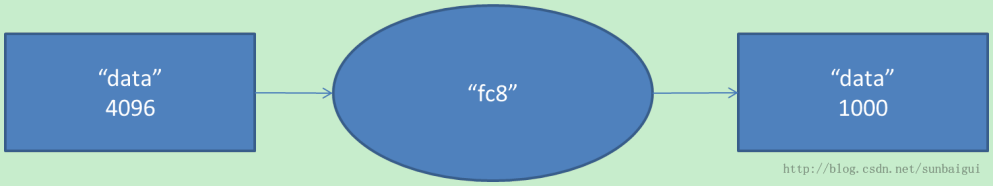

Figure 9: Fc8

3.2 在数据与模型上的技巧

- 数据
  - 增大训练样本
  - 用 PCA 增强训练数据
- 模型结构
  - Local Response Normalization
  - Dropout 策略
- 优化算法的参数
  - SGD（随机梯度下降法）