深度学习框架-Caffe

罗子幪

2018.11.16

内容

Caffe 数据结构

- blob
- layer
- net

训练模型

- 准备数据
- 网络结构配置
- 训练参数配置

Caffe?

Caffe是一个**结构清晰，高效，模块化**的深度学习框架

官方文档：http://caffe.berkeleyvision.org/tutorial/

源码： https://github.com/BVLC/caffe

模型库： https://github.com/BVLC/caffe/wiki/Model-Zoo

主流模型: https://github.com/soeaver/caffe-model

模型可视化: https://cwlacewe.github.io/netscope/#/editor

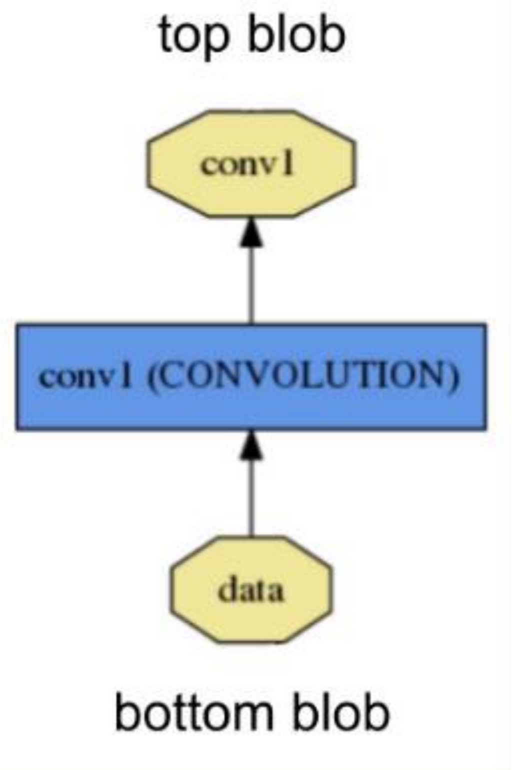
Caffe-Blob

Caffe使用blob来对数据进行**存储和交换**，blob提供了一个数据的统一接口。Caffe中图像数据，模型参数，反传的梯度等数据都是以blob的形式存储的。

Blob可以理解为一个4维的数组（num*channel*height*weight）

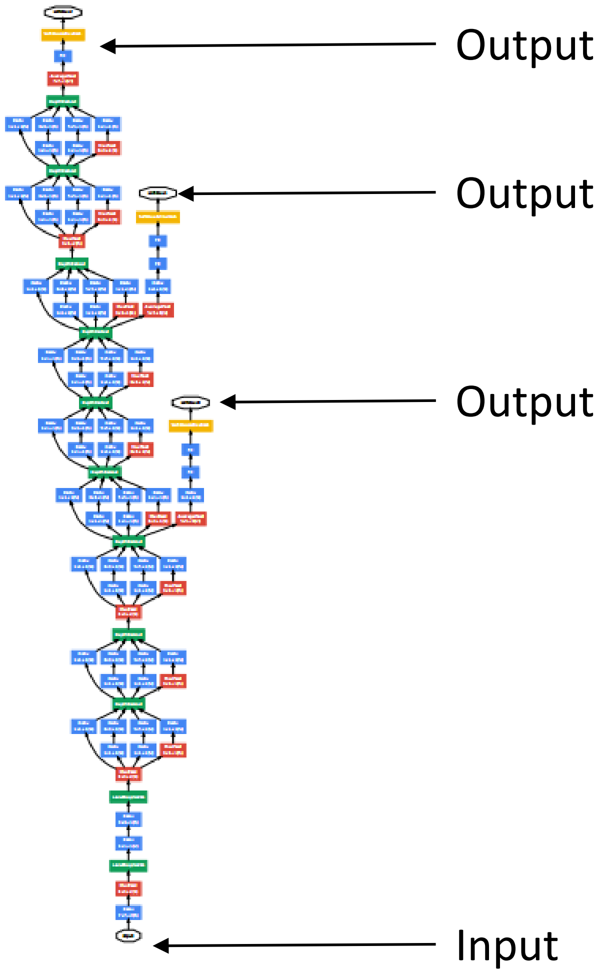
Caffe-Layer

层是一个模型的核心，它是一个基本的计算单元



Caffe-Net

网络是由一系列层连接起来构成的，是一个有向无环图(DGA)

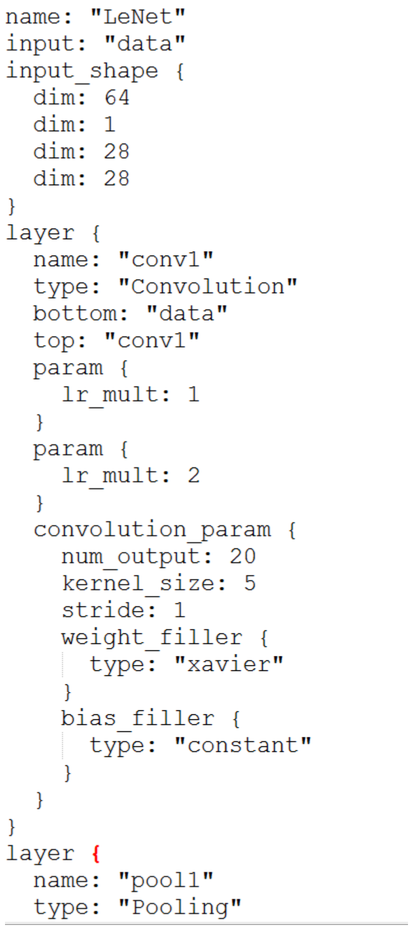


Output

Output

Output

Input



```
name: "LeNet"
input: "data"
input_shape {
  dim: 64
  dim: 1
  dim: 28
  dim: 28
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "pool1"
  type: "Pooling"
```

训练模型-准备数据

数据格式

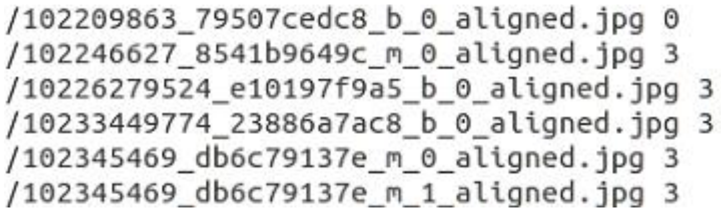
Images

Aaron_Eckhart_0001.bmp

Aaron_Guiel_0001.bmp

Aaron_Patterson_0001.bmp

file_list.txt



```
/102209863_79507cedc8_b_0_aligned.jpg 0
/102246627_8541b9649c_m_0_aligned.jpg 3
/10226279524_e10197f9a5_b_0_aligned.jpg 3
/10233449774_23886a7ac8_b_0_aligned.jpg 3
/102345469_db6c79137e_m_0_aligned.jpg 3
/102345469_db6c79137e_m_1_aligned.jpg 3
```

Lmdb

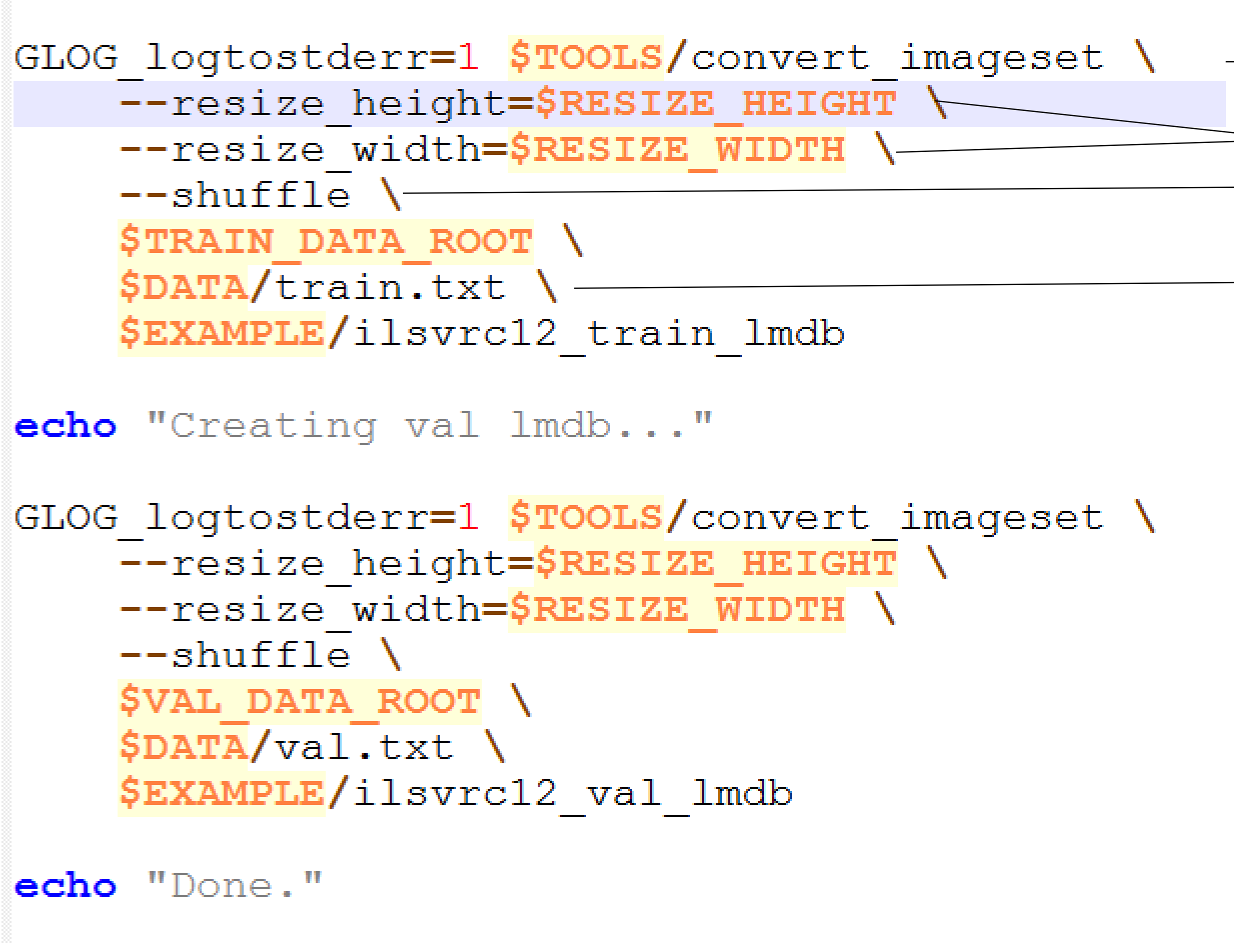
data.mdb　lock.mdb

Leveldb

LOCK　LOG.old

读取更快

训练模型-准备数据



```
GLOG_logtostderr=1 $TOOLS/convert_imageset \
    --resize_height=$RESIZE_HEIGHT \
    --resize_width=$RESIZE_WIDTH \
    --shuffle \
    $TRAIN_DATA_ROOT \
    $DATA/train.txt \
    $EXAMPLE/ilsvrc12_train_lmdb

echo "Creating val lmdb..."

GLOG_logtostderr=1 $TOOLS/convert_imageset \
    --resize_height=$RESIZE_HEIGHT \
    --resize_width=$RESIZE_WIDTH \
    --shuffle \
    $VAL_DATA_ROOT \
    $DATA/val.txt \
    $EXAMPLE/ilsvrc12_val_lmdb

echo "Done."
```

tools/create_imagenet.cpp

改变原始图像的尺寸

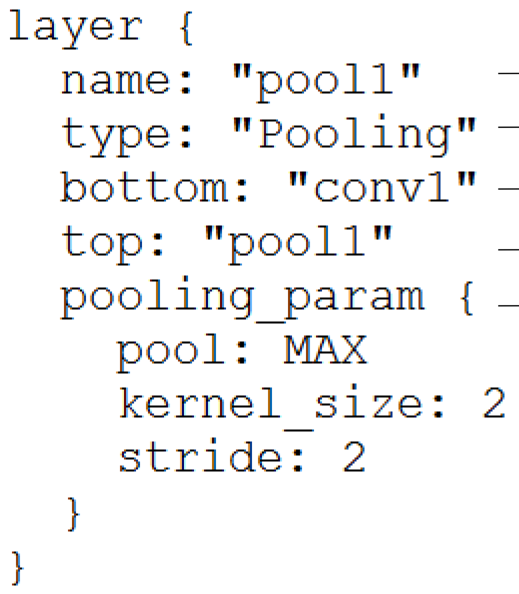
随机的奖图片和对应的标签顺序打乱

图片列表(文件名 标签)，标签最小值是0

例:examples/imagenet/creat_imagenet.sh

训练模型-建立模型

在prototxt文件里面定义网络



```
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 2
    stride: 2
  }
}
```

层的名字

层的类型

输入的blob

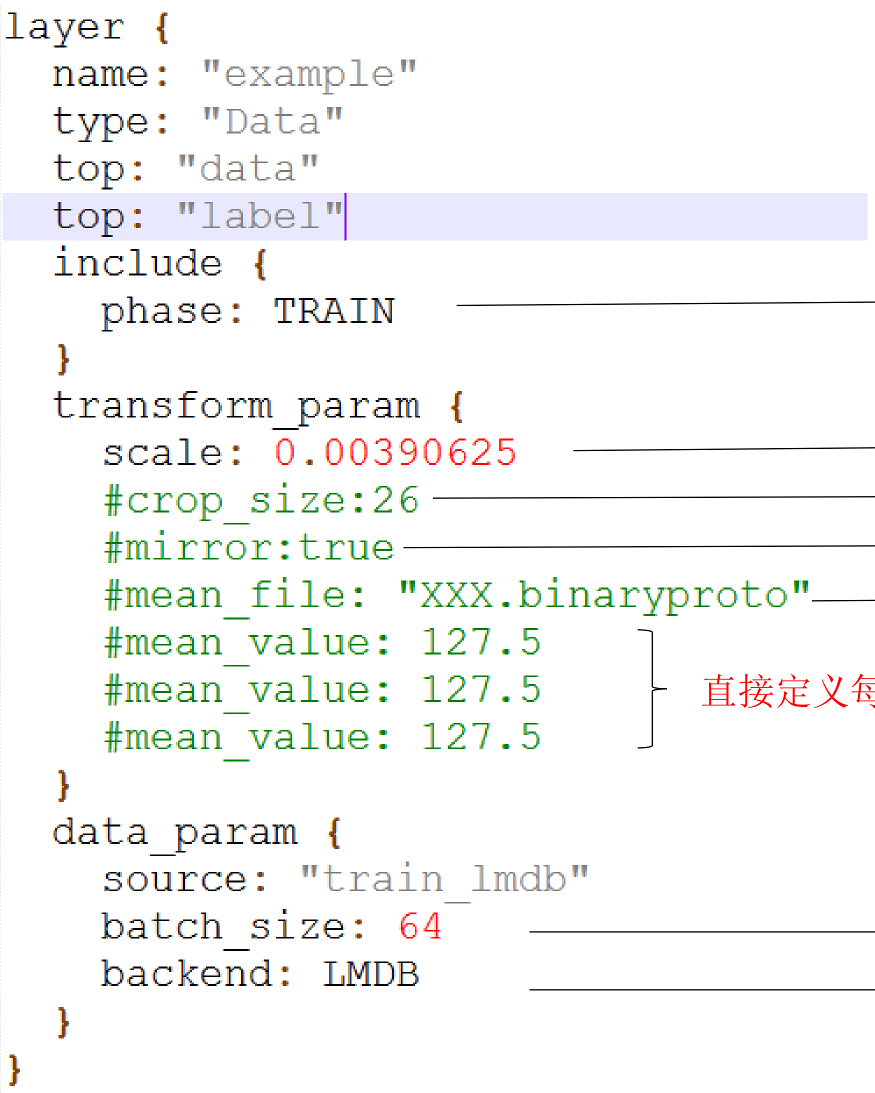
输出的blob

层的参数

例:examples/mnist/lenet_train_test.prototxt

训练模型-建立模型

数据层



```
layer {
  name: "example"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
    scale: 0.00390625
    #crop_size:26
    #mirror:true
    #mean_file: "XXX.binaryproto"
    #mean_value: 127.5
    #mean_value: 127.5
    #mean_value: 127.5
  }
  data_param {
    source: "train_lmdb"
    batch_size: 64
    backend: LMDB
  }
}
```

定义该层属于**训练网络**还是**测试网络**，如果不定义，则该层在训练网络和测试网络中都会存在。

对输入进行缩放(像素值*scale)

将输入图像进行随机裁剪

将输入图像进行水平的翻转

非常管用的数据增强方式

每个通道减去对应通道的均值，根据训练集计算出来的均值文件

例： examples/imagenet/make_imagenet_mean.sh

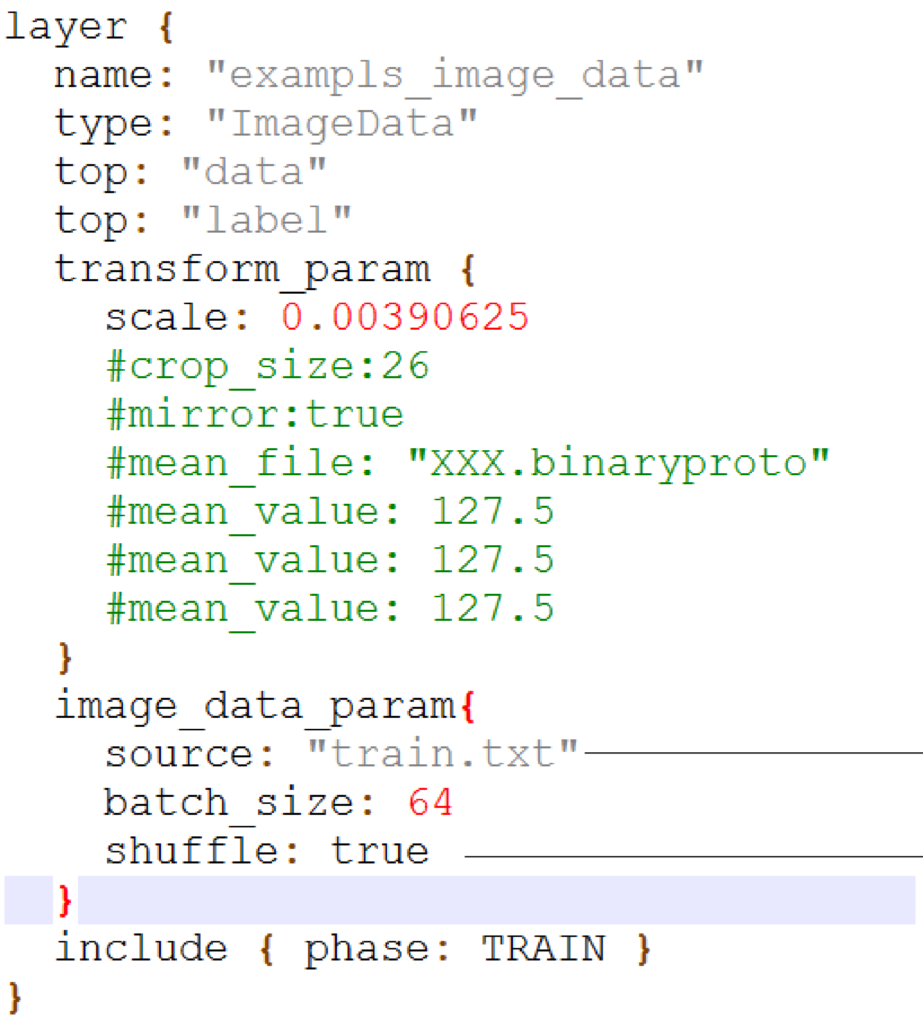
直接定义每个通道的均值

每个训练批次的大小

说明使用的数据格式

训练模型-建立模型

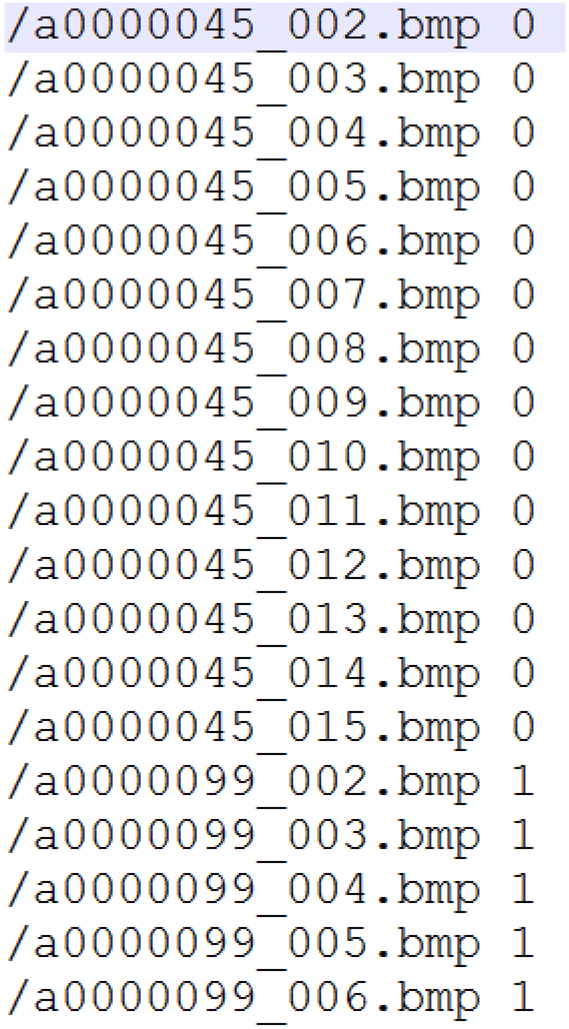
图像数据层



```
layer {
  name: "exampls_image_data"
  type: "ImageData"
  top: "data"
  top: "label"
  transform_param {
    scale: 0.00390625
    #crop_size:26
    #mirror:true
    #mean_file: "XXX.binaryproto"
    #mean_value: 127.5
    #mean_value: 127.5
    #mean_value: 127.5
  }
  image_data_param{
    source: "train.txt"
    batch_size: 64
    shuffle: true
  }
  include { phase: TRAIN }
}
```

源文件，定义文件路径，文件名和标签

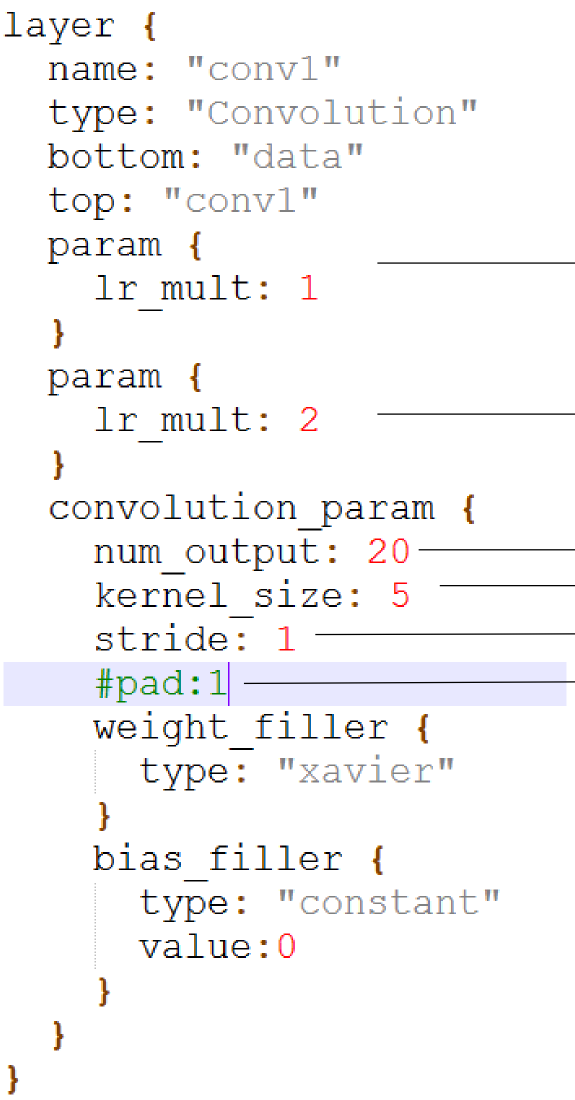
根据列表把训练数据和标签打乱



```
/a0000045_002.bmp 0
/a0000045_003.bmp 0
/a0000045_004.bmp 0
/a0000045_005.bmp 0
/a0000045_006.bmp 0
/a0000045_007.bmp 0
/a0000045_008.bmp 0
/a0000045_009.bmp 0
/a0000045_010.bmp 0
/a0000045_011.bmp 0
/a0000045_012.bmp 0
/a0000045_013.bmp 0
/a0000045_014.bmp 0
/a0000045_015.bmp 0
/a0000099_002.bmp 1
/a0000099_003.bmp 1
/a0000099_004.bmp 1
/a0000099_005.bmp 1
/a0000099_006.bmp 1
```

训练模型-建立模型

卷积层



```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
    #pad:1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
      value:0
    }
  }
}
```

卷积核参数学习速率系数

偏置参数学习速率系数

Learning_rate=lr_mult*base_lr

输出的通道数

卷积核大小

卷积步长

补0

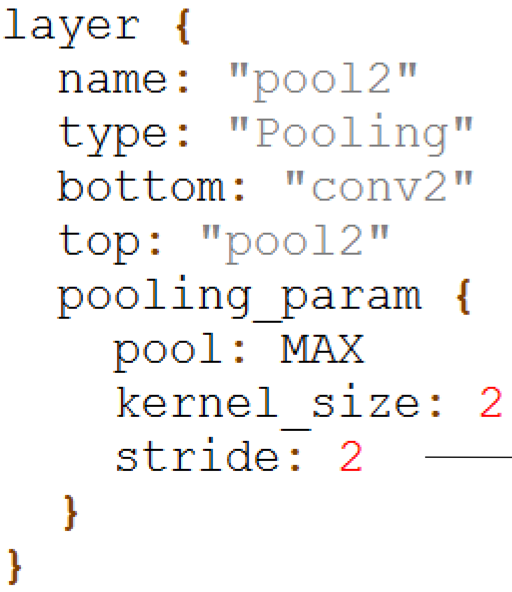
参数初始化方法

gaussian
xavier
msra
⋮

include/caffe/filler.hpp

训练模型-建立模型

池化层



```
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: MAX
    kernel_size: 2
    stride: 2
  }
}
```

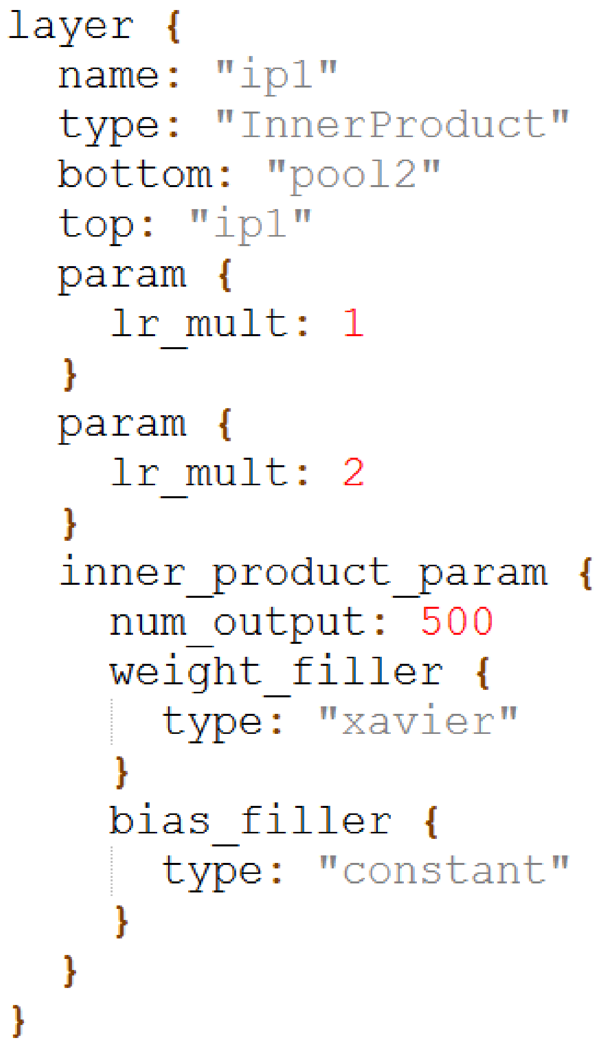
下采样方式

采样区域大小

采样步长

训练模型-建立模型

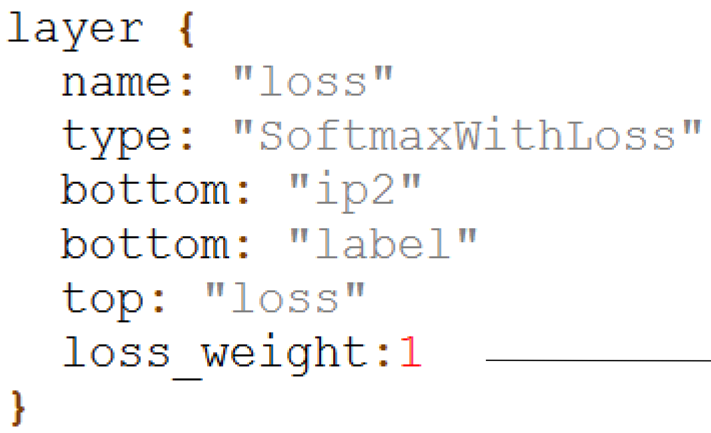
全连接层



```
layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "pool2"
  top: "ip1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 500
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
```

训练模型-建立模型

损失层

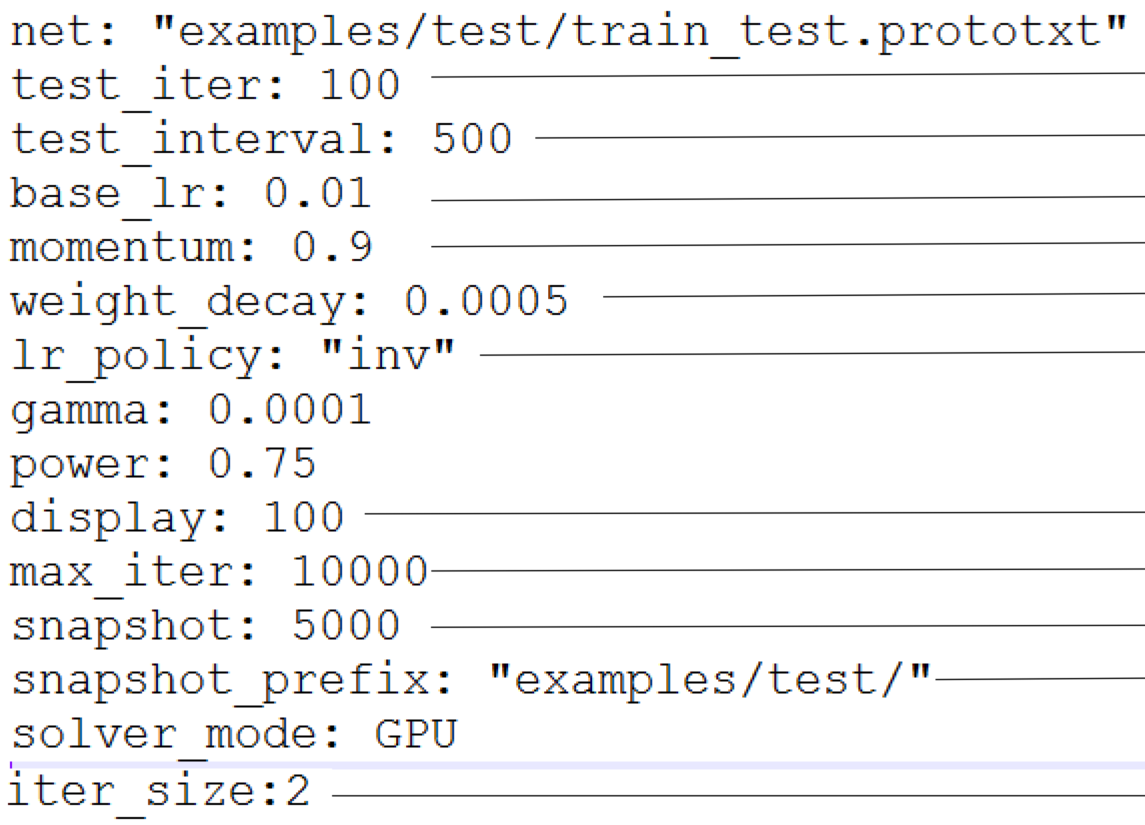


```
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip2"
  bottom: "label"
  top: "loss"
  loss_weight:1
}
```

损失系数

训练模型-配置网络训练参数

Solver-设置网络训练参数及优化方法



```
net: "examples/test/train_test.prototxt"
test_iter: 100
test_interval: 500
base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
lr_policy: "inv"
gamma: 0.0001
power: 0.75
display: 100
max_iter: 10000
snapshot: 5000
snapshot_prefix: "examples/test/"
solver_mode: GPU
iter_size:2
```

网络文件

测试时的迭代次数。测试图片数量=test_iter*batchsize(test)

每完成500次迭代的训练，就进行测试

最初的学习速率

冲量

权重衰减系数

学习速率更新策略

fixed
step
inv
⋮
src/caffe/solver/sgd_solver.cpp

每完成100次迭代，显示当前结果(损失大小，学习速率)

最大迭代次数

每完成5000次迭代，保存模型参数

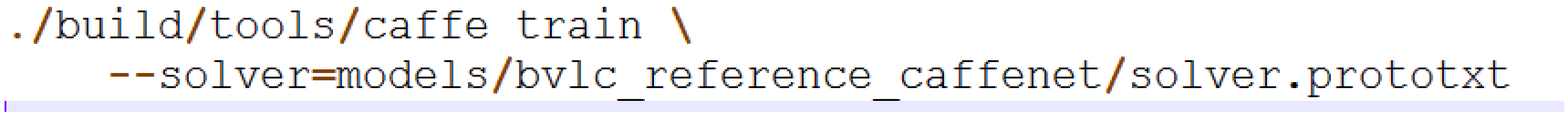
模型保存路径

前传iter_size*batchsize个样本后再计算梯度，当显存资源有限时使用

solver.prototxt

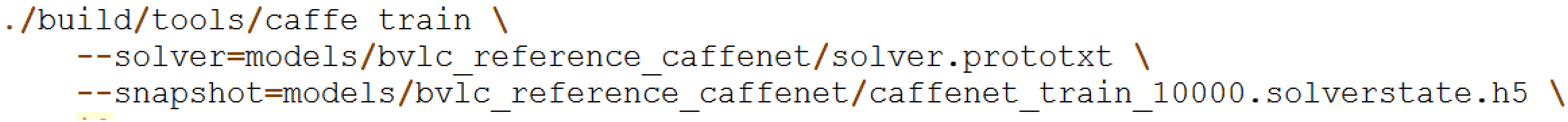
训练模型-训练

train.sh 参数根据网络配置文件里设定的初始化方法初始化



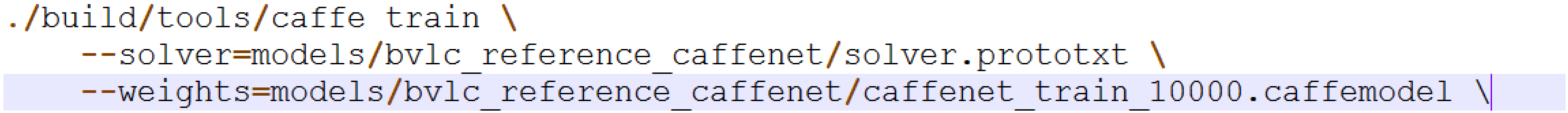
```
./build/tools/caffe train \
    --solver=models/bvlc_reference_caffenet/solver.prototxt
```

resume.sh 从某次迭代开始恢复之前的训练



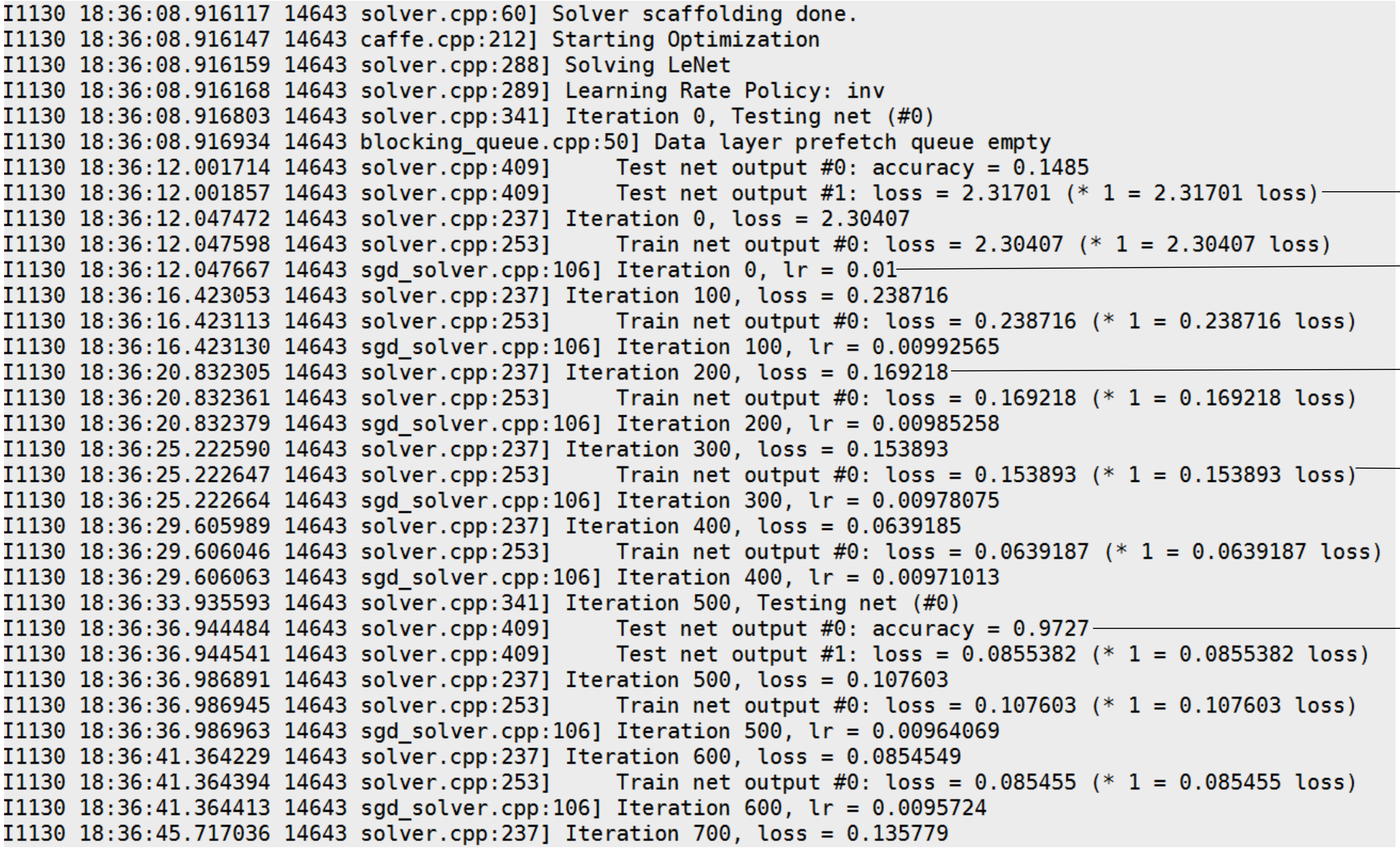
```
./build/tools/caffe train \
    --solver=models/bvlc_reference_caffenet/solver.prototxt \
    --snapshot=models/bvlc_reference_caffenet/caffenet_train_10000.solverstate.h5 \
```

fintune.sh 使用预训练的网络参数来初始化。通常在改变部分网络结构和在新的数据集合上训练时使用



```
./build/tools/caffe train \
    --solver=models/bvlc_reference_caffenet/solver.prototxt \
    --weights=models/bvlc_reference_caffenet/caffenet_train_10000.caffemodel \
```

训练模型-训练



```
I1130 18:36:08.916117 14643 solver.cpp:60] Solver scaffolding done.
I1130 18:36:08.916147 14643 caffe.cpp:212] Starting Optimization
I1130 18:36:08.916159 14643 solver.cpp:288] Solving LeNet
I1130 18:36:08.916168 14643 solver.cpp:289] Learning Rate Policy: inv
I1130 18:36:08.916803 14643 solver.cpp:341] Iteration 0, Testing net (#0)
I1130 18:36:08.916934 14643 blocking_queue.cpp:50] Data layer prefetch queue empty
I1130 18:36:12.001714 14643 solver.cpp:409]     Test net output #0: accuracy = 0.1485
I1130 18:36:12.001857 14643 solver.cpp:409]     Test net output #1: loss = 2.31701 (* 1 = 2.31701 loss)
I1130 18:36:12.047472 14643 solver.cpp:237] Iteration 0, loss = 2.30407
I1130 18:36:12.047598 14643 solver.cpp:253]     Train net output #0: loss = 2.30407 (* 1 = 2.30407 loss)
I1130 18:36:12.047667 14643 sgd_solver.cpp:106] Iteration 0, lr = 0.01
I1130 18:36:16.423053 14643 solver.cpp:237] Iteration 100, loss = 0.238716
I1130 18:36:16.423113 14643 solver.cpp:253]     Train net output #0: loss = 0.238716 (* 1 = 0.238716 loss)
I1130 18:36:16.423130 14643 sgd_solver.cpp:106] Iteration 100, lr = 0.00992565
I1130 18:36:20.832305 14643 solver.cpp:237] Iteration 200, loss = 0.169218
I1130 18:36:20.832361 14643 solver.cpp:253]     Train net output #0: loss = 0.169218 (* 1 = 0.169218 loss)
I1130 18:36:20.832379 14643 sgd_solver.cpp:106] Iteration 200, lr = 0.00985258
I1130 18:36:25.222590 14643 solver.cpp:237] Iteration 300, loss = 0.153893
I1130 18:36:25.222647 14643 solver.cpp:253]     Train net output #0: loss = 0.153893 (* 1 = 0.153893 loss)
I1130 18:36:25.222664 14643 sgd_solver.cpp:106] Iteration 300, lr = 0.00978075
I1130 18:36:29.605989 14643 solver.cpp:237] Iteration 400, loss = 0.0639185
I1130 18:36:29.606046 14643 solver.cpp:253]     Train net output #0: loss = 0.0639187 (* 1 = 0.0639187 loss)
I1130 18:36:29.606063 14643 sgd_solver.cpp:106] Iteration 400, lr = 0.00971013
I1130 18:36:33.935593 14643 solver.cpp:341] Iteration 500, Testing net (#0)
I1130 18:36:36.944484 14643 solver.cpp:409]     Test net output #0: accuracy = 0.9727
I1130 18:36:36.944541 14643 solver.cpp:409]     Test net output #1: loss = 0.0855382 (* 1 = 0.0855382 loss)
I1130 18:36:36.986891 14643 solver.cpp:237] Iteration 500, loss = 0.107603
I1130 18:36:36.986945 14643 solver.cpp:253]     Train net output #0: loss = 0.107603 (* 1 = 0.107603 loss)
I1130 18:36:36.986963 14643 sgd_solver.cpp:106] Iteration 500, lr = 0.00964069
I1130 18:36:41.364229 14643 solver.cpp:237] Iteration 600, loss = 0.0854549
I1130 18:36:41.364394 14643 solver.cpp:253]     Train net output #0: loss = 0.085455 (* 1 = 0.085455 loss)
I1130 18:36:41.364413 14643 sgd_solver.cpp:106] Iteration 600, lr = 0.0095724
I1130 18:36:45.717036 14643 solver.cpp:237] Iteration 700, loss = 0.135779
```

测试的损失

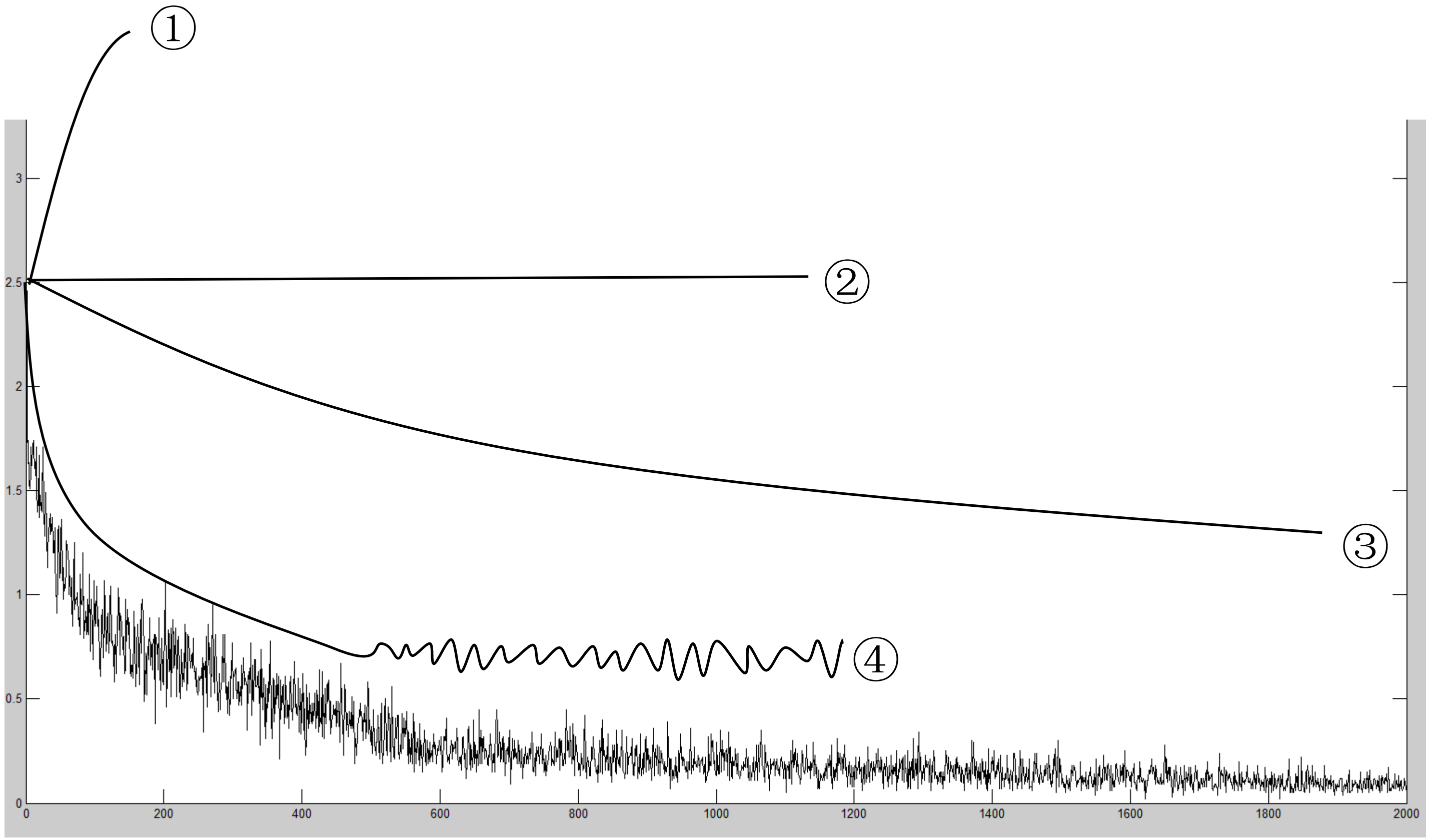
当前学习速率

平滑处理的损失

当前迭代次数损失

测试的分类准确率

训练模型-学习速率调节



① 初始学习率过大

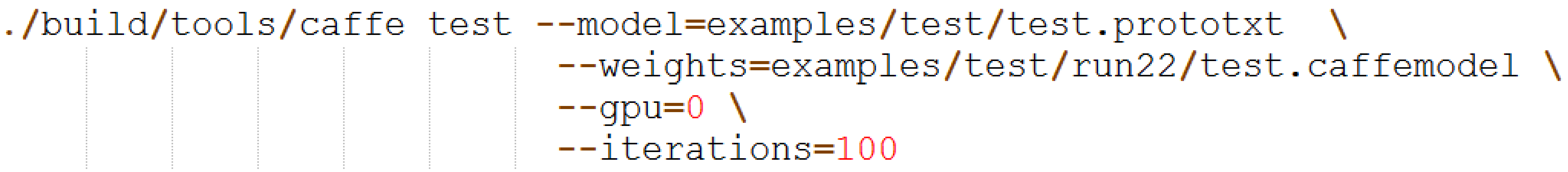
②参数初始化方式不合适

③初始学习率过小

④需要适当减小学习

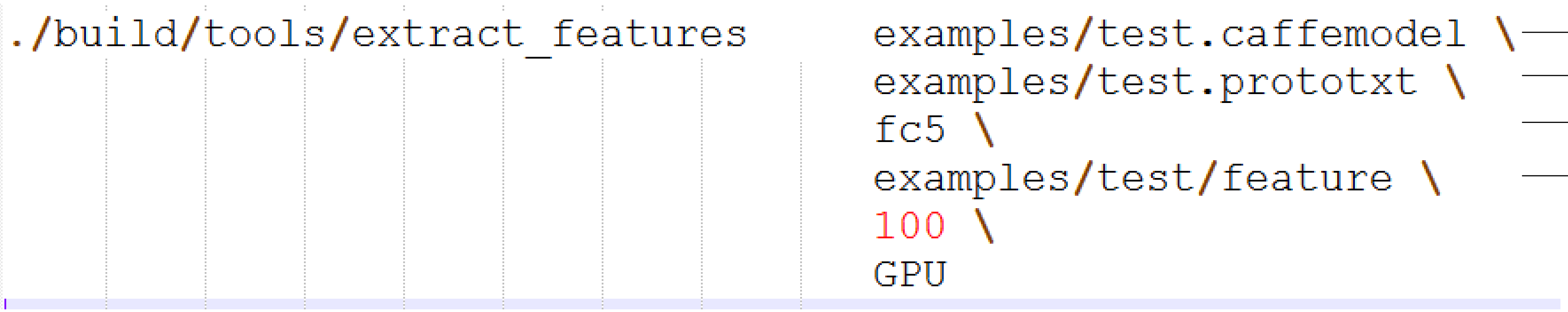
训练模型-测试

test.sh



```
./build/tools/caffe test --model=examples/test/test.prototxt  \
                          --weights=examples/test/run22/test.caffemodel \
                          --gpu=0 \
                          --iterations=100
```

extract_features.sh



```
./build/tools/extract_features     examples/test.caffemodel \
                                   examples/test.prototxt \
                                   fc5 \
                                   examples/test/feature \
                                   100 \
                                   GPU
```

模型保存路径

网络文件

需要提取输出的层的名字

提取特征保存的路径

进一步了解caffe

参数定义： src/caffe/proto/caffe.proto

层的实现： src/caffe/layers

层的声明： include/caffe/layers

参数初始化方法： include/caffe/filler.hpp

模型的优化: src/caffe/solvers/sgd_solver.cpp

数学函数: src/caffe/util/math_functions.cpp

谢谢！！