**[深度学习与计算机视觉(11)\_基于deep learning的快速图像检索系统](https://blog.csdn.net/yaoqiang2011/article/details/50856583)**

作者：[寒小阳](http://blog.csdn.net/han_xiaoyang?viewmode=contents" \t "_blank)   
时间：2016年3月。   
出处：<http://blog.csdn.net/han_xiaoyang/article/details/50856583>   
声明：版权所有，转载请联系作者并注明出处

## 1.引言

本系统是基于CVPR2015的论文[《Deep Learning of Binary Hash Codes for Fast Image Retrieval》](http://www.iis.sinica.edu.tw/~kevinlin311.tw/cvprw15.pdf" \t "_blank)实现的海量数据下的基于内容图片检索系统，250w图片下，对于给定图片，检索top 1000相似时间约为1s，其基本背景和原理会在下文提到。

## 2.基本问题与技术

大家都知道，基于内容的图像检索系统是根据图像的内容，在已有图像集中找到最『相近』的图片。而这类系统的效果(精准度和速度)和两个东西直接相关：

* **图片特征的表达能力**
* **近似最近邻的查找**

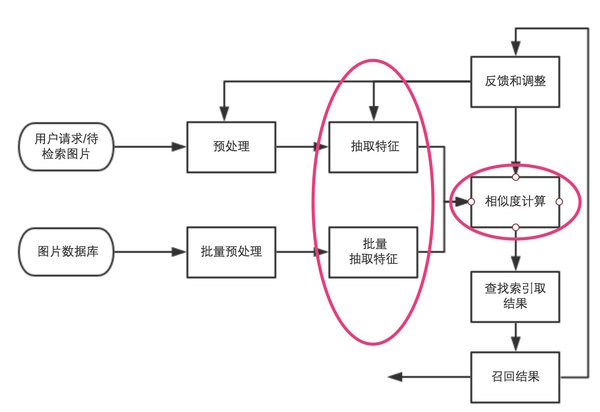
根据我们这个简单系统里的情况粗浅地谈谈这两个点。

首先说图像特征的表达能力，这一直是基于内容的图像检索最核心却又困难的点之一，计算机所『看到』的图片**像素层面表达的低层次信息**与人所理解的图像**多维度高层次信息内容**之间有很大的差距，因此我们需要一个尽可能丰富地表达图像层次信息的特征。我们前面的博客也提到了，deep learning是一个对于图像这种层次信息非常丰富的数据，有更好表达能力的框架，其中每一层的中间数据都能表达图像某些维度的信息，相对于传统的Hist，Sift和Gist，表达的信息可能会丰富一下，因此这里我们用deep learning产出的特征来替代传统图像特征，希望能对图像有更精准的描绘程度。

再说『近似最近邻』，ANN(Approximate Nearest Neighbor)/近似最近邻一直是一个很热的研究领域。因为在海量样本的情况下，遍历所有样本，计算距离，精确地找出最接近的Top K个样本是一个非常耗时的过程，尤其有时候**样本向量的维度也相当高**，因此有时候我们会牺牲掉一小部分精度，来完成在很短的时间内找到近似的top K个最近邻，也就是ANN，最常见的ANN算法包括[局部敏感度哈希/locality-sensitive hashing](https://en.wikipedia.org/wiki/Locality-sensitive_hashing" \t "_blank)，[最优节点优先/best bin first](https://en.wikipedia.org/wiki/Best_bin_first" \t "_blank)和[Balanced box-decomposition tree](https://en.wikipedia.org/wiki/Balanced_box-decomposition_tree" \t "_blank)等，我们系统中将采用LSH/局部敏感度哈希来完成这个过程。有一些非常专业的ANN库，比如[FLANN](http://www.cs.ubc.ca/research/flann/" \t "_blank)，有兴趣的同学可以了解一下。

## 3. 本检索系统原理

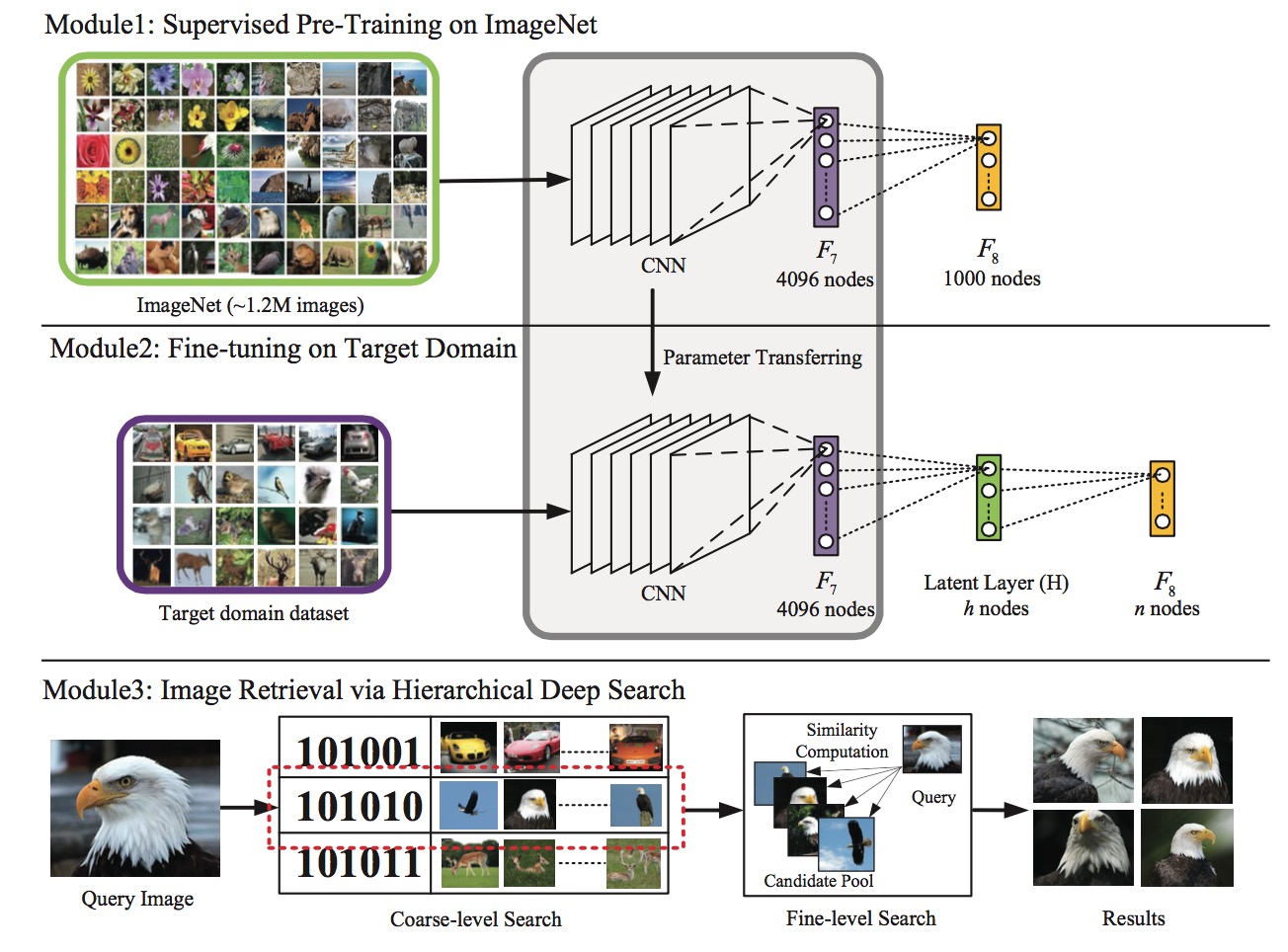
图像检索系统和关键环节如下图所示：



图像检索过程简单说来就是对图片数据库的每张图片抽取特征(一般形式为特征向量)，存储于数据库中，对于待检索图片，抽取同样的特征向量，然后并对该向量和数据库中向量的距离，找出最接近的一些特征向量，其对应的图片即为检索结果。

基于内容的图像检索系统最大的难点在上节已经说过了，其一为大部分神经网络产出的中间层特征维度非常高，比如Krizhevsky等的在2012的ImageNet比赛中用到的[AlexNet神经网](http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks" \t "_blank)，第7层的输出包含丰富的图像信息，但是维度高达4096维。4096维的浮点数向量与4096维的浮点数向量之间求相似度，运算量较大，因此Babenko等人在论文[Neural codes for image retrieval](http://arxiv.org/pdf/1404.1777v2.pdf" \t "_blank)中提出用PCA对4096维的特征进行PCA降维压缩，然后用于基于内容的图像检索，此场景中效果优于大部分传统图像特征。同时因为高维度的特征之间相似度运算会消耗一定的时间，因此线性地逐个比对数据库中特征向量是显然不可取的。大部分的ANN技术都是将高维特征向量压缩到低维度空间，并且以01二值的方式表达，因为在低维空间中计算两个二值向量的汉明距离速度非常快，因此可以在一定程度上缓解时效问题。ANN的这部分hash映射是在拿到特征之外做的，本系统框架试图让卷积神经网在训练过程中学习出对应的『二值检索向量』，或者我们可以理解成对全部图先做了一个分桶操作，每次检索的时候只取本桶和临近桶的图片作比对，而不是在全域做比对，以提高检索速度。

[论文](http://www.iis.sinica.edu.tw/~kevinlin311.tw/cvprw15.pdf)是这样实现『二值检索向量』的：在Krizhevsky等2012年用于ImageNet中的卷积神经网络结构基础上，在第7层(4096个神经元)和output层之间多加了一个隐层(全连接层)。隐层的神经元激励函数，可以选用sigmoid，这样输出值在0-1之间值，可以设定阈值(比如说0.5)之后，将这一层输出变换为01二值向量作为『二值检索向量』，这样在使用卷积神经网做图像分类训练的过程中，会『学到』和结果类别最接近的01二值串，也可以理解成，我们把第7层4096维的输出特征向量，通过神经元关联压缩成一个低维度的01向量，但不同于其他的降维和二值操作，这是在一个神经网络里完成的，每对图片做一次完整的前向运算拿到类别，就产出了表征图像丰富信息的第7层output(4096维)和代表图片分桶的第8层output(神经元个数自己指定，一般都不会很多，因此维度不会很高)。引用论文中的图例解释就是如下的结构：



上方图为ImageNet比赛中使用的卷积神经网络；中间图为调整后，在第7层和output层之间添加隐层(假设为128个神经元)后的卷积神经网络，我们将复用ImageNet中得到最终模型的前7层权重做fine-tuning，得到第7层、8层和output层之间的权重。下方图为实际检索过程，对于所有的图片做卷积神经网络前向运算得到第7层4096维特征向量和第8层128维输出(设定阈值0.5之后可以转成01二值检索向量)，对于待检索的图片，同样得到4096维特征向量和128维01二值检索向量，在数据库中查找二值检索向量对应『桶』内图片，比对4096维特征向量之间距离，做重拍即得到最终结果。图上的检索例子比较直观，对于待检索的”鹰”图像，算得二值检索向量为101010，取出桶内图片(可以看到基本也都为鹰)，比对4096维特征向量之间距离，重新排序拿得到最后的检索结果。

## 4. 预训练好的模型

一般说来，在自己的图片训练集上，针对特定的场景进行图像类别训练，得到的神经网络，中间层特征的表达能力会更有针对性一些。具体训练的过程可以第3节中的说明。对于不想自己重新费时训练，或者想快速搭建一个基于内容的图片检索系统的同学，这里也提供了100w图片上训练得到的卷积神经网络模型供大家使用。

这里提供了2个预先训练好的模型，供大家提取『图像特征』和『二值检索串』用。2个模型训练的数据集一致，卷积神经网络搭建略有不同。对于几万到十几万级别的小量级图片建立检索系统，请使用模型Image\_Retrieval\_20\_hash\_code.caffemodel，对于百万以上的图片建立检索系统，请使用模型Image\_Retrieval\_128\_hash\_code.caffemodel。

对于同一张图片，两者产出的特征均为4096维度，但用作分桶的『二值检索向量』长度，前者为20，后者为128。

模型下载地址为[云盘地址](http://pan.baidu.com/s/1eQS8l6y)。

# 傻瓜式环境配置手册

## 1.关于系统

这个说明是关于linux系统的，最好是centOS 7.0以上，或者ubuntu 14.04 以上。低版本的系统可能会出现boost，opencv等库版本不兼容问题。

## 2. centOS配置方法

### 2.1 配置yum源

配置合适的yum源是一种『偷懒』的办法，可以简化很多后续操作。不进行这一步的话很多依赖库都需要自己手动编译和指定caffe编译路径，耗时且经常编译不成功。   
在国内的话用sohu或者163的源   
rpm -Uvh http://mirrors.sohu.com/fedora-epel/7/x86\_64/e/epel-release-7-2.noarch.rpm

如果身处国外的话，可以查一下[fedora mirror list](https://admin.fedoraproject.org/mirrormanager/mirrors/EPEL/7/x86_64" \t "_blank)，找到合适的yum源添加。

接着我们让新的源生效：   
yum repolist

### 2.2 安装依赖的库

该图像检索系统依赖于caffe深度学习框架，因此需要安装caffe依赖的部分库：比如protobuf是caffe中定义layers的配置文件解析时需要的，leveldb是训练时存储图片数据的数据库，opencv是图像处理库，boost是通用C++库，等等…

我们用yum install一键安装：   
sudo yum install protobuf-devel leveldb-devel snappy-devel opencv-devel boost-devel hdf5-devel

### 2.3 安装科学计算库

这个部分大家都懂的，因为要训练和识别过程，涉及到大量的科学计算，因此必要的科学计算库也需要安装。同时python版本caffe中会依赖一些python科学计算库，pip和easy\_install有时候安装起来会有一些问题，因此部分库这里也用yum install直接安装了。   
yum install openblas-devel.x86\_64 gcc-c++.x86\_64 numpy.x86\_64 scipy.x86\_64 python-matplotlib.x86\_64 lapack-devel.x86\_64 python-pillow.x86\_64 libjpeg-turbo-devel.x86\_64 freetype-devel.x86\_64 libpng-devel.x86\_64 openblas-devel.x86\_64

### 2.4 其余依赖

包括lmdb等：   
sudo yum install gflags-devel glog-devel lmdb-devel   
若此处yum源中找不到这些拓展package，可是手动编译(要有root权限)：

# glog

wget https://google-glog.googlecode.com/files/glog-0.3.3.tar.gz

tar zxvf glog-0.3.3.tar.gz

cd glog-0.3.3

./configure

make && make install

# gflags

wget https://github.com/schuhschuh/gflags/archive/master.zip

unzip master.zip

cd gflags-master

mkdir build && cd build

export CXXFLAGS="-fPIC" && cmake .. && make VERBOSE=1

make && make install

# lmdb

git clone https://github.com/LMDB/lmdb

cd lmdb/libraries/liblmdb

make && make install

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17

### 2.5 python版本依赖

编译pycaffe的时候，我们需要更多的一些python的依赖库。这时候我们可以用pip或者easy\_install完成。

pip和easy\_install的配置方法为：

wget --no-check-certificate https://bootstrap.pypa.io/ez\_setup.py

python ez\_setup.py --insecure

wget https://bootstrap.pypa.io/get-pip.py

python get-pip.py

* 1
* 2
* 3
* 4

在caffe/python/requirements.txt中有pycaffe的python依赖包，如下：

Cython>=0.19.2

numpy>=1.7.1

scipy>=0.13.2

scikit-image>=0.9.3

matplotlib>=1.3.1

ipython>=3.0.0

h5py>=2.2.0

leveldb>=0.191

networkx>=1.8.1

nose>=1.3.0

pandas>=0.12.0

python-dateutil>=1.4,<2

protobuf>=2.5.0

python-gflags>=2.0

pyyaml>=3.10

Pillow>=2.3.0

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16

通过以下shell命令可以全部安装：

for req in $(cat requirements.txt); do pip install $req; done

* 1

## 3. ubuntu配置方法

基本与centOS一致，这里简单列出需要执行的shell命令：

sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler

sudo apt-get install --no-install-recommends libboost-all-dev

sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev

* 1
* 2
* 3
* 4

python部分的依赖包安装方式同上。

## 4. caffe的编译与准备

保证caffe所需依赖都安装完成后，在caffe目录下执行：   
cp Makefile.config.example Makefile.config   
根据自己的实际情况，修改Makefile.config的内容，主要修改的几个如下：

* 如果**没有GPU**，只打算用CPU进行实验，将# CPU\_ONLY := 1前的#号去掉。
* 如果使用GPU，且有**cuDNN加速**，将# USE\_CUDNN := 1前的#号去掉。
* 如果使用openBLAS，将BLAS := atlas改成BLAS := open，并添加BLAS\_INCLUDE := /usr/include/openblas(Caffe中默认的矩阵运算库为[ATLAS](http://math-atlas.sourceforge.net/" \t "_blank)，但是[OpenBLAS](http://www.openblas.net/" \t "_blank)有一些性能优化，因此建议换做OpenBLAS)

未完待续…