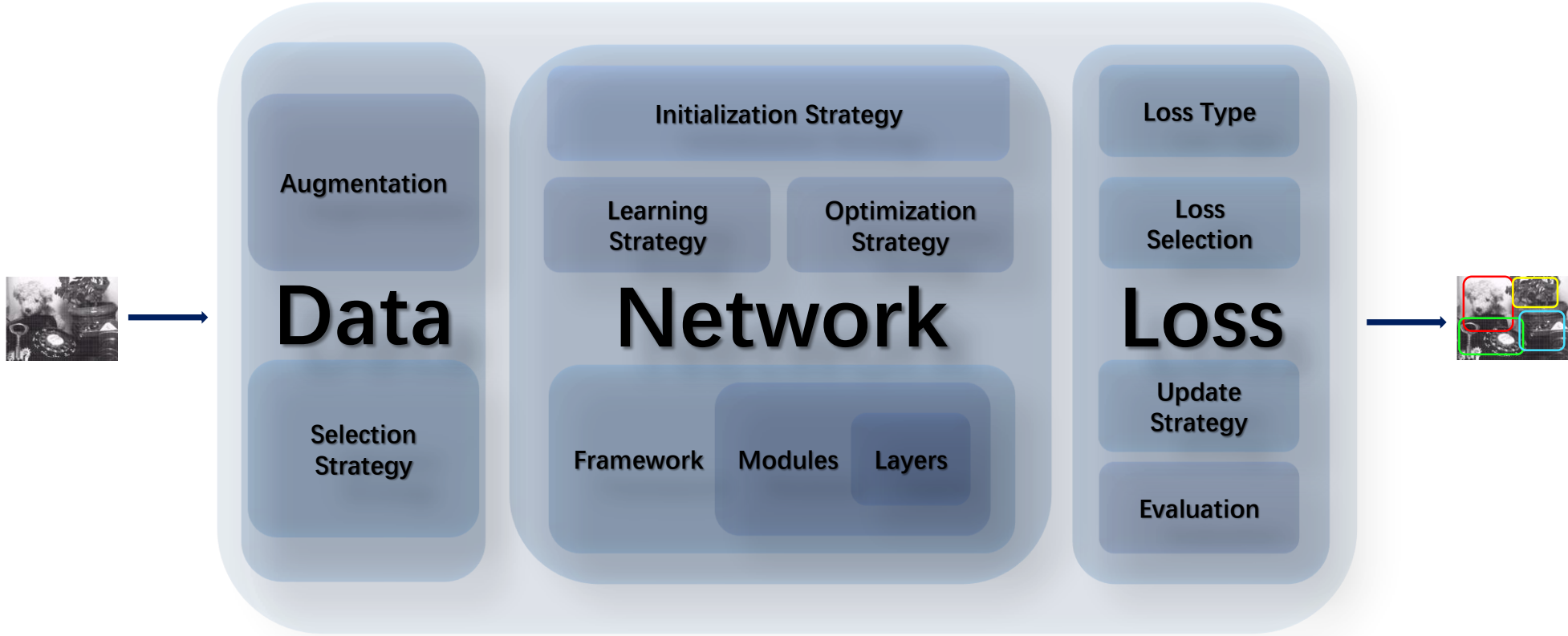


CNN for CV

AI for CV Group
2019



How To Use Caffe & PyTorch



Contents:

I. Caffe

- A. Net
- B. Solver
- C. Data

II. PyTorch


- D. Let's see code directly

I. Caffè




I. Training With Caffe

- Three Parts:

 `train_shrunked_tiny_resnet10_hat_v3.3.prototxt`

Net

 `train_shrunked_tiny_resnet10_hat_solver_v3.3.prototxt`


Solver

 `train_shrunked_tiny_resnet10_hat_v3.3.lmdb`

Data

I. Training With Caffe

- Three Parts: Net

 train_shrunked_tiny_resnet10_hat_v3.3.prototxt

Net

Your main CNN structure

I. Training With Caffe

- Three Parts: Net-Training Data

```
1 name: "tiny_resnet" 网络名字
2 #input: "data"
3 #input_dim: 1
4 #input_dim: 3 对于Test阶段 勘误: 对于Inference阶段
5 #input_dim: 56
6 #input_dim: 56
7 layer {
8   name: "data"
9   type: "Data" 第一层是数据层
10  top: "data"
11  top: "label"
12  include {
13    phase: TRAIN 表明这一层是针对Training阶段
14  }
15  transform_param {
16    # mean_file: "/data/mean.binaryproto"
17    mirror: true 一些简单的数据增强/预处理
18  }
19  data_param {
20    source: "/train_hat_v3.3.lmdb"
21    batch_size: 16 数据来源 # 32
22    backend: LMDB
23  }
24 }
```


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" 承接data层
45     top: "conv1" 输出conv层
46     name: "conv1"
47     type: "Convolution" Conv层
48     convolution_param {
49         num_output: 32
50         kernel_size: 3
51         pad: 1
52         stride: 1
53         weight_filler {
54             type: "msra" kaiming_normal
55         }
56         bias_term: false
57     }
58 }
59 }
```

I. Training With Caffe

- Three Parts: Net-Batch Norm(Train)

```
61 layer {
62     bottom: "conv1"
63     top: "conv1"
64     name: "bn_conv1"      batchnorm层为了
65     type: "BatchNorm"    获得N(0,1)数据
66     batch_norm_param {
67         moving_average_fraction: 0.9
68     } average的momentum
69 }
70     两层合起来组成完整的batch_norm层
71 layer {
72     bottom: "conv1"
73     top: "conv1"
74     name: "scale_conv1"  scale层为了得到
75     type: "Scale"        gamma和beta
76     scale_param {
77         bias_term: true
78     }
79 }
```

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" 确定好哪两层  
471     bottom: "res3a_branch2b" 做操作  
472     top: "res3a"  
473     name: "res3a"           Element-wise层  
474     type: "Eltwise"  
475     eltwise_param {         目的是做  
476         operation: SUM      Element-wise相加  
477     }  
478 }
```


I. Training With Caffe

- Three Parts: Net-Softmax

```
732 #layer {  
733 #   name: "prob"  
734 #   type: "Softmax"  
735 #   bottom: "pool5"  
736 #   top: "prob"  
737 #}
```

对于Inference阶段,
无需回传loss

```
738  
739 layer {  
740   name: "loss"  
741   type: "SoftmaxWithLoss"  
742   bottom: "pool5"  
743   bottom: "label"  
744   top: "loss"  
745 }
```

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

The screenshot displays the Caffe Net-Summary interface, which is used for visualizing Caffe network definitions. The interface is divided into two main sections: "Your Net: code" on the left and "Your Net: pic" on the right.


Left Panel (Your Net: code): This panel shows the Caffe net definition for a "tiny_resnet" model. The code is as follows:

```
1 # Enter your network definition here.
2 # Use Shift-Enter to update the visualization.
3 name: "tiny_resnet"
4 input: "data"
5 #input_dim: 1
6 #input_dim: 3
7 #input_dim: 56
8 #input_dim: 56
9 layer {
10   name: "data"
11   type: "Data"
12   top: "data"
13   top: "label"
14   include {
15     phase: TRAIN
16   }
17   transform_param {
18     # mean_file: "/data/mean.binaryproto"
19     mirror: true
20   }
21   data_param {
22     source: "/train_hat_v3.3.lmdb"
23     batch_size: 16 # 32
24     backend: LMDB
25   }
26 }
27 layer {
28   name: "data"
29   type: "Data"
30   top: "data"
31   top: "label"
32   include {
33     phase: TEST
34   }
35   transform_param {
36     # mean_file: "/data/mean.binaryproto"
37     #
38   }
39   data_param {
40     source: "/valid_hat_v3.3.lmdb"
41     batch_size: 16 # 16
42     backend: LMDB
43   }
44 }
45 layer {
46   bottom: "data"
47   top: "conv1"
48   type: "Convolution"
49   convolution_param {
50     num_output: 10
51     kernel_size: 3
52     pad: 1
53     stride: 1
54     weight_filler {
55       type: "msra"
56     }
57     bias_term: false
58   }
59 }
60 layer {
61   bottom: "conv1"
62   top: "conv1"
```

Right Panel (Your Net: pic): This panel shows a visual representation of the network structure. The diagram is titled "tiny_resnet" and illustrates the flow of data through the network layers. It starts with a "data" input, followed by a "conv1" layer. The network then branches into two parallel paths, each containing a "res2a_branch2" and a "scale2a_branch1" layer. These paths are then combined into a "res2a" layer. This is followed by another "res2b_branch2" and "scale2b_branch1" layer, which are combined into a "res2b" layer. Finally, the network branches into two parallel paths, each containing a "res3a_branch2" and a "scale3a_branch1" layer, which are combined into a "res3a" layer.

I. Training With Caffe

- Three Parts: Solver

 train_shrunked_tiny_resnet10_hat_solver_v3.3.prototxt

Solver

Your CNN hyperparameters

I. Training With Caffe

- Three Parts: Solver

```
1 net: "/week7.prototxt"
2
3 test_iter: 1000
4 test_interval: 500
5
6
7 base_lr: 0.01
8 momentum: 0.9
9 weight_decay: 0.0005
10 lr_policy: "multistep"
11 gamma: 0.2
12
13 max_iter: 100000
14 stepvalue: 30000
15 stepvalue: 60000
16
17 display: 100
18
19 snapshot: 2000
20 snapshot_prefix: "tiny_resnet_model/"
21
22 solver_mode: GPU
23
24
25
```

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

 `train_shrunked_tiny_resnet10_hat_v3.3.lmdb`

Data

Your data going to be trained / tested

I. Training With Caffe

- Three Parts: Data-Type

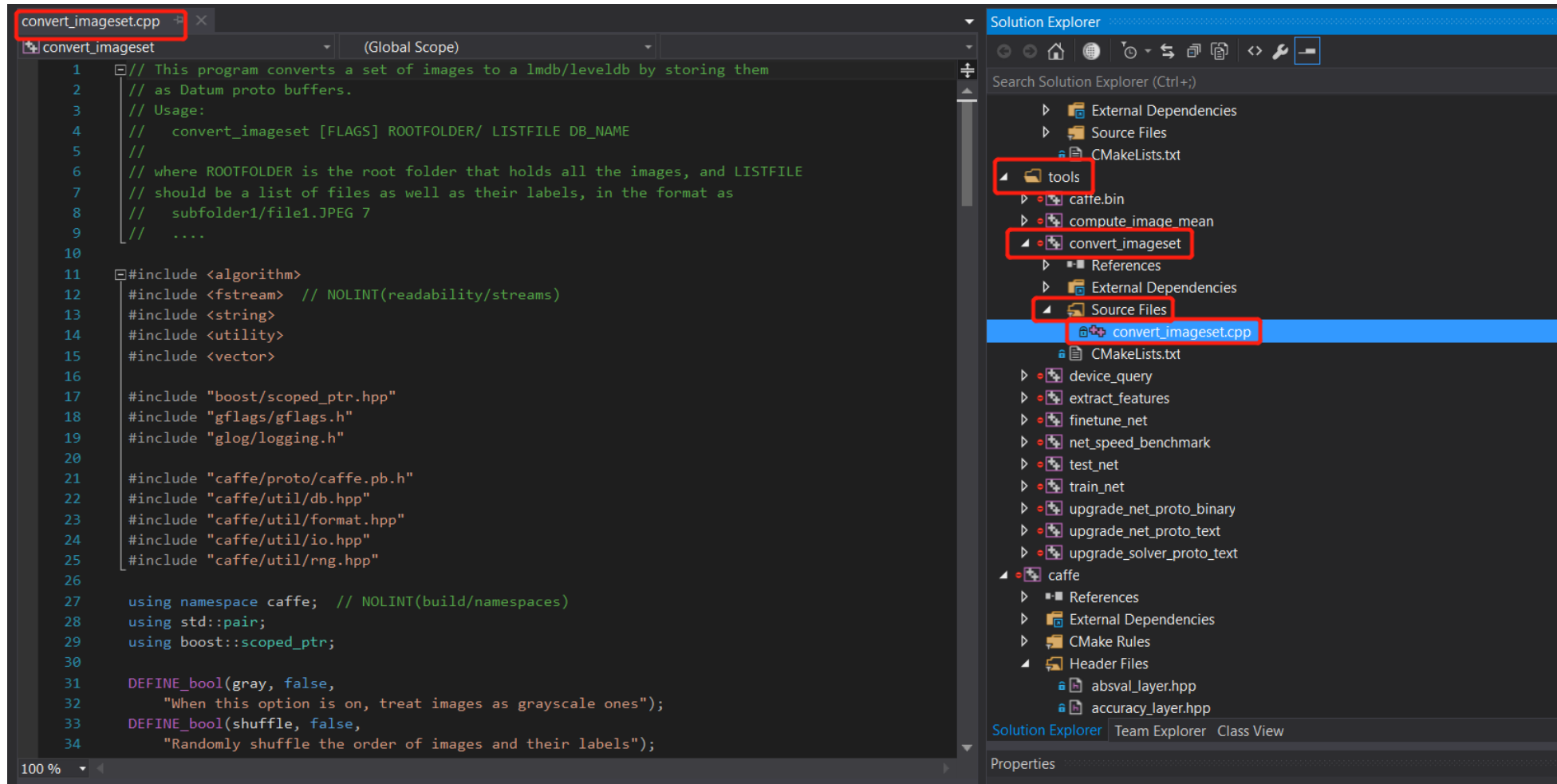
`.leveldb`

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

`.lmdb`

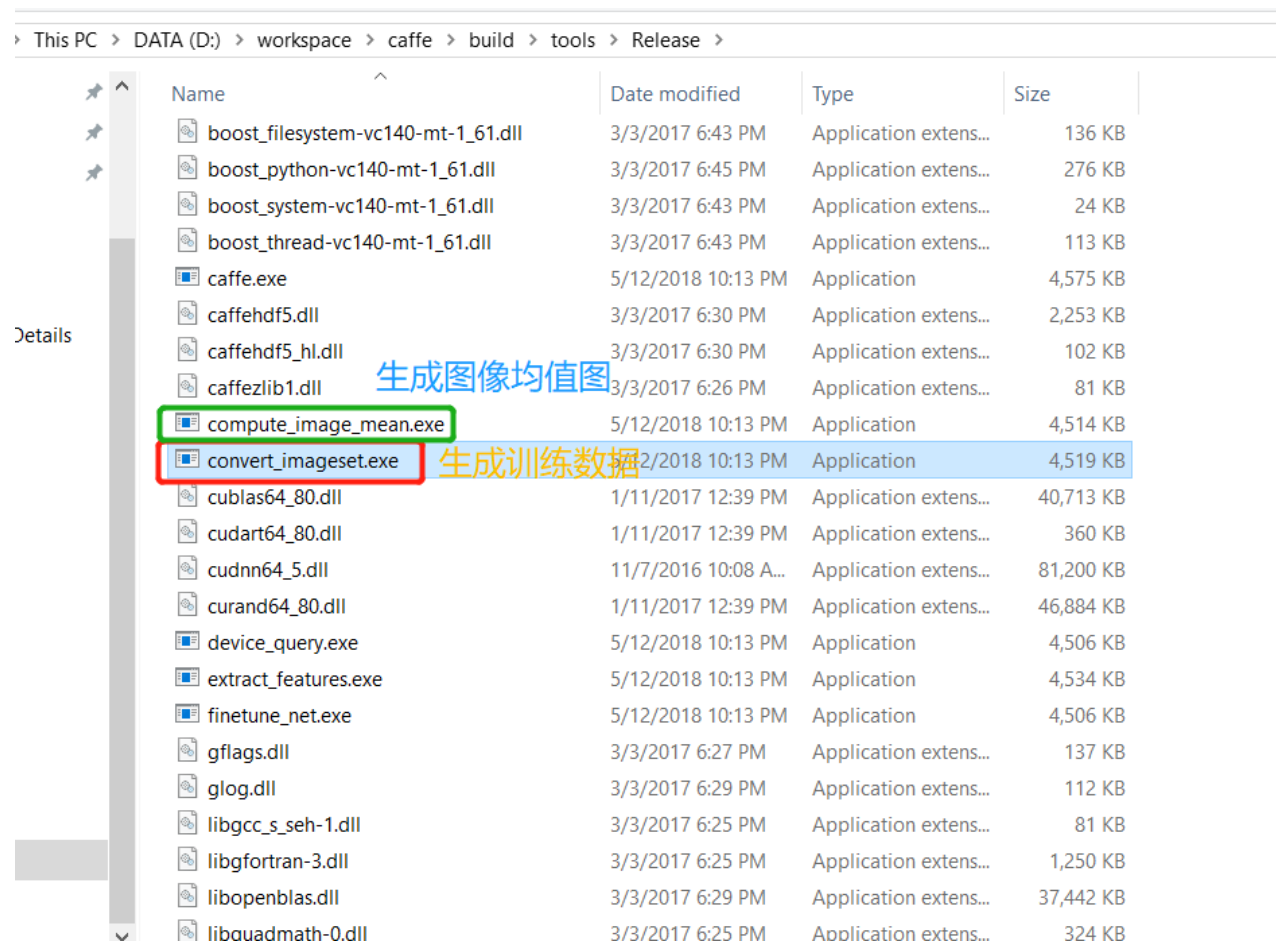
I. Training With Caffe

- Three Parts: Data-Generation



I. Training With Caffe






- Three Parts: Data-Generation(Windows)



This PC > DATA (D:) > workspace > caffe > build > tools > Release >				
Name	Date modified	Type	Size	
boost_filesystem-vc140-mt-1_61.dll	3/3/2017 6:43 PM	Application extens...	136 KB	
boost_python-vc140-mt-1_61.dll	3/3/2017 6:45 PM	Application extens...	276 KB	
boost_system-vc140-mt-1_61.dll	3/3/2017 6:43 PM	Application extens...	24 KB	
boost_thread-vc140-mt-1_61.dll	3/3/2017 6:43 PM	Application extens...	113 KB	
caffe.exe	5/12/2018 10:13 PM	Application	4,575 KB	
caffe_hdf5.dll	3/3/2017 6:30 PM	Application extens...	2,253 KB	
caffe_hdf5_hl.dll	3/3/2017 6:30 PM	Application extens...	102 KB	
caffe_zlib1.dll	3/3/2017 6:26 PM	Application extens...	81 KB	
compute_image_mean.exe	5/12/2018 10:13 PM	Application	4,514 KB	生成图像均值图
convert_imageset.exe	5/12/2018 10:13 PM	Application	4,519 KB	生成训练数据
cublas64_80.dll	1/11/2017 12:39 PM	Application extens...	40,713 KB	
cudart64_80.dll	1/11/2017 12:39 PM	Application extens...	360 KB	
cudnn64_5.dll	11/7/2016 10:08 A...	Application extens...	81,200 KB	
curand64_80.dll	1/11/2017 12:39 PM	Application extens...	46,884 KB	
device_query.exe	5/12/2018 10:13 PM	Application	4,506 KB	
extract_features.exe	5/12/2018 10:13 PM	Application	4,534 KB	
finetune_net.exe	5/12/2018 10:13 PM	Application	4,506 KB	
gflags.dll	3/3/2017 6:27 PM	Application extens...	137 KB	
glog.dll	3/3/2017 6:29 PM	Application extens...	112 KB	
libgcc_s_seh-1.dll	3/3/2017 6:25 PM	Application extens...	81 KB	
libgfortran-3.dll	3/3/2017 6:25 PM	Application extens...	1,250 KB	
libopenblas.dll	3/3/2017 6:29 PM	Application extens...	37,442 KB	
libquadmath-0.dll	3/3/2017 6:25 PM	Application extens...	324 KB	

I. Training With Caffe

- Three Parts: Data-Generation(Linux)

This PC > DATA (D:) > workspace > caffe > examples > imagenet				
	Name	Date modified	Type	Size
	 create_imagenet.sh	5/9/2018 8:57 AM	Shell Script	2 KB
	 make_imagenet_mean.sh	5/9/2018 8:57 AM	Shell Script	1 KB
	 readme.md	5/9/2018 8:57 AM	MD File	8 KB
	 resume_training.sh	5/9/2018 8:57 AM	Shell Script	1 KB
	 train_caffenet.sh	5/9/2018 8:57 AM	Shell Script	1 KB

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
1 ID=v3.3
2 ATTR=hat
3
4 CAFFE_ROOT=/home/workspace/caffe/build/tools
5 SRC_DATA_ROOT=/home/workspace/human_attribute/v3/${ATTR}/${ID}
6 DST_DATA_ROOT=/v3/lmdb/${ATTR}/${ID}
7
8 CREATE_DB=${CAFFE_ROOT}/convert_imageset
9 TRAIN_LIST=${SRC_DATA_ROOT}/train_${ATTR}_${ID}.txt
10 VALID_LIST=${SRC_DATA_ROOT}/valid_${ATTR}_${ID}.txt
11 TRAIN_DB=${DST_DATA_ROOT}/${ATTR}_${ID}_lmdb/train_${ATTR}_${ID}_lmdb
12 VALID_DB=${DST_DATA_ROOT}/${ATTR}_${ID}_lmdb/valid_${ATTR}_${ID}_lmdb
13
14 BACK_END=lmdb
15
16 RESIZE_WIDTH=56
17 RESIZE_HEIGHT=56
18 RESIZE_BORDER=56
19
20 echo "Create train lmdb..."
21 rm -rf ${TRAIN_DB}
22
23 ${CREATE_DB} --resize_width=${RESIZE_BORDER} --resize_height=${RESIZE_BORDER} --backend=${BACK_END} "" ${TRAIN_LIST} ${TRAIN_DB}
24 echo "Create valid lmdb..."
25 rm -rf ${VALID_DB}
26 ${CREATE_DB} --resize_width=${RESIZE_BORDER} --resize_height=${RESIZE_BORDER} --backend=${BACK_END} "" ${VALID_LIST} ${VALID_DB}
27 echo "All Done!"
```

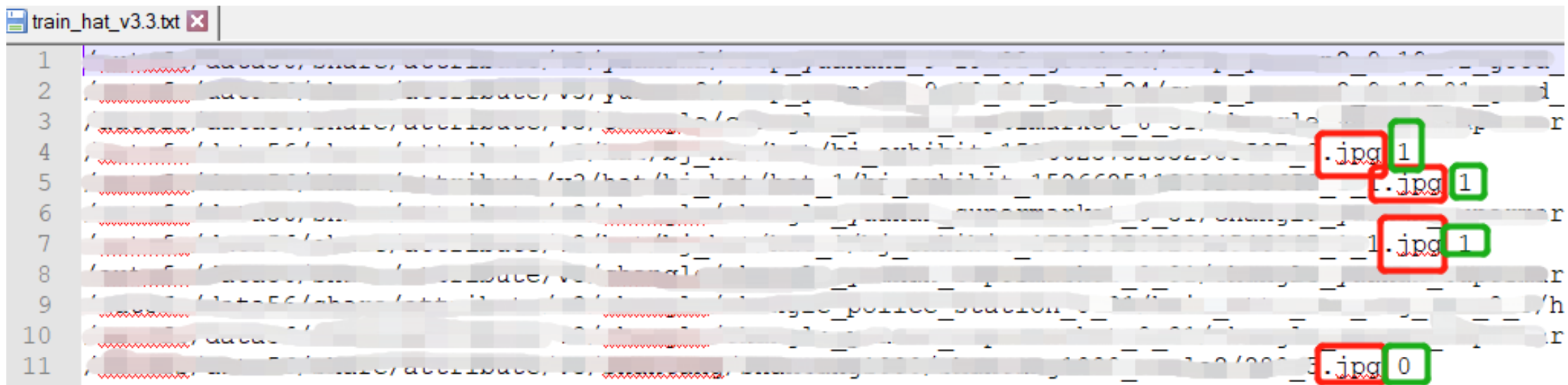
I. Training With Caffe

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

```
create_hat_db_v3.3.sh
1 ID=v3.3
2 ATTR=hat
3
4 CAFE_ROOT=/home/workspace/caffe/build/tools
5 SRC_DATA_ROOT=/home/workspace/human_attribute/v3/${ATTR}/${ID}
6 DST_DATA_ROOT=/v3/lmdb/${ATTR}/${ID}
7
8 CREATE_DB=${CAFE_ROOT}/convert_imageset
9 TRAIN_LIST=${SRC_DATA_ROOT}/train_${ATTR}_${ID}.txt
10 VALID_LIST=${SRC_DATA_ROOT}/valid_${ATTR}_${ID}.txt
11 TRAIN_DB=${DST_DATA_ROOT}/${ATTR}_${ID}_lmdb/train_${ATTR}_${ID}_lmdb
12 VALID_DB=${DST_DATA_ROOT}/${ATTR}_${ID}_lmdb/valid_${ATTR}_${ID}_lmdb
13
14 BACK_END=lmdb
15
16 RESIZE_WIDTH=56
17 RESIZE_HEIGHT=56
18 RESIZE_BORDER=56
19
20 echo "Create train lmdb..."
21 rm -rf ${TRAIN_DB}
22
23 ${CREATE_DB} --resize_width=${RESIZE_BORDER} --resize_height=${RESIZE_BORDER} --backend=${BACK_END} "" ${TRAIN_LIST} ${TRAIN_DB}
24 echo "Create valid lmdb..."
25 rm -rf ${VALID_DB}
26 ${CREATE_DB} --resize_width=${RESIZE_BORDER} --resize_height=${RESIZE_BORDER} --backend=${BACK_END} "" ${VALID_LIST} ${VALID_DB}
27 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: