学习笔记 3　用自己的数据训练和测试“CaffeNet”

2014.7.22 薛开宇

本次学习笔记作用比较大，也是重点，知道如何在 caffe 上搭建自己的数据库。

3.1 数据准备

本学习笔记有点脱离了原文，原文是用 ImageNet1000 类的数据库，而因为电脑内存不足，只能自己模仿做一个小的数据库进行下去。

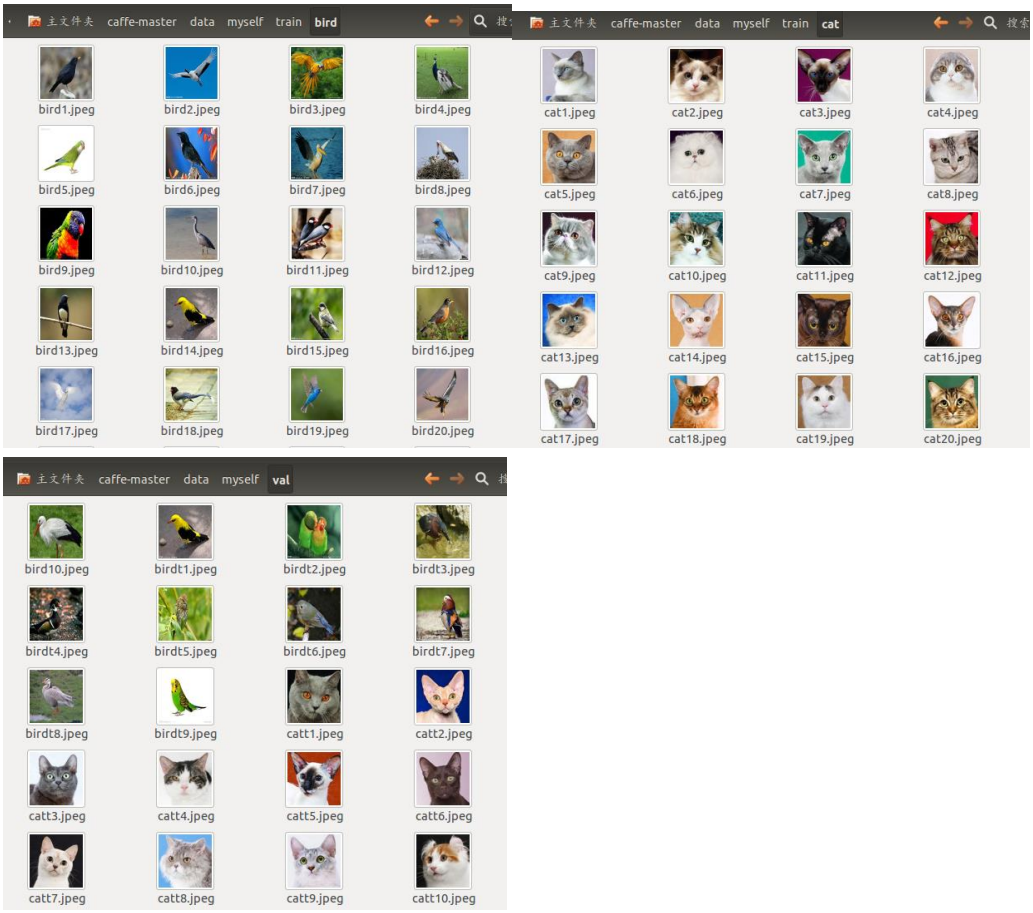
本来教程是假设已经下载了 ImageNet 训练数据和验证数据(非常大)，并以下面的格式存储在磁盘：

/path/to/imagenet/train/n01440764/n01440764_10026.JPEG
/path/to/imagenet/val/ILSVRC2012_val_00000001.JPEG

里面是各种的分类图。

因为实在太大，所以我们改为模仿搭建自己的数据库。

在 data 中新建文件夹 myself，本人在网上下载了训练猫的图片 50 张，测试猫 10 张，训练鸟的图片 50 张，测试鸟 10 张。如图所示：



如果坚持用 Imagenet 的话，我们还需要一些标签数据进行训练，用以下指令可以下载，如果不用，就可以不执行下面指令。

cd $CAFFE_ROOT/data/ilsvrc12/
./get_ilsvrc_aux.sh

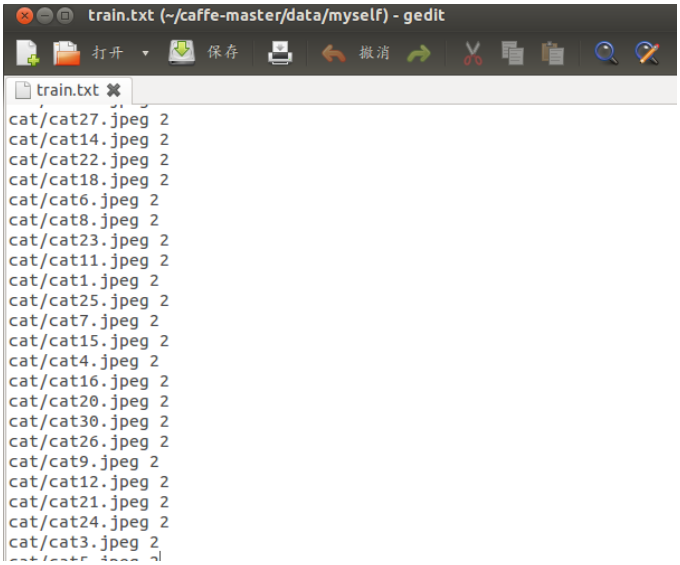
培训和测试的输入是用 train.txt 和 val.txt 描述的，这些文档列出所有文件和他们的标签。注意，我们分类的名字是 ASCII 码的顺序，即 0-999，对应的分类名和数字的映射在 synset_words.txt（自己写）中。

运行以下指令：

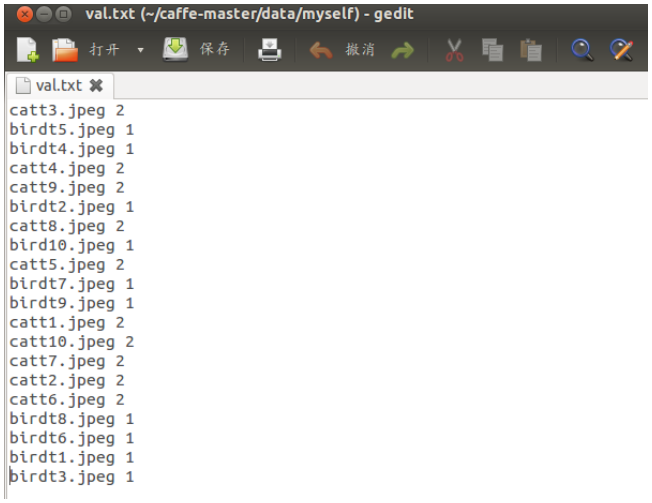
find -name *.jpeg |cut -d '/' -f2-3 > train.txt

注意路径

然后，因为样本数比较少，可以自行手动做分类标签。在 train.txt 的每个照片后用 0-999 分类。当样本过多，就自己编写指令批量处理。



同理，获得 val.txt



Test.txt 和 val.txt 一样，不过后面不能标签，全部设置成 0。

我们还需要把图片的大小变成 256X256，但在一个集群环境，可以不明确的进行，使用 Mapreduce 就可以了，像杨青就是这样做的。如果我们希望更简单，用下面的命令：

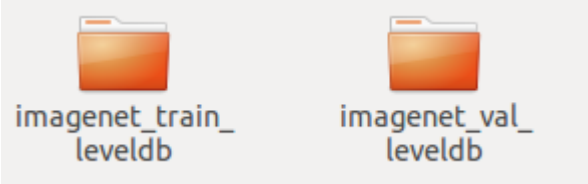
```
for name in /path/to/imagenet/val/*.JPEG; do
    convert -resize 256x256\! $name $name
done
```

我做成了 sh，方便以后使用。

然后在 caffe-master 创建 myself 文件夹，然后将 imagenet 的 create_imagenet.sh. 复制到该文件夹下进行修改，进行训练和测试路径的设置，运行该 sh.

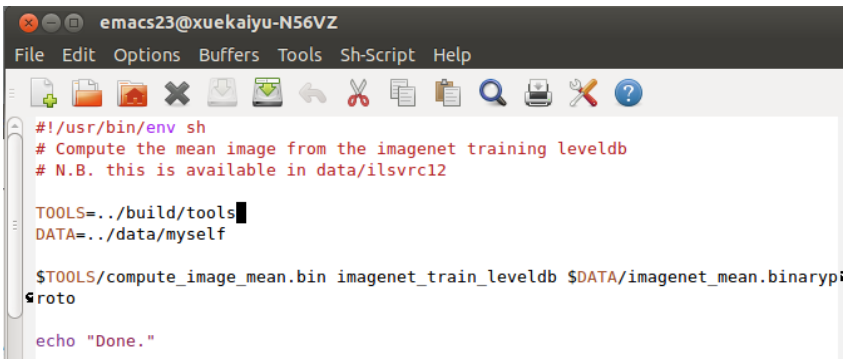
```
DATA=../../data/myself

echo "Creating leveldb..."

GLOG_logtostderr=1 $TOOLS/convert_imageset.bin \
    /home/xuekaiyu/caffe-master/data/myself/train/ \
    $DATA/train.txt \
    imagenet_train_leveldb 1

GLOG_logtostderr=1 $TOOLS/convert_imageset.bin \
    /home/xuekaiyu/caffe-master/data/myself/val/ \
    $DATA/val.txt \
    imagenet_val_leveldb 1
```

最后得到



3.2 计算图像均值

模型需要我们从每张图片减去均值，所以我们必须获得训练的均值，用 tools/compute_image_mean.cpp 实现，这个 cpp 是一个很好的例子去熟悉如何操作多个组建，例如协议的缓冲区，leveldbs，登录等。我们可以直接复制 imagenet 的 ./make_imagenet_mean.加以修改应用就行,注意路径。



3.3 网络的定义

从 imagenet 中复制修改 imagenet_train.prototxt，注意修改数据层的路径。

source: "ilvsrc12_train_leveldb"
mean_file: "../../data/ilsvrc12/imagenet_mean.binaryproto"

我改成

```
data_param {
  source: "imagenet_train_leveldb"
  mean_file: "../data/myself/imagenet_mean.binaryproto"
  batch_size: 256
  crop_size: 227
  mirror: true
}
```

同理，复制修改 imagenet_val.prototxt.
改成

```
data_param {
  source: "imagenet_val_leveldb"
  mean_file: "../data/myself/imagenet_mean.binaryproto"
  batch_size: 50
  crop_size: 227
  mirror: false
}
```

如果你细心观察 imagenet_train.prototxt and imagenet_val.prototxt，你会发现他们除了数据来源不同和最后一层不同，其他基本相同。在训练中，我们用一个 softmax——loss 层计算损失函数和初始化反向传播，而在验证，我们使用精度层检测我们的精度。

我们还有一个运行的协议 imagenet_train.prototxt，复制过来，从里面可以观察到，我们将运行 256 批次，迭代 4500000 次（90 期），每 1000 次迭代，我们测试学习网络验证数据，我们设置初始的学习率为 0.01，每 100000（20 期）次迭代减少学习率，显示一次信息，训练的 weight_decay 为 0.0005，每 10000 次迭代，我们显示一下当前状态。

以上是教程的，实际上，以上需要耗费很长时间，因此，我们稍微改一下

test_iter: 1000 是指测试的批次，我们就 10 张照片，设置 10 就可以了。

test_interval: 1000 是指每 1000 次迭代测试一次，我们改成 500 次测试一次。

base_lr: 0.01 是基础学习率，因为数据量小，0.01 就会下降太快了，因此改成 0.001

lr_policy: "step"学习率变化

gamma: 0.1 学习率变化的比率

stepsize: 100000 每 100000 次迭代减少学习率

display: 20 每 20 层显示一次

max_iter: 450000 最大迭代次数，

momentum: 0.9 学习的参数，不用变

weight_decay: 0.0005 学习的参数，不用变

snapshot: 10000 每迭代 10000 次显示状态，这里改为 2000 次

solver_mode: GPU 末尾加一行，代表用 GPU 进行

3.4 训练

把./train_imagenet.sh 复制过来运行，然后就像学习笔记 1 一样训练了，当然，只有两类，正确率还是相当的高。

```
xuekaiyu@xuekaiyu-N56VZ: ~/caffe-master/myself
I0716 20:36:26.403406 22176 solver.cpp:142] Test score #1: 0.295262
I0716 20:38:35.474187 22176 solver.cpp:237] Iteration 100, lr = 0.001
I0716 20:38:35.501572 22176 solver.cpp:87] Iteration 100, loss = 0.116571
I0716 20:38:35.501607 22176 solver.cpp:106] Iteration 100, Testing net
I0716 20:38:39.976903 22176 solver.cpp:142] Test score #0: 0.9
I0716 20:38:39.976945 22176 solver.cpp:142] Test score #1: 0.276086
I0716 20:40:49.117224 22176 solver.cpp:106] Iteration 120, Testing net
I0716 20:40:53.598922 22176 solver.cpp:142] Test score #0: 0.85
I0716 20:40:53.598964 22176 solver.cpp:142] Test score #1: 0.225632
I0716 20:43:02.717257 22176 solver.cpp:106] Iteration 140, Testing net
I0716 20:43:07.201568 22176 solver.cpp:142] Test score #0: 0.85
I0716 20:43:07.201611 22176 solver.cpp:142] Test score #1: 0.25258
I0716 20:45:16.278803 22176 solver.cpp:106] Iteration 160, Testing net
I0716 20:45:20.745877 22176 solver.cpp:142] Test score #0: 0.9
I0716 20:45:20.745918 22176 solver.cpp:142] Test score #1: 0.506309
I0716 20:47:29.819383 22176 solver.cpp:106] Iteration 180, Testing net
I0716 20:47:34.291712 22176 solver.cpp:142] Test score #0: 0.9
I0716 20:47:34.291764 22176 solver.cpp:142] Test score #1: 0.465867
I0716 20:49:43.363847 22176 solver.cpp:237] Iteration 200, lr = 0.001
I0716 20:49:43.390959 22176 solver.cpp:87] Iteration 200, loss = 0.0109258
I0716 20:49:43.391000 22176 solver.cpp:106] Iteration 200, Testing net
I0716 20:49:47.858572 22176 solver.cpp:142] Test score #0: 0.9
I0716 20:49:47.858611 22176 solver.cpp:142] Test score #1: 0.291695
```

3.5 恢复数据

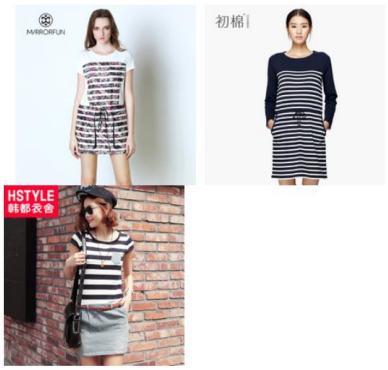
我们用指令./resume_training.sh 即可。

3.6 网上一些有趣的做法：

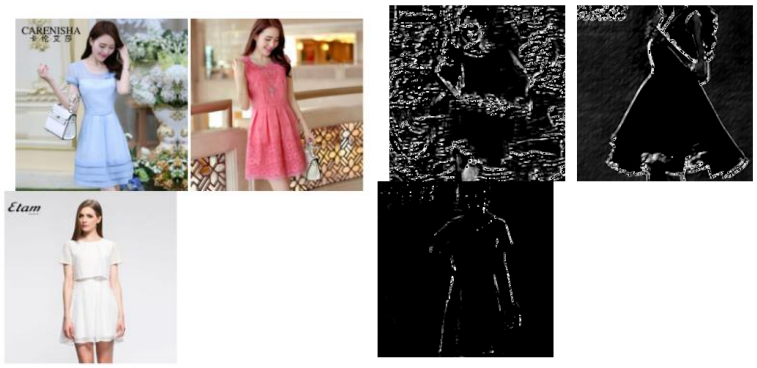
http://www.tuicool.com/articles/MZN3IvU

用 caffe 自己做衣服识别，两类，条纹衣服和纯色衣服。

条纹衣服的原图和特征图



纯色衣服和特征图：



看条纹衣服的特征图比较有意思，把“条纹”特征给抽取出来了。也许这就是神经网络神奇的地方，在没有人的干扰的情况下，竟然能学习出来“条纹”特征。

3.7 该文章总结的做样本库的注意事项：

1 数据集要保证质量。曾经玩过一字领和 polo 领的分类，刚开始效果很差，后来发现有一些“错误”的标签，于是把那些样本给去掉。效果好了很多。

2 learning rate 要调整。有一次训练了很久，准确率几乎不变，于是我减少了 lr，发现好了很多。这点深有同感，可以尝试把学习率调回 0.01，保证小数据不能超过 50 的识别。

3 均值化图片。实践证明，均值化后再训练收敛速度更快，准确率更高。

3.8 我们能做什么？

1 对于哪些数据集，深度学习比较适合？

2 对于效果差的数据集，如何能提高准确率？

主要资料来源：caffe 官网的教程

- ImageNet tutorial

  Train and test "CaffeNet" on ImageNet challenge data.