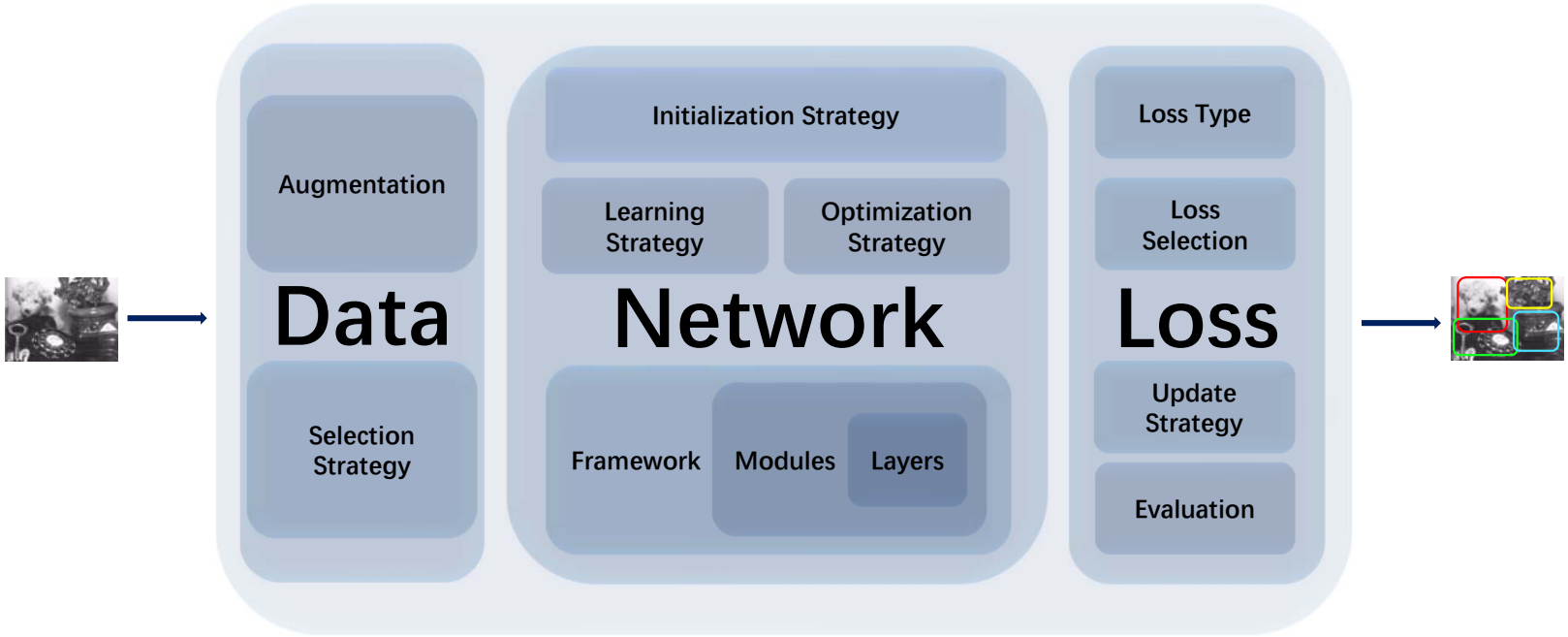
CNN for CV

AI for CV Group
2019


How To Use
Caffe & PyTorch

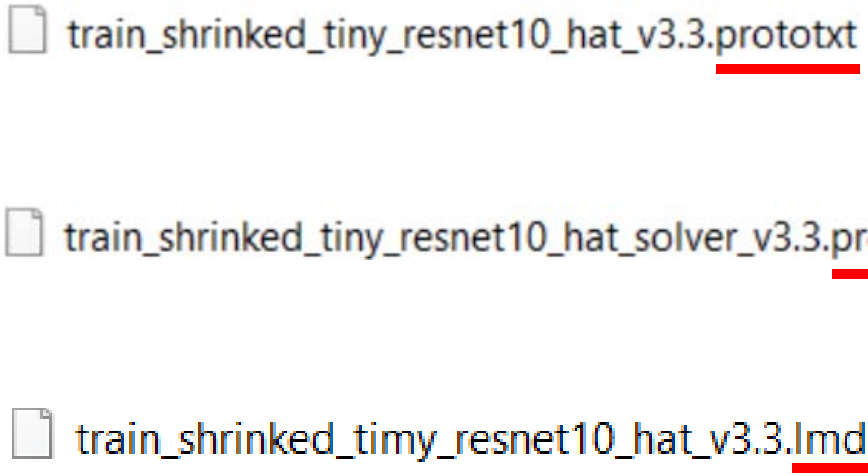

Augmentation
Data
Selection Strategy
Initialization Strategy
Learning Strategy
Optimization Strategy
Network
Framework
Modules
Layers
Loss Type
Loss Selection
Loss
Update Strategy
Evaluation

Contents:

I. Caffe

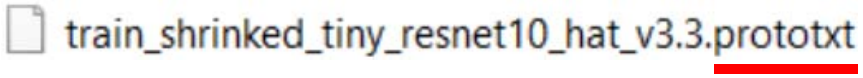
I. Training With Caffe

- Three Parts:



I. Training With Caffe

- Three Parts: Net



Net

Your main CNN structure

I. Training With Caffe

- Three Parts: Net-Training Data

```
1  name: "tiny_resnet"
2  #input: "data"
3  #input_dim: 1
4  #input_dim: 3
5  #input_dim: 56
6  #input_dim: 56
7  layer {
8    name: "data"
9    type: "Data"
10   top: "data"
11   top: "label"
12   include {
13     phase: TRAIN
14   }
15   transform_param {
16   #  mean_file: "/data/mean.binaryproto"
17     mirror:true
18   }
19   data_param {
20     source: "/train_hat_v3.3.lmdb"
21     batch_size: 16          # 32
22     backend: LMDB
23   }
24 }
```

网络名字

对于Test阶段

勘误: 对于Inference阶段

第一层是数据层

表明这一层是针对Training阶段

一些简单的数据增强/预处理

数据来源

I. Training With Caffe

- Three Parts: Net-Testing Data

```
25  layer {
26    name: "data"
27    type: "Data"
28    top: "data"
29    top: "label"
30    include {
31      phase: TEST 检测阶段
32    }
33    #transform_param {
34    #  mean_file: "/data/mean.binaryproto"
35    #}
36    data_param {
37      source: "/valid_hat_v3.3.lmdb"
38      batch_size: 16          # 16
39      backend: LMDB 检测阶段数据来源
40    }
41  }
42  ####################################
```

I. Training With Caffe

- Three Parts: Net-Conv

```
43  layer {
44      bottom: "data"
45      top: "conv1"
46      name: "conv1"
47      type: "Convolution"
48      convolution_param {
49          num_output: 32
50          kernel_size: 3
51          pad: 1
52          stride: 1
53          weight_filler {
54              type: "msra"
55          }
56          bias_term: false
57
58      }
59  }
```

承接data层

输出conv层

Conv层

kaiming_normal

I. Training With Caffe

- Three Parts: Net-Batch Norm(Train)

```
61  layer {
62      bottom: "conv1"
63      top: "conv1"
64      name: "bn_conv1"
65      type: "BatchNorm"
66      batch_norm_param {
67        moving_average_fraction: 0.9
68      } average的momentum
69  }
70
71  layer {
72      bottom: "conv1"
73      top: "conv1"
74      name: "scale_conv1"
75      type: "Scale"
76      scale_param {
77          bias_term: true
78      }
79  }
```

batchnorm层为了获得N(0,1)数据

两层合起来组成完整的batch_norm层

scale层为了得到gamma和beta

I. Training With Caffe

- Three Parts: Net-Batch Norm(Test)

```
62  layer {
63      bottom: "conv1"
64      top: "conv1"
65      name: "bn_conv1"
66      type: "BatchNorm"
67      batch_norm_param {
68          #moving_average_fraction: 0.9
69          use_global_stats: 1
70      }
71  }
```

I. Training With Caffe

- Three Parts: Net-ReLU

```
81  layer {
82      bottom: "conv1"
83      top: "conv1"
84      name: "conv1_relu"  ReLU层
85      type: "ReLU"
86  }
```

I. Training With Caffe

- Three Parts: Net-Pooling

```
88  #layer {
89  #    bottom: "conv1"
90  #    top: "pool1"
91  #    name: "pool1"
92  #    type: "Pooling"
93  #    pooling_param {
94  #        kernel_size: 3
95  #        stride: 2
96  #        pool: MAX
97  #    }
98  #}
```

Pooling层
可以这样写

用#起到注释
作用，所以这
里pooling没
有作用

Pooling方式

I. Training With Caffe

- Three Parts: Net-Pooling

```
721  layer {
722    name: "pool5"
723    type: "Pooling"
724    bottom: "res4a_branch2b"
725    top: "pool5"
726    pooling_param {
727      pool: AVE
728      global_pooling: true
729    }
730  }
```

其他Pooling方式

I. Training With Caffe

- Three Parts: Net-Elementwise

```
469  layer {
470      bottom: "res3a_branch1"
471      bottom: "res3a_branch2b"
472      top: "res3a"
473      name: "res3a"
474      type: "Eltwise"
475      eltwise_param {
476          operation: SUM
477      }
478  }
```

确定好哪两层做操作

Element-wise层

目的是做Element-wise相加

I. Training With Caffe

- Three Parts: Net-Softmax

```
732  #layer {
733  #  name: "prob"
734  #  type: "Softmax"
735  #  bottom: "pool5"
736  #  top: "prob"
737  #}
738
739  layer {
740    name: "loss"
741    type: "SoftmaxWithLoss"
742    bottom: "pool5"
743    bottom: "label"
744    top: "loss"
745  }
```

对于Inference阶段，
无需回传loss

对于training阶段，需要回传loss

I. Training With Caffe

- Three Parts: Net-Accuracy

```
746 layer {
747     bottom: "pool5"
748     bottom: "label"
749     top: "acc/top-1"
750     top: "class"
751     name: "acc/top-1"
752     type: "Accuracy"
753     include {
754         phase: TEST
755     }
756     include {
757         phase: TRAIN
758     }
759 }
```

top-n: 前n个预测结果里有正确结果的概率

我们关注top-1的accuracy

表明test/train阶段
都要计算accuracy

I. Training With Caffe

- Three Parts: Net-Summary

1. Like LEGO, just build your own structure.
   NO NEED TO CODE AT ALL

2. Usually, leave default values default
   [moving_average_fraction…]

3. Be careful of the sequence of dimensions of a blob:
   [N, C, W, H]

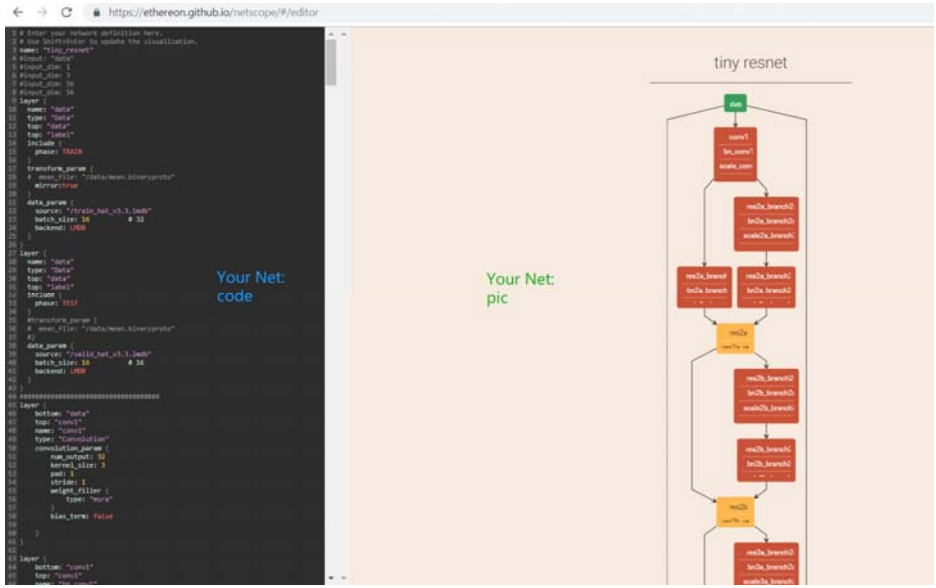
4. 3 nets actually. We can at least write Train / Test together by adding “Phase”
   [Train / Test / Inference]

I. Training With Caffe

- Three Parts: Net-Summary

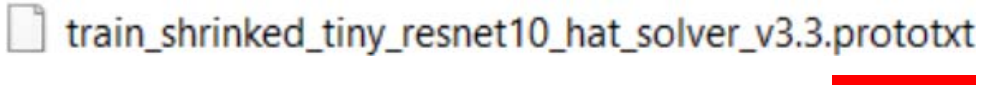
  5. See our structure:

     netscope: online plot system



I. Training With Caffe

- Three Parts: Solver



Solver

Your CNN hyperparameters

I. Training With Caffe

- Three Parts: Solver

```
net: "/week7.prototxt"

test_iter: 1000
test_interval: 500


base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
lr_policy: "multistep"
gamma: 0.2

max_iter: 100000
stepvalue: 30000
stepvalue: 60000

display: 100

snapshot: 2000
snapshot_prefix: "tiny_resnet_model/"

solver_mode: GPU
```

网络文件名字：要与写有网络的文件的名字相同；而不是与网络名字相同
表明要从这个文件当中读取网络

test_iter: test时，需要迭代次数 = 测试集大小 / 测试集的batchsize
test_interval：每迭代test_interval次，进行一次测试

learning rate变更的策略。这里展示的是multistep

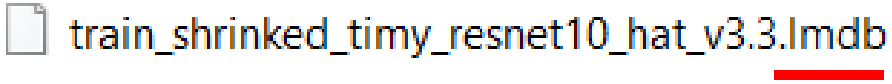
每迭代100次，显示一次训练结果

对训练好的model的存储。每迭代2000次，存一次；
存取路径加前缀是snapshot_prefix控制

显示表明是GPU来训练

I. Training With Caffe

- Three Parts: Data



Data

Your data going to be trained / tested

I. Training With Caffe

- Three Parts: Data-Type

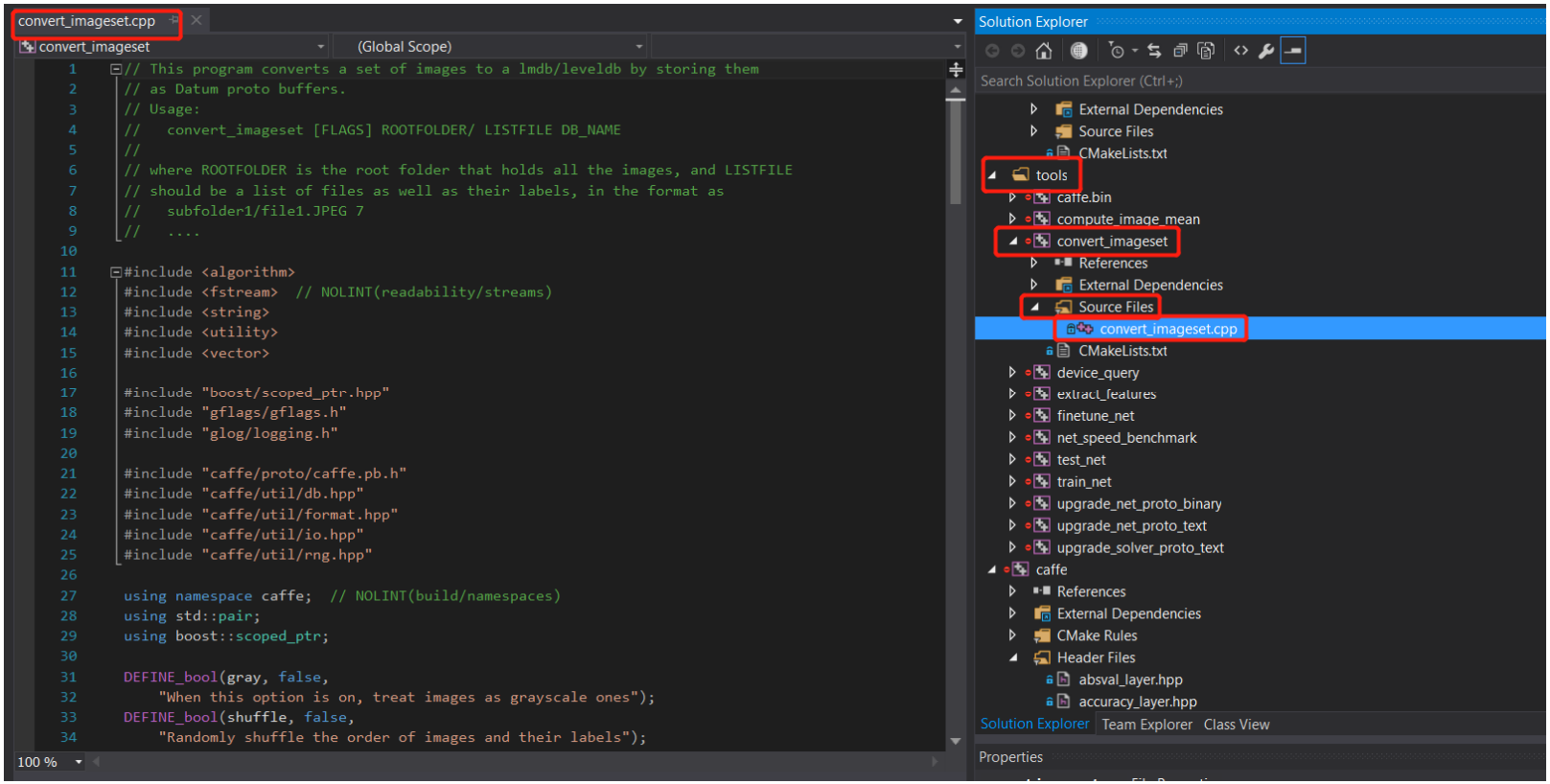
  .leveldb

  .lmdb

https://groups.google.com/forum/#!topic/caffe-users/At649aC3lks

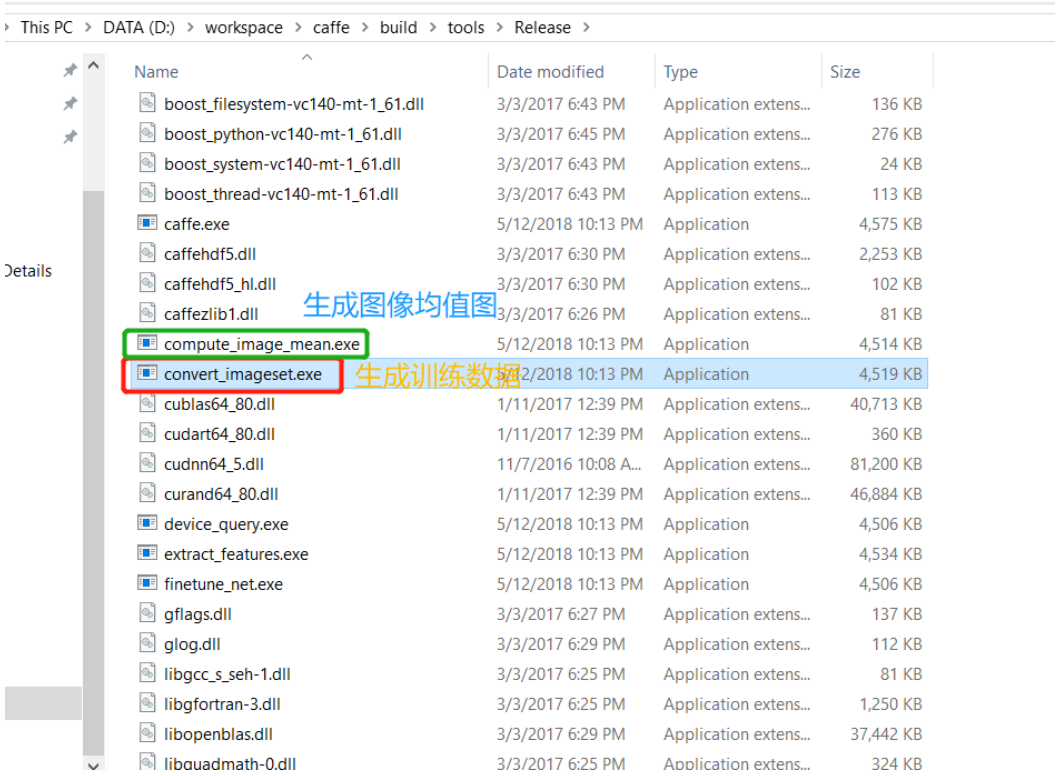
I. Training With Caffe

- Three Parts: Data-Generation



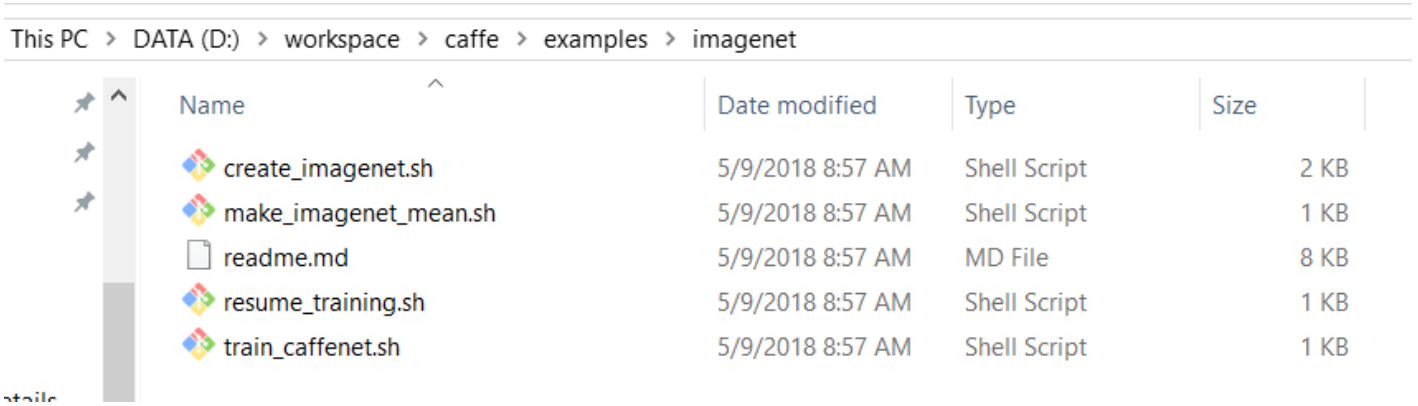
I. Training With Caffe

- Three Parts: Data-Generation(Windows)



I. Training With Caffe

- Three Parts: Data-Generation(Linux)



I. Training With Caffe

- Three Parts: Data-Generation
  - Need to use ***cmd*** to generate
  - Can write a bash file to run

create_hat_db_v3.3.sh

```
ID=v3.3
ATTR=hat

CAFFE_ROOT=/home/workspace/caffe/build/tools
SRC_DATA_ROOT=/home/workspace/human_attribute/v3/${ATTR}/${ID}
DST_DATA_ROOT=/v3/lmdbs/${ATTR}/${ID}

CREATE_DB=${CAFFE_ROOT}/convert_imageset
TRAIN_LIST=${SRC_DATA_ROOT}/train_${ATTR}_${ID}.txt
VALID_LIST=${SRC_DATA_ROOT}/valid_${ATTR}_${ID}.txt
TRAIN_DB=${DST_DATA_ROOT}/${ATTR}_${ID}_lmdb/train_${ATTR}_${ID}_lmdb
VALID_DB=${DST_DATA_ROOT}/${ATTR}_${ID}_lmdb/valid_${ATTR}_${ID}_lmdb

BACK_END=lmdb

RESIZE_WIDTH=56
RESIZE_HEIGHT=56
RESIZE_BORDER=56

echo "Create train lmdb..."
rm -rf ${TRAIN_DB}

${CREATE_DB} --resize_width=${RESIZE_BORDER} --resize_height=${RESIZE_BORDER} --backend=${BACK_END} "" ${TRAIN_LIST} ${TRAIN_DB}
echo "Create valid lmdb..."
rm -rf ${VALID_DB}
${CREATE_DB} --resize_width=${RESIZE_BORDER} --resize_height=${RESIZE_BORDER} --backend=${BACK_END} "" ${VALID_LIST} ${VALID_DB}
echo "All Done!"
```

I. Training With Caffe

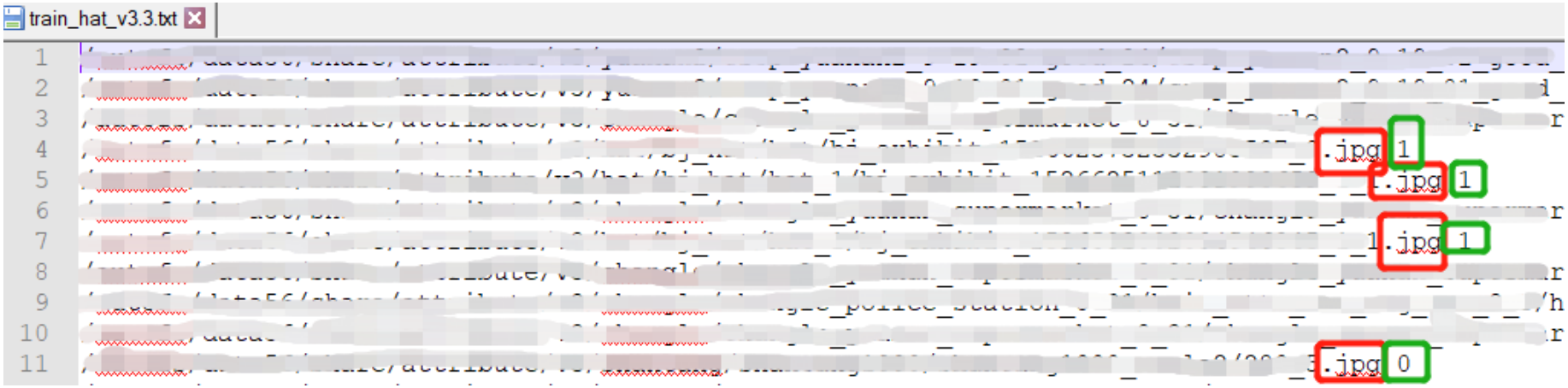
- Three Parts: Data-Generation
  - Need to use ***cmd*** to generate
  - Can write a bash file to run

create_hat_db_v3.3.sh

```
ID=v3.3
ATTR=hat

CAFFE_ROOT=/home/workspace/caffe/build/tools
SRC_DATA_ROOT=/home/workspace/human_attribute/v3/${ATTR}/${ID}
DST_DATA_ROOT=/v3/lmdbs/${ATTR}/${ID}

CREATE_DB=${CAFFE_ROOT}/convert_imageset
TRAIN_LIST=${SRC_DATA_ROOT}/train_${ATTR}_${ID}.txt
VALID_LIST=${SRC_DATA_ROOT}/valid_${ATTR}_${ID}.txt
TRAIN_DB=${DST_DATA_ROOT}/${ATTR}_${ID}_lmdb/train_${ATTR}_${ID}_lmdb
VALID_DB=${DST_DATA_ROOT}/${ATTR}_${ID}_lmdb/valid_${ATTR}_${ID}_lmdb

BACK_END=lmdb

RESIZE_WIDTH=56
RESIZE_HEIGHT=56
RESIZE_BORDER=56

echo "Create train lmdb..."
rm -rf ${TRAIN_DB}

${CREATE_DB} --resize_width=${RESIZE_BORDER} --resize_height=${RESIZE_BORDER} --backend=${BACK_END} "" ${TRAIN_LIST} ${TRAIN_DB}
echo "Create valid lmdb..."
rm -rf ${VALID_DB}
${CREATE_DB} --resize_width=${RESIZE_BORDER} --resize_height=${RESIZE_BORDER} --backend=${BACK_END} "" ${VALID_LIST} ${VALID_DB}
echo "All Done!"
```

I. Training With Caffe

- Three Parts: Data-Origin



I. Training With Caffe

- Three Parts: Data-Summary

  1. It's up to you to choose leveldb or lmdb

  2. It's up to you whether you want to compute mean image

  3. Use imageset.exe directly if you just need a classification

  4. **You have to code yourself if you are not just doing a simple classification**
     Rewrite "convert_imageset" is a good choice.

I. Training With Caffe

- Training Command

```
1  ID=v3.3
2  BACKBONE=tiny_resnet
3  ATTR=hat
4
5  CAFFE_ROOT=/home/workspace/caffe/build/tools
6  SOLVER_ROOT=/home/workspace/human_attribute/v3/${ATTR}/${ID}/${BACKBONE}
7
8  TRAIN=${CAFFE_ROOT}/caffe
9  SOLVER=${SOLVER_ROOT}/train_${BACKBONE}_${ATTR}_solver_${ID}.prototxt
10 LOG=${SOLVER_ROOT}/${ATTR}_${ID}_${BACKBONE}_log/${ATTR}_${ID}_${BACKBONE}_log.log
11 GPU=4,5
12
13 ${TRAIN} train --solver=${SOLVER} --gpu=${GPU} 2>&1 | tee ${LOG}
14
```

必写的

可选的

I. Training With Caffe

- Advanced Techniques:

  1. Rewrite data generating code
  2. Finetune / Retraining using trained model
  3. Add new existing layers
  4. Freeze / Share layers
  5. Write your own layers
  6. Write your own layers in cudnn

I. Training With Caffe

- Panorama Of Caffe:

II. Training With PyTorch

II. Training With PyTorch

- Let’s see code directly: