关于框架

Caffe / Torch / TensorFlow / MxNet

Caffe

http://caffe.berkeleyvision.org

- 为啥安装那么贵?


专业深度学习调试
caffe安装 10元
CNN 5元/层
RNN 8元/层
专业
@南大周志华
weibo.com/zhouzh2012

- 依赖了大量的第三方库
  - 为了读取图像，以及简单的图像处理，链接很重的OpenCV库
  - boost来实现一些C++ 11的特征
  - HD5/LMDB/LEVELDB 用来做数据的IO
  - ProtoBuffer 使用随处可见

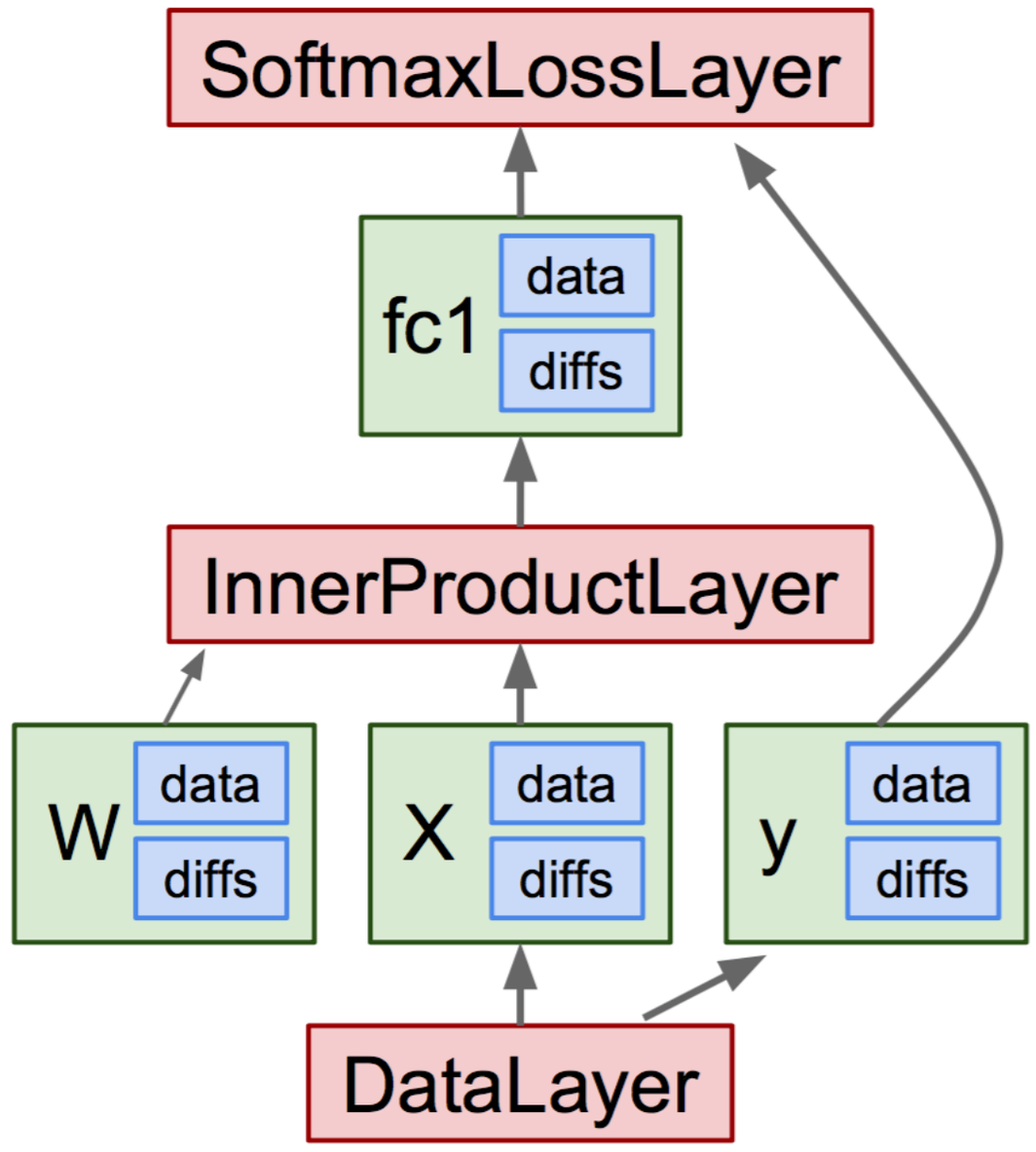
- 基于层的设计思路

  - **Blob** 模块：实现了Tensor的功能，保存数据和梯度值

  - **Layer** 模块：根据输入(bottom)blog计算输出(top)blob，同时保存权重 / 梯度

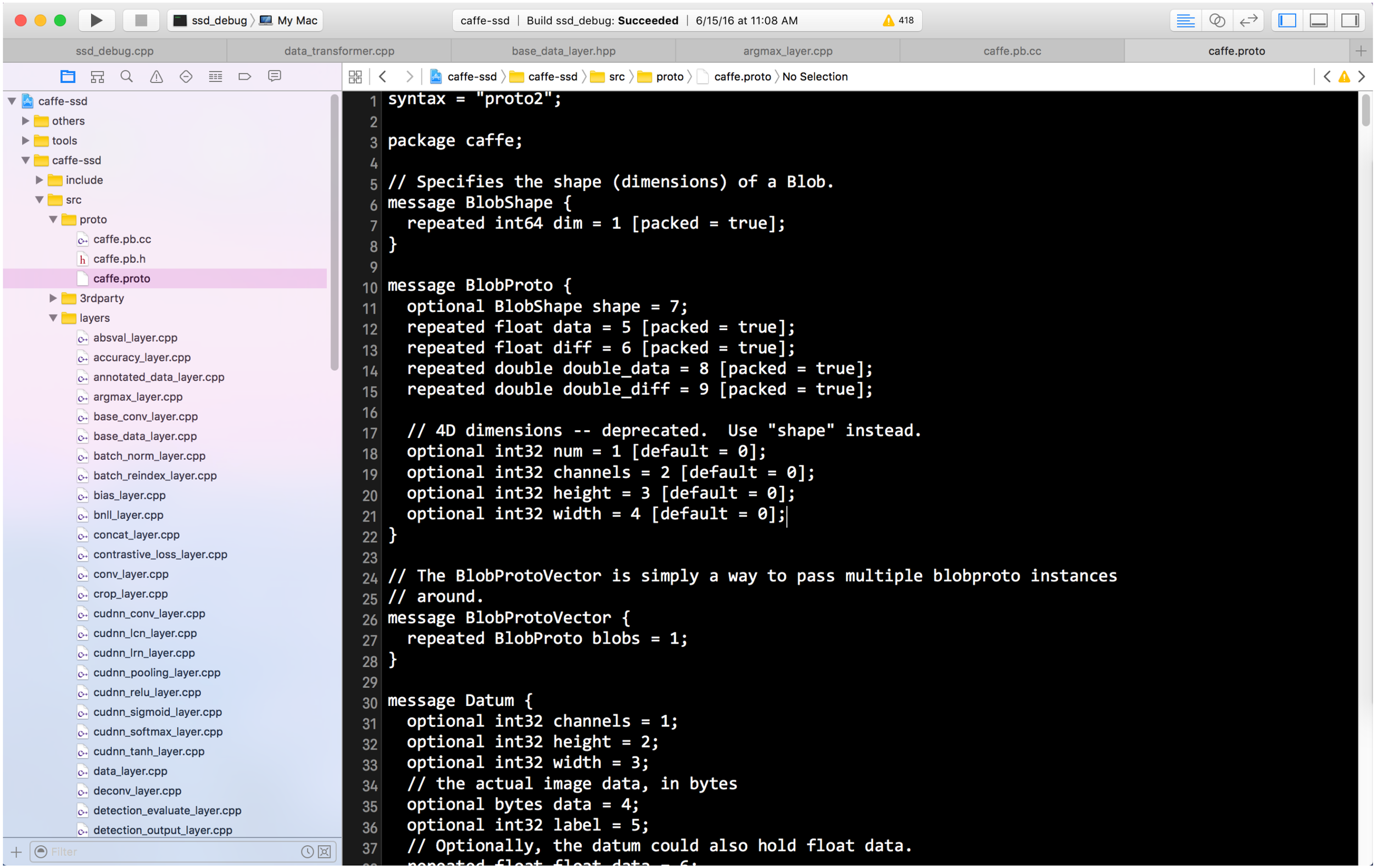
  - **Net** 模块：由多个layer组成，实现forward/backward计算

  - **Solver** 模块：最优化模块，利用梯度值更新权重



- Protocol Buffer 技术
  - 引入protocol buffer 技术，省去编写大量描述性的C++代码，比如配置参数、属性变量等等
  - 方便序列化，用户可以直接阅读prototxt文件，来了解网络结构

- .prototxt .caffemodel文件
  - prototxt 描述网络，通过Google的Protobuffer 编译器直接读取 / 序列化C++的对象
  - caffemodel 权重文件


ssd_debug
My Mac
caffe-ssd | Build ssd_debug: Succeeded | 6/15/16 at 11:08 AM
418
ssd_debug.cpp
data_transformer.cpp
base_data_layer.hpp
argmax_layer.cpp
caffe.pb.cc
caffe.proto
caffe-ssd
caffe-ssd
src
proto
caffe.proto
No Selection
caffe-ssd
others
tools
caffe-ssd
include
src
proto
caffe.pb.cc
caffe.pb.h
caffe.proto
3rdparty
layers
absval_layer.cpp
accuracy_layer.cpp
annotated_data_layer.cpp
argmax_layer.cpp
base_conv_layer.cpp
base_data_layer.cpp
batch_norm_layer.cpp
batch_reindex_layer.cpp
bias_layer.cpp
bnll_layer.cpp
concat_layer.cpp
contrastive_loss_layer.cpp
conv_layer.cpp
crop_layer.cpp
cudnn_conv_layer.cpp
cudnn_lcn_layer.cpp
cudnn_lrn_layer.cpp
cudnn_pooling_layer.cpp
cudnn_relu_layer.cpp
cudnn_sigmoid_layer.cpp
cudnn_softmax_layer.cpp
cudnn_tanh_layer.cpp
data_layer.cpp
deconv_layer.cpp
detection_evaluate_layer.cpp
detection_output_layer.cpp
Filter
1 syntax = "proto2";
2
3 package caffe;
4
5 // Specifies the shape (dimensions) of a Blob.
6 message BlobShape {
7 repeated int64 dim = 1 [packed = true];
8 }
9
10 message BlobProto {
11 optional BlobShape shape = 7;
12 repeated float data = 5 [packed = true];
13 repeated float diff = 6 [packed = true];
14 repeated double double_data = 8 [packed = true];
15 repeated double double_diff = 9 [packed = true];
16
17 // 4D dimensions -- deprecated. Use "shape" instead.
18 optional int32 num = 1 [default = 0];
19 optional int32 channels = 2 [default = 0];
20 optional int32 height = 3 [default = 0];
21 optional int32 width = 4 [default = 0];
22 }
23
24 // The BlobProtoVector is simply a way to pass multiple blobproto instances
25 // around.
26 message BlobProtoVector {
27 repeated BlobProto blobs = 1;
28 }
29
30 message Datum {
31 optional int32 channels = 1;
32 optional int32 height = 2;
33 optional int32 width = 3;
34 // the actual image data, in bytes
35 optional bytes data = 4;
36 optional int32 label = 5;
37 // Optionally, the datum could also hold float data.

- Caffe 的训练
  - 需要提前做数据准备，保存为LMDB/LevelDB等格式
  - 不用编写C++代码，直接编写 .prototxt 定义 net 对象
  - 直接编写.prototxt 定义 solver 对象
  - 通过参数直接执行 caffe 命令进行训练

- 第一步：数据准备
  - 准备数据文件：
    - xx/xx/xx.jpg　label　##每行都是这样的格式
  - 通过tools/create_xxxx 命令生产 LMDB/LevelDB等格式的数据文件

```
mywork:dummy teaonly$ ls
create_db.sh        dummy_test_size.txt images              test_db
dummy_test.list     dummy_train.list    infos               train_db
mywork:dummy teaonly$
mywork:dummy teaonly$
mywork:dummy teaonly$ cat dummy_train.list
./images/train.jpg ./infos/train.json
mywork:dummy teaonly$ ls train_db/
data.mdb lock.mdb
mywork:dummy teaonly$ cat dummy_test.list
./images/test.jpg ./infos/test.json
mywork:dummy teaonly$ ls test_db/
data.mdb lock.mdb
mywork:dummy teaonly$
```

```
rm -rf ./data/dummy/train_db
./build/tools/convert_annoset --min_dim=0 --max_dim=0 --resize_height=0 --resize_width=0 --backend=lmdb --shuffle=False --check_size=Fal
se --encode_type=jpg --encoded=True --gray=False \
        --anno_type=detection \
        --label_type=json \
        --label_map_file=./data/traffic/labelmap_traffic.prototxt \
        /Users/teaonly/Workspace/traffic/caffe-ssd/data/dummy/ \
        ./data/dummy/dummy_train.list \
        ./data/dummy/train_db
```

```
75 template <typename Dtype>
76 void BasePrefetchingDataLayer<Dtype>::InternalThreadEntry() {
77 #ifndef CPU_ONLY
78   cudaStream_t stream;
79   if (Caffe::mode() == Caffe::GPU) {
80     CUDA_CHECK(cudaStreamCreateWithFlags(&stream, cudaStreamNonBlocking));
81   }
82 #endif
83
84   try {
85     while (!must_stop()) {
86       Batch<Dtype>* batch = prefetch_free_.pop();
87       load_batch
88 #ifndef CPU_ONL
89       if (Caffe
90         batch->                                    sh(stream);
91         CUDA_CH                                    ));
92       }
93 #endif
94       prefetch_
95     }
96   } catch (boos
97     // Interrup                                    tdown
98   }
99 #ifndef CPU_ONL
100   if (Caffe::mo
101     CUDA_CHECK(
102   }
103 #endif
104 }
```

- AnnotatedDataLayer::load_batch()
- *BasePrefetchingDataLayer::load_batch()*
- caffe::load_batch() (video_data_layer.cpp:72)
- load_batch (video_data_layer.hpp:43)
- caffe::load_batch() (window_data_layer.cpp:228)
- load_batch (window_data_layer.hpp:38)
- load_batch (annotated_data_layer.hpp:32)
- caffe::load_batch() (data_layer.cpp:56)
- load_batch (data_layer.hpp:32)
- *load_batch (base_data_layer.hpp:75)*
- caffe::load_batch() (image_data_layer.cpp:100)
- load_batch (image_data_layer.hpp:38)

Search for Definitions

Show in Symbol Navigator

- 第二步：编写网络文件
  - 规则简单的，可以直接用文本编辑器生成
  - 网络结构复杂的，利用Python脚本生成
  - 注意net prototxt文件描述的是有向无环图

```
name: "LogisticRegressionNet"
layers {
  top: "data"
  top: "label"
  name: "data"
  type: HDF5_DATA
  hdf5_data_param {
    source: "examples/hdf5_classification/data/train.txt"
    batch_size: 10
  }
  include {
    phase: TRAIN
  }
}
layers {
  bottom: "data"
  top: "fc1"
  name: "fc1"
  type: INNER_PRODUCT
  blobs_lr: 1
  blobs_lr: 2
  weight_decay: 1
  weight_decay: 0
  inner_product_param {
    num_output: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layers {
  bottom: "fc1"
  bottom: "label"
  top: "loss"
  name: "loss"
  type: SOFTMAX_LOSS
}
```

```
name: "LogisticRegressionNet"
layers {
  top: "data"
  top: "label"
  name: "data"
  type: HDF5_DATA
  hdf5_data_param {
    source: "examples/hdf5_classification/data/train.txt"
    batch_size: 10
  }
  include {
    phase: TRAIN
  }
}
layers {
  bottom: "data"
  top: "fc1"
  name: "fc1"
  type: INNER_PRODUCT
  blobs_lr: 1
  blobs_lr: 2
  weight_decay: 1
  weight_decay: 0
```

Layers and Blobs often have same name!

Set these to 0 to freeze a layer

Learning rates (weight + bias)

Regularization (weight + bias)

Number of output classes

```
inner_product_param {
  num_output: 2
  weight_filler {
    type: "gaussian"
    std: 0.01
  }
  bias_filler {
    type: "constant"
    value: 0
  }
}
layers {
  bottom: "fc1"
  bottom: "label"
  top: "loss"
  name: "loss"
  type: SOFTMAX_LOSS
}
```

- 复杂的网络
  - 网络结构复杂：比如残差网络，手写非常容易出错
  - 层数较多的，手写非常低效
  - 网络结构无法复用，prototxt不具备可编程性
- 采用Python来编写网络文件

创建net对象

配置net对象

```
385 # Create train net.
386 net = caffe.NetSpec()
387 net.data, net.label = CreateAnnotatedDataLayer(train_data, batch_size=batch_size_per_device,
388         train=True, output_label=True, label_map_file=label_map_file,
389         transform_param=train_transform_param, batch_sampler=batch_sampler)
390
391 VGGNetBody(net, from_layer='data', fully_conv=True, reduced=True, dilated=True,
392     dropout=False, freeze_layers=freeze_layers)
393
394 AddExtraLayers(net, use_batchnorm)
395
396 mbox_layers = CreateMultiBoxHead(net, data_layer='data', from_layers=mbox_source_layers,
397         use_batchnorm=use_batchnorm, min_sizes=min_sizes, max_sizes=max_sizes,
398         aspect_ratios=aspect_ratios, normalizations=normalizations,
399         num_classes=num_classes, share_location=share_location, flip=flip, clip=clip,
400         prior_variance=prior_variance, kernel_size=3, pad=1)
401
402 # Create the MultiBoxLossLayer.
403 name = "mbox_loss"
404 mbox_layers.append(net.label)
405 net[name] = L.MultiBoxLoss(*mbox_layers, multibox_loss_param=multibox_loss_param,
406         loss_param=loss_param, include=dict(phase=caffe_pb2.Phase.Value('TRAIN')),
407         propagate_down=[True, True, False, False])
408
409 with open(train_net_file, 'w') as f:
410     print('name: "{}_train"'.format(model_name), file=f)
411     print(net.to_proto(), file=f)
```

输出.prototxt文件

- 丰富的C++实现Layer

```
teaonly@deeper:~/workspace/caffe-ssd-teaonly/src/caffe/layers$ ls *.cpp
absval_layer.cpp            cudnn_sigmoid_layer.cpp       im2col_layer.cpp                  reduction_layer.cpp
accuracy_layer.cpp          cudnn_softmax_layer.cpp       image_data_layer.cpp              relu_layer.cpp
annotated_data_layer.cpp    cudnn_tanh_layer.cpp          infogain_loss_layer.cpp           reshape_layer.cpp
argmax_layer.cpp            data_layer.cpp                inner_product_layer.cpp           scale_layer.cpp
base_conv_layer.cpp         deconv_layer.cpp              input_layer.cpp                   sigmoid_cross_entropy_loss_layer.cpp
base_data_layer.cpp         detection_evaluate_layer.cpp  log_layer.cpp                     sigmoid_layer.cpp
batch_norm_layer.cpp        detection_output_layer.cpp    loss_layer.cpp                    silence_layer.cpp
batch_reindex_layer.cpp     dropout_layer.cpp             lrn_layer.cpp                     slice_layer.cpp
bias_layer.cpp              dummy_data_layer.cpp          memory_data_layer.cpp             smooth_L1_loss_layer.cpp
bnll_layer.cpp              eltwise_layer.cpp             multibox_loss_layer.cpp           softmax_layer.cpp
concat_layer.cpp            elu_layer.cpp                 multinomial_logistic_loss_layer.cpp  softmax_loss_layer.cpp
contrastive_loss_layer.cpp  embed_layer.cpp               mvn_layer.cpp                     split_layer.cpp
conv_layer.cpp              euclidean_loss_layer.cpp      neuron_layer.cpp                  spp_layer.cpp
crop_layer.cpp              exp_layer.cpp                 normalize_layer.cpp               tanh_layer.cpp
cudnn_conv_layer.cpp        filter_layer.cpp              permute_layer.cpp                 threshold_layer.cpp
cudnn_lcn_layer.cpp         flatten_layer.cpp             pooling_layer.cpp                 tile_layer.cpp
cudnn_lrn_layer.cpp         hdf5_data_layer.cpp           power_layer.cpp                   video_data_layer.cpp
cudnn_pooling_layer.cpp     hdf5_output_layer.cpp         prelu_layer.cpp                   window_data_layer.cpp
cudnn_relu_layer.cpp        hinge_loss_layer.cpp          prior_box_layer.cpp
```

- 第三步：定义solver文件
  - 简单手动编写
  - 设置好需要的参数

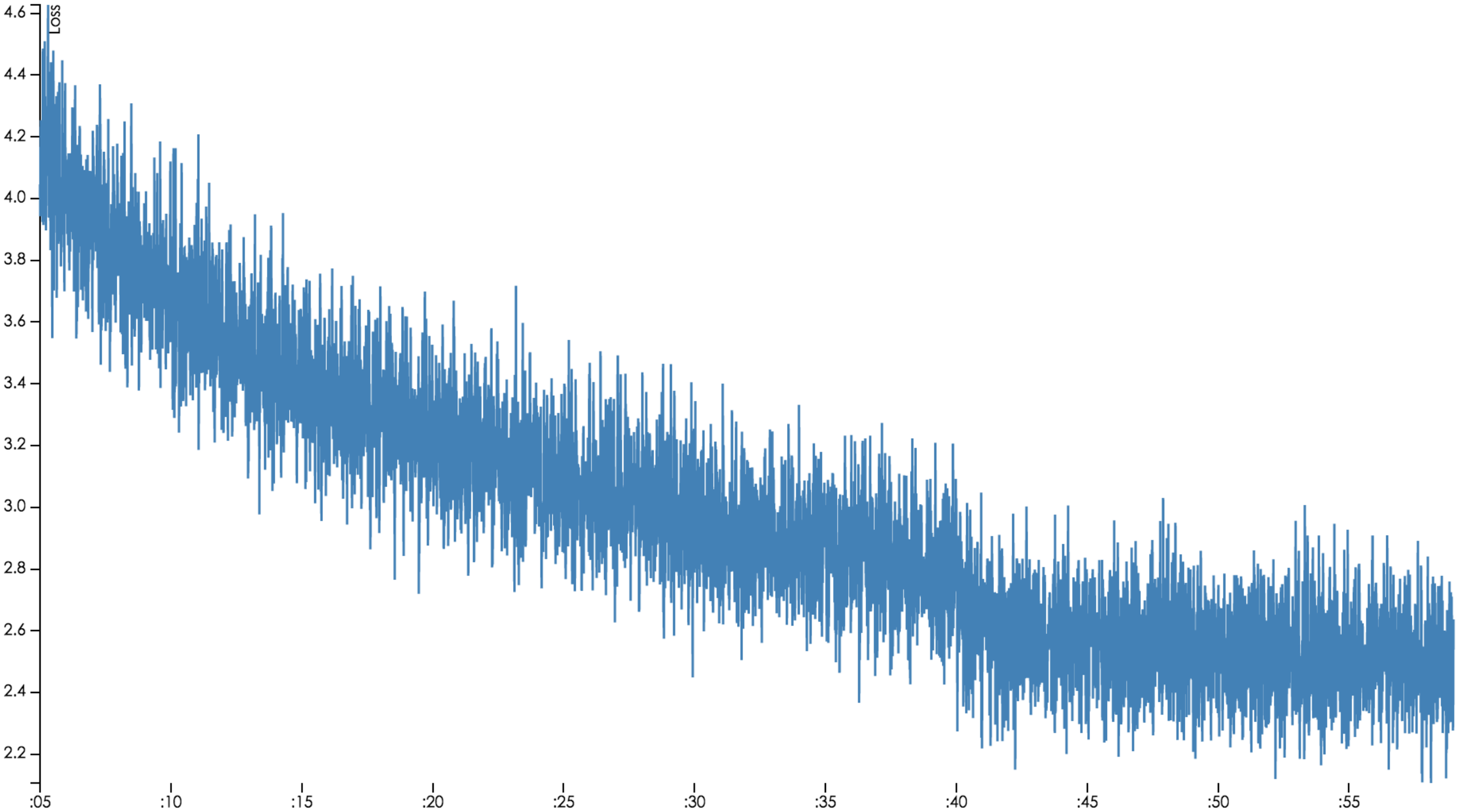
```
net: "models/bvlc_alexnet/train_val.prototxt"
test_iter: 1000
test_interval: 1000
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 100000
display: 20
max_iter: 450000
momentum: 0.9
weight_decay: 0.0005
snapshot: 10000
snapshot_prefix: "models/bvlc_alexnet/caffe_alexnet_train"
solver_mode: GPU
```

- 第四步：训练

```
./build/tools/caffe train \
-gpu 0 \
-model path/to/trainval.prototxt \
-solver path/to/solver.prototxt \
-weights path/to/pretrained_weights.caffemodel
```

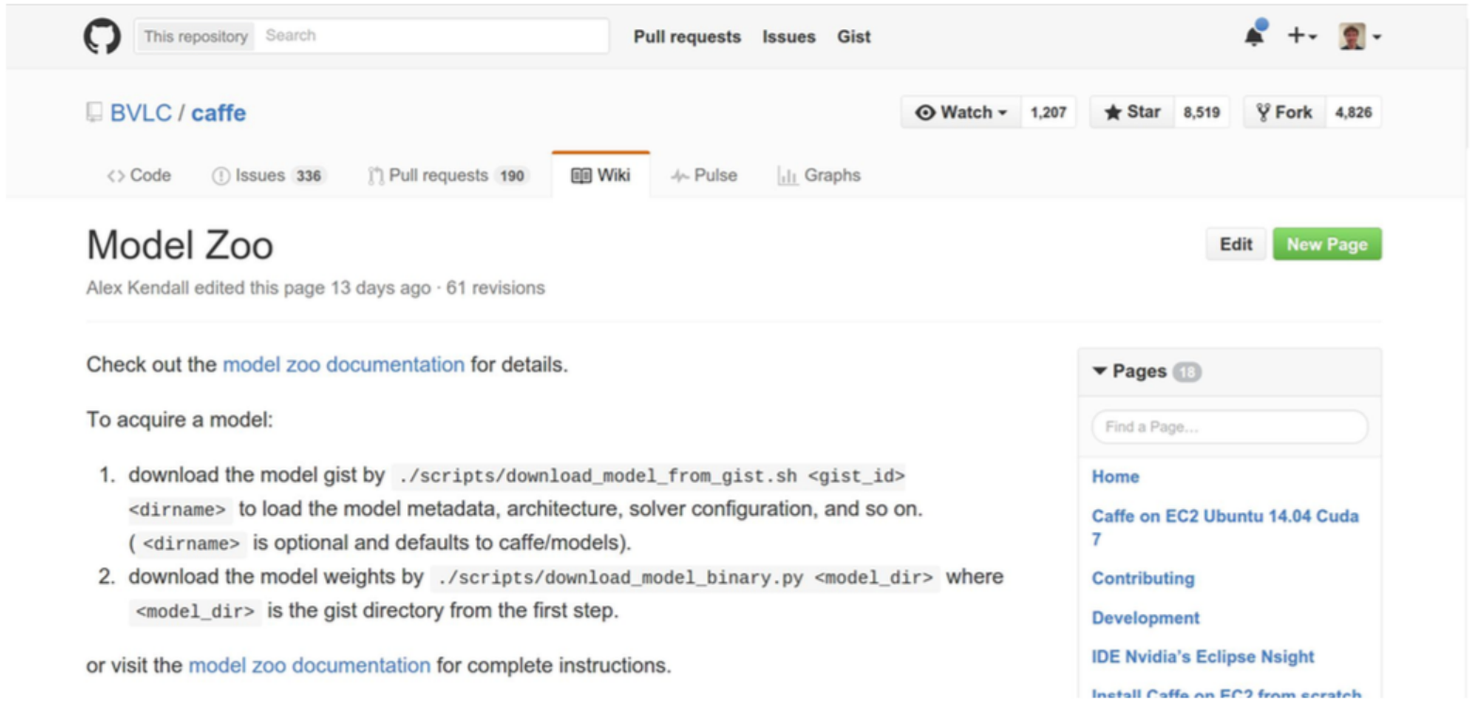
```
I0618 17:38:48.692409 18454 solver.cpp:247]     Train net output #0: mbox_loss = 2.32637 (* 1 = 2.32637 loss)
I0618 17:38:49.887291 18454 sgd_solver.cpp:106] Iteration 74250, lr = 4e-05
I0618 17:39:45.494685 18454 solver.cpp:231] Iteration 74260, loss = 2.47432
I0618 17:39:45.495261 18454 solver.cpp:247]     Train net output #0: mbox_loss = 2.62012 (* 1 = 2.62012 loss)
I0618 17:39:45.495306 18454 sgd_solver.cpp:106] Iteration 74260, lr = 4e-05
I0618 17:40:41.864326 18454 solver.cpp:231] Iteration 74270, loss = 2.28506
I0618 17:40:41.864955 18454 solver.cpp:247]     Train net output #0: mbox_loss = 1.56052 (* 1 = 1.56052 loss)
I0618 17:40:43.048810 18454 sgd_solver.cpp:106] Iteration 74270, lr = 4e-05
I0618 17:41:37.417719 18454 solver.cpp:231] Iteration 74280, loss = 2.38529
I0618 17:41:37.418252 18454 solver.cpp:247]     Train net output #0: mbox_loss = 2.38979 (* 1 = 2.38979 loss)
I0618 17:41:37.418298 18454 sgd_solver.cpp:106] Iteration 74280, lr = 4e-05
I0618 17:42:34.138180 18454 solver.cpp:231] Iteration 74290, loss = 2.54569
I0618 17:42:34.138672 18454 solver.cpp:247]     Train net output #0: mbox_loss = 2.57161 (* 1 = 2.57161 loss)
I0618 17:42:35.282429 18454 sgd_solver.cpp:106] Iteration 74290, lr = 4e-05
I0618 17:43:30.756151 18454 solver.cpp:231] Iteration 74300, loss = 2.61987
I0618 17:43:30.756969 18454 solver.cpp:247]     Train net output #0: mbox_loss = 2.48623 (* 1 = 2.48623 loss)
I0618 17:43:30.757016 18454 sgd_solver.cpp:106] Iteration 74300, lr = 4e-05
I0618 17:44:29.362432 18454 solver.cpp:231] Iteration 74310, loss = 2.46113
I0618 17:44:29.362619 18454 solver.cpp:247]     Train net output #0: mbox_loss = 1.4801 (* 1 = 1.4801 loss)
I0618 17:44:30.625130 18454 sgd_solver.cpp:106] Iteration 74310, lr = 4e-05
I0618 17:45:28.054224 18454 solver.cpp:231] Iteration 74320, loss = 2.73291
I0618 17:45:28.054728 18454 solver.cpp:247]     Train net output #0: mbox_loss = 2.9215 (* 1 = 2.9215 loss)
I0618 17:45:29.267699 18454 sgd_solver.cpp:106] Iteration 74320, lr = 4e-05
I0618 17:46:24.492121 18454 solver.cpp:231] Iteration 74330, loss = 2.23412
I0618 17:46:24.492725 18454 solver.cpp:247]     Train net output #0: mbox_loss = 1.53063 (* 1 = 1.53063 loss)
I0618 17:46:25.684329 18454 sgd_solver.cpp:106] Iteration 74330, lr = 4e-05
I0618 17:47:22.291318 18454 solver.cpp:231] Iteration 74340, loss = 2.35016
I0618 17:47:22.291916 18454 solver.cpp:247]     Train net output #0: mbox_loss = 2.57094 (* 1 = 2.57094 loss)
I0618 17:47:22.291965 18454 sgd_solver.cpp:106] Iteration 74340, lr = 4e-05
I0618 17:48:20.975242 18454 solver.cpp:231] Iteration 74350, loss = 2.40527
I0618 17:48:20.975785 18454 solver.cpp:247]     Train net output #0: mbox_loss = 1.32838 (* 1 = 1.32838 loss)
```

l Loss 函数分析


Loss
4.6
4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
2.4
2.2
:05
:10
:15
:20
:25
:30
:35
:40
:45
:50
:55

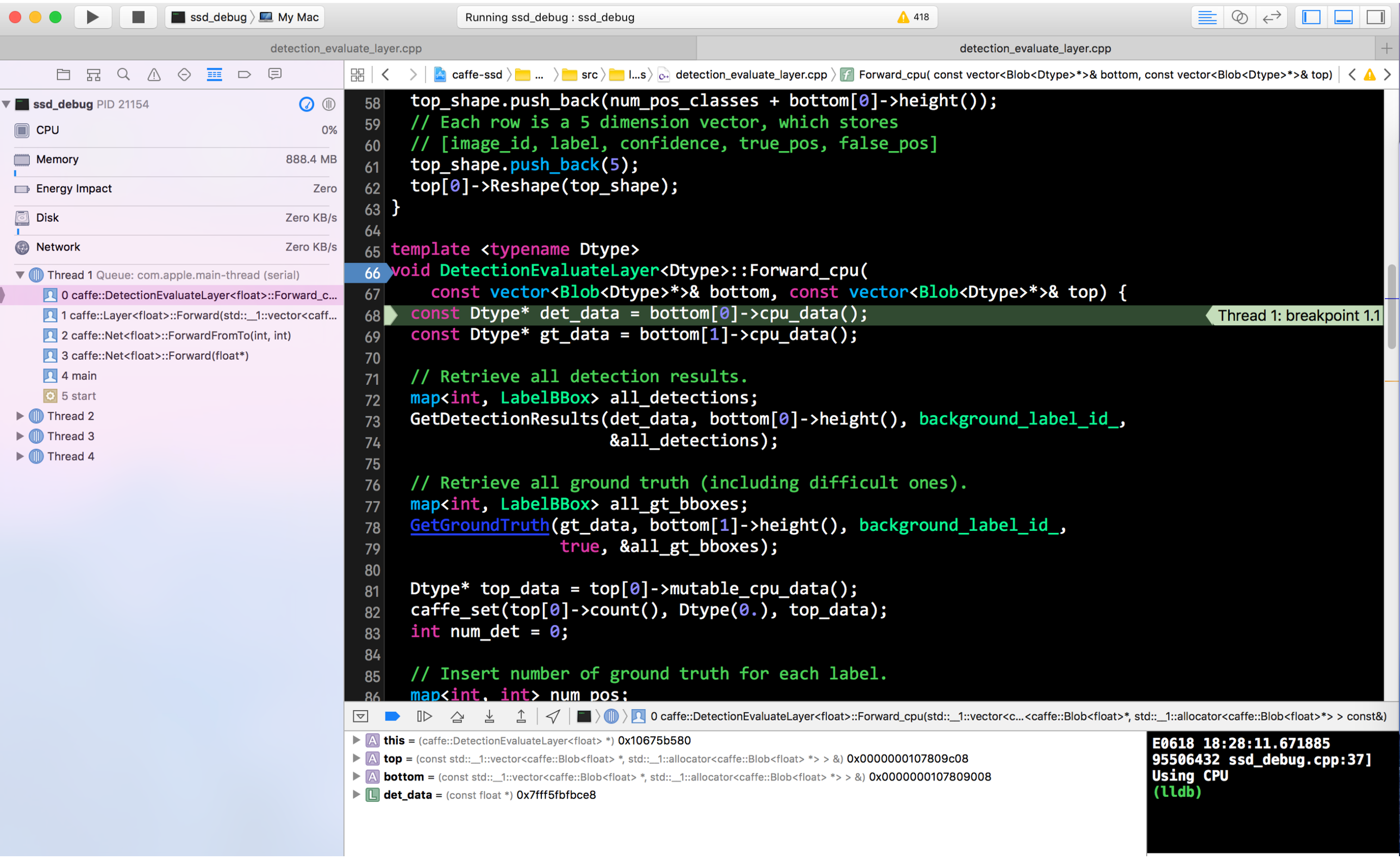
Caffe: Model Zoo

AlexNet, VGG,
GoogLeNet, ResNet,
plus others



https://github.com/BVLC/caffe/wiki/Model-Z

- 代码阅读


ssd_debug 〉 My Mac
Running ssd_debug : ssd_debug
418
detection_evaluate_layer.cpp
detection_evaluate_layer.cpp
caffe-ssd 〉 ... 〉 src 〉 l...s 〉 detection_evaluate_layer.cpp 〉 Forward_cpu(const vector<Blob<Dtype>*>& bottom, const vector<Blob<Dtype>*>& top)
ssd_debug PID 21154
CPU 0%
Memory 888.4 MB
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s
Thread 1 Queue: com.apple.main-thread (serial)
0 caffe::DetectionEvaluateLayer<float>::Forward_c...
1 caffe::Layer<float>::Forward(std::__1::vector<caff...
2 caffe::Net<float>::ForwardFromTo(int, int)
3 caffe::Net<float>::Forward(float*)
4 main
5 start
Thread 2
Thread 3
Thread 4
58 top_shape.push_back(num_pos_classes + bottom[0]->height());
59 // Each row is a 5 dimension vector, which stores
60 // [image_id, label, confidence, true_pos, false_pos]
61 top_shape.push_back(5);
62 top[0]->Reshape(top_shape);
63 }
64
65 template <typename Dtype>
66 void DetectionEvaluateLayer<Dtype>::Forward_cpu(
67 const vector<Blob<Dtype>*>& bottom, const vector<Blob<Dtype>*>& top) {
68 const Dtype* det_data = bottom[0]->cpu_data();
Thread 1: breakpoint 1.1
69 const Dtype* gt_data = bottom[1]->cpu_data();
70
71 // Retrieve all detection results.
72 map<int, LabelBBox> all_detections;
73 GetDetectionResults(det_data, bottom[0]->height(), background_label_id_,
74 &all_detections);
75
76 // Retrieve all ground truth (including difficult ones).
77 map<int, LabelBBox> all_gt_bboxes;
78 GetGroundTruth(gt_data, bottom[1]->height(), background_label_id_,
79 true, &all_gt_bboxes);
80
81 Dtype* top_data = top[0]->mutable_cpu_data();
82 caffe_set(top[0]->count(), Dtype(0.), top_data);
83 int num_det = 0;
84
85 // Insert number of ground truth for each label.
86 map<int, int> num_pos;
0 caffe::DetectionEvaluateLayer<float>::Forward_cpu(std::__1::vector<c...<caffe::Blob<float>*, std::__1::allocator<caffe::Blob<float>*> > const&)
this = (caffe::DetectionEvaluateLayer<float> *) 0x10675b580
top = (const std::__1::vector<caffe::Blob<float> *, std::__1::allocator<caffe::Blob<float> *> > &) 0x0000000107809c08
bottom = (const std::__1::vector<caffe::Blob<float> *, std::__1::allocator<caffe::Blob<float> *> > &) 0x0000000107809008
det_data = (const float *) 0x7fff5fbfbce8
E0618 18:28:11.671885 95506432 ssd_debug.cpp:37] Using CPU
(lldb)

- 生产环境的Caffe
  - 依赖库裁剪
  - 性能优化

```
$ldd ./build/lib/libcaffe.so.1.0.0-rc3
        linux-vdso.so.1 =>  (0x00007fff45fa0000)
        libcudart.so.6.5 => /usr/local/cuda/lib64/libcudart.so.6.5 (0x00007f4ee4dff000)
        libcublas.so.6.5 => /usr/local/cuda/lib64/libcublas.so.6.5 (0x00007f4ee335c000)
        libcurand.so.6.5 => /usr/local/cuda/lib64/libcurand.so.6.5 (0x00007f4edfa0f000)
        libglog.so.0 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/glog_lib/lib/libglog.so.0 (0x00007f4edf7ca000)
        libprotobuf.so.7 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/protobuf_lib/lib/libprotobuf.so.7 (0x00007f4edf4e7000)
        libboost_system.so.1.58.0 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/boost_lib/lib/libboost_system.so.1.58.0 (0x00007f4edf2e4000)
        libboost_filesystem.so.1.58.0 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/boost_lib/lib/libboost_filesystem.so.1.58.0 (0x00007f4edf0ce000)
        libboost_regex.so.1.58.0 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/boost_lib/lib/libboost_regex.so.1.58.0 (0x00007f4edede1000)
        libhdf5_hl.so.10 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/anaconda_lib/lib/libhdf5_hl.so.10 (0x00007f4edebc2000)
        libhdf5.so.10 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/anaconda_lib/lib/libhdf5.so.10 (0x00007f4ede6e5000)
        libleveldb.so.1 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/leveldb_lib/lib/libleveldb.so.1 (0x00007f4ede494000)
        libsnappy.so.1 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/snappy_lib/lib/libsnappy.so.1 (0x00007f4ede28e000)
        liblmdb.so => /gruntdata/app_data/lanbo.llb/deps_lib_shared/lmdb_lib/lib/liblmdb.so (0x00007f4ede07d000)
        libopencv_core.so.2.4 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/opencv_lib/lib/libopencv_core.so.2.4 (0x00007f4edd9e2000)
        libopencv_highgui.so.2.4 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/opencv_lib/lib/libopencv_highgui.so.2.4 (0x00007f4edd5b7000)
        libopencv_imgproc.so.2.4 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/opencv_lib/lib/libopencv_imgproc.so.2.4 (0x00007f4edd105000)
        libboost_thread.so.1.58.0 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/boost_lib/lib/libboost_thread.so.1.58.0 (0x00007f4edcee2000)
        libstdc++.so.6 => /usr/lib64/libstdc++.so.6 (0x00007f4edcbdc000)
        libopenblas.so.0 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/OpenBLAS_lib/lib/libopenblas.so.0 (0x00007f4edbd15000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f4edba90000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f4edb87a000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f4edb65d000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f4edb2ca000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f4edb0c6000)
        librt.so.1 => /lib64/librt.so.1 (0x00007f4edaebe000)
        libunwind.so.7 => not found
        libz.so.1 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/anaconda_lib/lib/libz.so.1 (0x00007f4edaca7000)
        libnppc.so.6.5 => /usr/local/cuda/lib64/libnppc.so.6.5 (0x00007f4edaa4a000)
        libnppi.so.6.5 => /usr/local/cuda/lib64/libnppi.so.6.5 (0x00007f4ed677f000)
        libnpps.so.6.5 => /usr/local/cuda/lib64/libnpps.so.6.5 (0x00007f4ed5f32000)
        libcufft.so.6.5 => /usr/local/cuda/lib64/libcufft.so.6.5 (0x00007f4ed350d000)
        libpng12.so.0 => /usr/lib64/libpng12.so.0 (0x00007f4ed32e7000)
        libgtk-x11-2.0.so.0 => /usr/lib64/libgtk-x11-2.0.so.0 (0x00007f4ed2c82000)
        libgdk-x11-2.0.so.0 => /usr/lib64/libgdk-x11-2.0.so.0 (0x00007f4ed29c6000)
        libatk-1.0.so.0 => /usr/lib64/libatk-1.0.so.0 (0x00007f4ed27a6000)
        libgio-2.0.so.0 => /lib64/libgio-2.0.so.0 (0x00007f4ed24fb000)
        libpangoft2-1.0.so.0 => /usr/lib64/libpangoft2-1.0.so.0 (0x00007f4ed22d0000)
        libgdk_pixbuf-2.0.so.0 => /usr/lib64/libgdk_pixbuf-2.0.so.0 (0x00007f4ed20b1000)
        libpangocairo-1.0.so.0 => /usr/lib64/libpangocairo-1.0.so.0 (0x00007f4ed1ea5000)
        libcairo.so.2 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/anaconda_lib/lib/libcairo.so.2 (0x00007f4ed1b64000)
        libpango-1.0.so.0 => /usr/lib64/libpango-1.0.so.0 (0x00007f4ed1919000)
        libfreetype.so.6 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/anaconda_lib/lib/libfreetype.so.6 (0x00007f4ed1686000)
        libfontconfig.so.1 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/anaconda_lib/lib/libfontconfig.so.1 (0x00007f4ed1448000)
        libgobject-2.0.so.0 => /lib64/libgobject-2.0.so.0 (0x00007f4ed1203000)
        libgmodule-2.0.so.0 => /lib64/libgmodule-2.0.so.0 (0x00007f4ed1000000)
        libglib-2.0.so.0 => /lib64/libglib-2.0.so.0 (0x00007f4ed0d1a000)
        libgthread-2.0.so.0 => /lib64/libgthread-2.0.so.0 (0x00007f4ed0b16000)
        libgstbase-0.10.so.0 => /usr/lib64/libgstbase-0.10.so.0 (0x00007f4ed08d8000)
        libgstreamer-0.10.so.0 => /usr/lib64/libgstreamer-0.10.so.0 (0x00007f4ed05f4000)
        libxml2.so.2 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/anaconda_lib/lib/libxml2.so.2 (0x00007f4ed029a000)
        /lib64/ld-linux-x86-64.so.2 (0x000000377d000000)
        libgfortran.so.3 => /usr/lib64/libgfortran.so.3 (0x00007f4ecffa7000)
        libX11.so.6 => /usr/lib64/libX11.so.6 (0x00007f4ecfc68000)
        libXfixes.so.3 => /usr/lib64/libXfixes.so.3 (0x00007f4ecfa62000)
        libXext.so.6 => /usr/lib64/libXext.so.6 (0x00007f4ecf850000)
        libXrender.so.1 => /usr/lib64/libXrender.so.1 (0x00007f4ecf646000)
        libXinerama.so.1 => /usr/lib64/libXinerama.so.1 (0x00007f4ecf443000)
        libXi.so.6 => /usr/lib64/libXi.so.6 (0x00007f4ecf234000)
        libXrandr.so.2 => /usr/lib64/libXrandr.so.2 (0x00007f4ecf02c000)
        libXcursor.so.1 => /usr/lib64/libXcursor.so.1 (0x00007f4ecee21000)
        libXcomposite.so.1 => /usr/lib64/libXcomposite.so.1 (0x00007f4ecec1f000)
        libXdamage.so.1 => /usr/lib64/libXdamage.so.1 (0x00007f4ecea1d000)
        libresolv.so.2 => /lib64/libresolv.so.2 (0x00007f4ece802000)
        libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f4ece5e3000)
        libpixman-1.so.0 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/anaconda_lib/lib/./libpixman-1.so.0 (0x00007f4ece363000)
        libpng16.so.16 => /gruntdata/app_data/lanbo.llb/deps_lib_shared/anaconda_lib/lib/./libpng16.so.16 (0x00007f4ece121000)
        libxcb.so.1 => /usr/lib64/libxcb.so.1 (0x0000
        libXau.so.6 => /usr/lib64/libXau.so.6 (0x0000
```

开发环境的依赖库

```
$ldd ./build/lib/libcaffe.so.1.0.0-rc3
        linux-vdso.so.1 =>  (0x00007fffd9f79000)
        libglog.so.0 => not found
        libgflags.so.2.2 => not found
        libprotobuf.so.7 => /apsara/lib64/libprotobuf.so.7 (0x00007f274a7cd000)
        libboost_system.so.1.60.0 => not found
        libboost_filesystem.so.1.60.0 => not found
        libboost_regex.so.1.60.0 => not found
        libhdf5_hl.so.10 => not found
        libhdf5.so.10 => not found
        libopencv_core.so.2.4 => not found
        libopencv_highgui.so.2.4 => not found
        libopencv_imgproc.so.2.4 => not found
        libboost_thread.so.1.60.0 => not found
        libstdc++.so.6 => /usr/lib64/libstdc++.so.6 (0x00007f274a4ca000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f274a246000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f274a038000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f2749e1d000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f2749ac4000)
        libz.so.1 => /lib64/libz.so.1 (0x00007f27498b0000)
        /lib64/ld-linux-x86-64.so.2 (0x00000036e7800000)
```

生产环境的依赖库

主要特点

- C++
  - 技术细节全面采用C++实现
  - 依赖proto buffer/boost开发库
  - GPU加速采用cuda开发
  - OpenCV/LMDB等仅仅只是补充功能，并不是核心模块
- Layer based
  - 采用DSL定义网络结构
  - layer/blob/net/solver 等结构

- 优点/缺点

- (+) Good for feedforward networks
- (+) Good for finetuning existing networks
- (+) Train models without writing any code!
- (+) Python interface is pretty useful!
- (-) Need to write C++ / CUDA for new GPU layers
- (-) Not good for recurrent networks
- (-) Cumbersome for big networks (GoogLeNet, ResNet)

Torch

http://torch.ch

- From NYU + IDIAP
- Written in C and Lua
- Used a lot a Facebook, DeepMind

- 最灵活，最容易学习的深度学习框架

- Lua 语言
  - 类似Javascript的动态语言，LuaJit性能高于大多数的脚本语言
  - 结合C语言开发非常简单，通过LuaFFI直接调用C编写的模块

Learn Lua in 15 Minutes

more or less

For a more in-depth Lua tutorial, watch this video or check out a transcript of the video.

```
-- Two dashes start a one-line comment.

--[[
     Adding two ['s and ]'s makes it a
     multi-line comment.
--]]

----------------------------------------------------
-- 1. Variables and flow control.
----------------------------------------------------

num = 42  -- All numbers are doubles.
-- Don't freak out, 64-bit doubles have 52 bits for
-- storing exact int values; machine precision is
-- not a problem for ints that need < 52 bits.

s = 'walternate'  -- Immutable strings like Python.
t = "double-quotes are also fine"
u = [[ Double brackets
       start and end
       multi-line strings.]]
t = nil  -- Undefines t; Lua has garbage collection.

-- Blocks are denoted with keywords like do/end:
while num < 50 do
  num = num + 1  -- No ++ or += type operators.
end
```

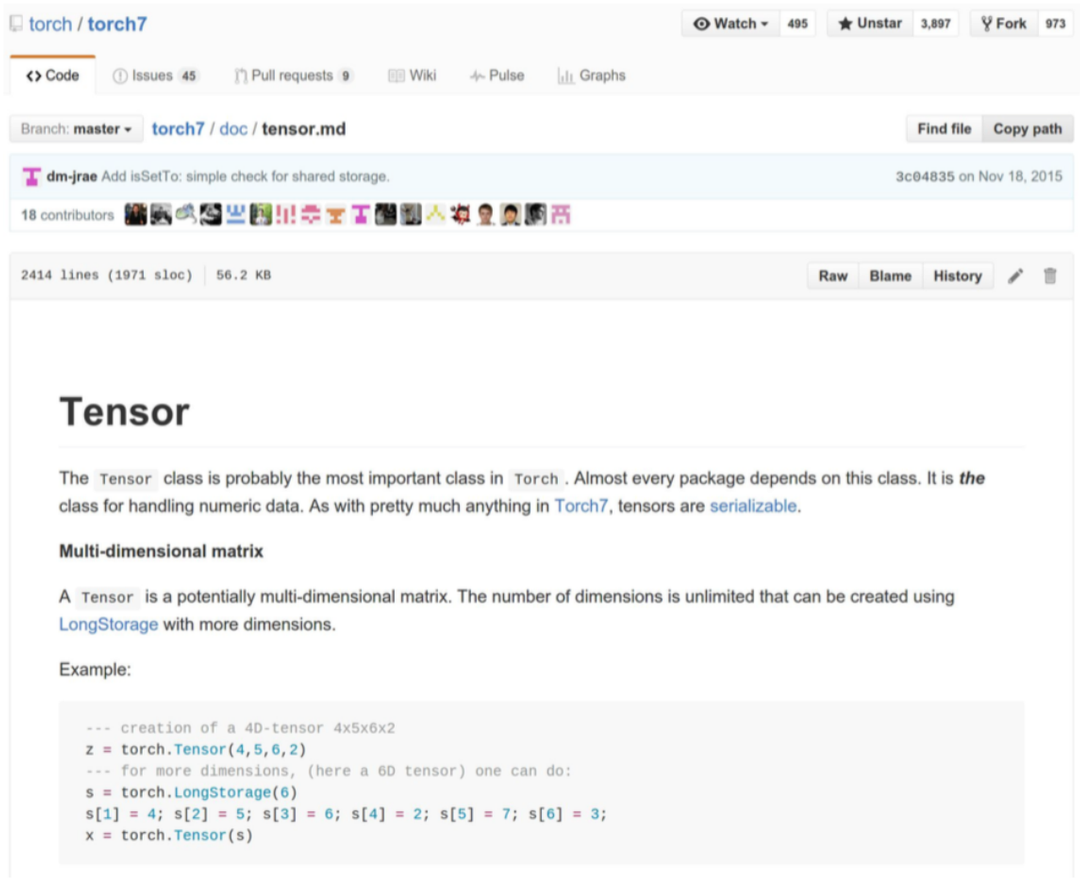
Torch: Tensors

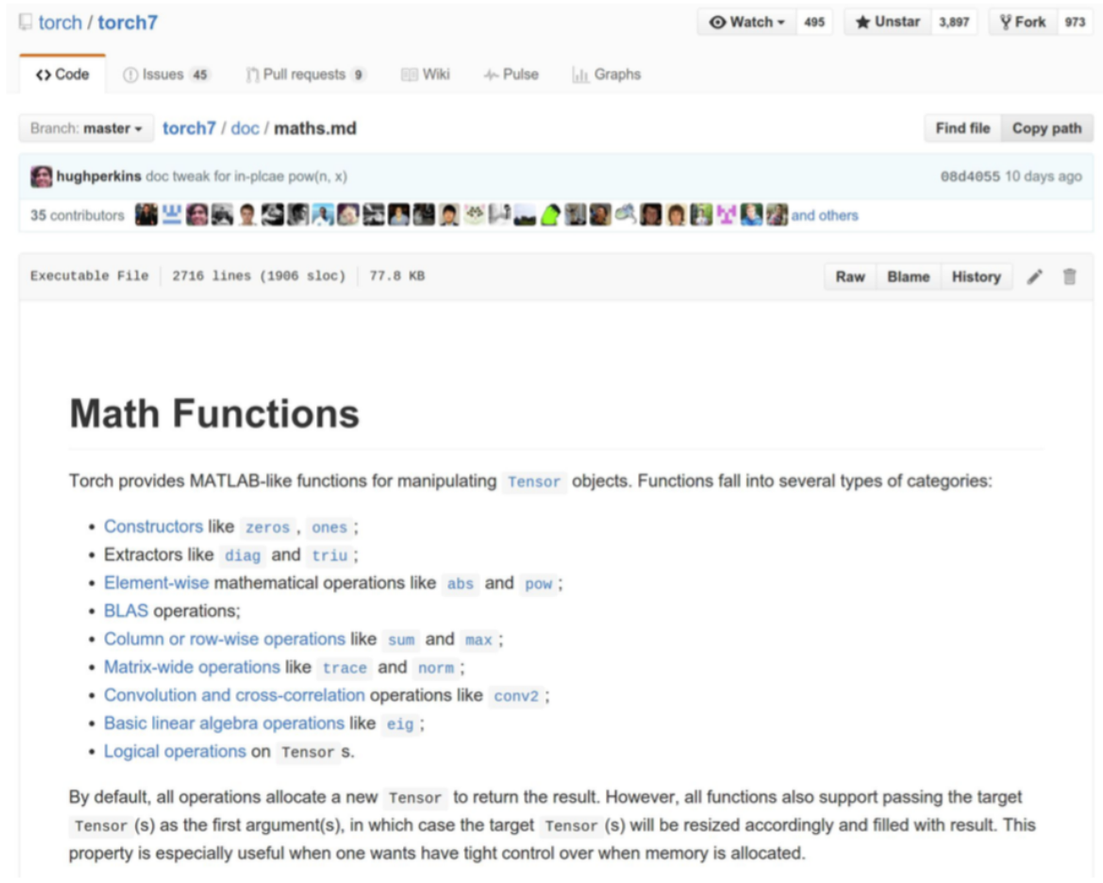
Torch tensors are just like numpy arrays

```python
import numpy as np

# Simple feedforward network (no biases) in numpy

# Batch size, input dim, hidden dim, num classes
N, D, H, C = 100, 1000, 100, 10

# First and second layer weights
w1 = np.random.randn(D, H)
w2 = np.random.randn(H, C)

# Random input data
x = np.random.randn(N, D)

# Forward pass
a = x.dot(w1)         # First layer
a = np.maximum(a, 0) # In-place ReLU
scores = a.dot(w2)    # Second layer

print scores
```

```lua
require 'torch'

-- Simple feedforward network (no biases) in torch

-- Batch size, input dim, hidden dim, num classes
local N, D, H, C = 100, 1000, 100, 10

-- First and second layer weights
local w1 = torch.randn(D, H)
local w2 = torch.randn(H, C)

-- Random input data
local x = torch.randn(N, D)

-- Forward pass
local a = torch.mm(x, w1)       -- First layer
a:cmax(0)                       -- In-place ReLU
local scores = torch.mm(a, w2)  -- Second layer

print(scores)
```

Documentation on GitHub:





- 所见即所得的交互式环境

```
aliwork:show_learning teaonly$ th

                               |  Torch7
  ______                  __   |  Scientific computing for Lua.
 /_  __/__  ________ ____/ /   |  Type ? for help
  / / / _ \/ __/ __/ _ \ _ \   |  https://github.com/torch
 /_/  \___/_/  \__/_//_/       |  http://torch.ch

th> require('nn');
                                                                [0.0471s]
th> sm = nn.SoftMax();
                                                                [0.0001s]
th> x = torch.rand(10);
                                                                [0.0080s]
th> sm:forward(x)
 0.1167
 0.0916
 0.0840
 0.1522
 0.1108
 0.1063
 0.0741
 0.0862
 0.0886
 0.0896
[torch.DoubleTensor of size 10]

                                                                [0.0108s]
th>
```

- 基于”nn”软件包生成各种网络

```lua
require('nn')

-- Model Container
local model = nn.Sequential()

model:add(nn.SpatialConvolution(3, 32, 3, 3, 1, 1, 1))
model:add(nn.ReLU())
model:add(nn.SpatialMaxPooling(2, 2))
model:add(nn.SpatialConvolution(32, 64, 3, 3, 1, 1, 1))
model:add(nn.ReLU())
model:add(nn.SpatialMaxPooling(2, 2))
model:add(nn.SpatialConvolution(64, 64, 3, 3, 1, 1, 1))
model:add(nn.ReLU())
model:add(nn.SpatialMaxPooling(2, 2))
model:add(nn.SpatialConvolution(64, 128, 3, 3, 1, 1, 1))
model:add(nn.ReLU())
model:add(nn.SpatialMaxPooling(2, 2))
model:add(nn.SpatialConvolution(128, 128, 3, 3, 1, 1, 1))
model:add(nn.ReLU())
model:add(nn.SpatialConvolution(128, 128, 3, 3, 1, 1, 1))
model:add(nn.ReLU())
model:add(nn.SpatialMaxPooling(2, 2))
model:add(nn.Reshape(128*4*4))
model:add(nn.Dropout(0.5))
model:add(nn.Linear(2048, 2048))
model:add(nn.Dropout(0.5))
model:add(nn.ReLU())
model:add(nn.Linear(2048, 2048))
model:add(nn.Dropout(0.5))
model:add(nn.ReLU())
model:add(nn.Linear(2048, 1))
model:add(nn.Sigmoid())
return model
```

- 训练过程全程可控

网络定义

权重/梯度

loss function

forward 计算

backword计算

权重更新

```lua
require 'torch'
require 'nn'


-- Batch size, input dim, hidden dim, num classes
local N, D, H, C = 100, 1000, 100, 10

-- Build a one-layer ReLU network
local net = nn.Sequential()
net:add(nn.Linear(D, H))
net:add(nn.ReLU())
net:add(nn.Linear(H, C))

-- Collect all weights and gradients in a single Tensor
local weights, grad_weights = net:getParameters()

-- Loss functions are called "criterions"
local crit = nn.CrossEntropyCriterion()  -- Softmax loss

-- Generate some random input data
local x = torch.randn(N, D)
local y = torch.Tensor(N):random(C)

-- Forward pass: Compute scores and loss
local scores = net:forward(x)
local loss = crit:forward(scores, y)

-- Backward pass: compute gradients
grad_weights:zero()
local dscores = crit:backward(scores, y)
local dx = net:backward(x, dscores)

-- Make a gradient step
local learning_rate = 1e-3
weights:add(-learning_rate, grad_weights)

```

- GPU / CPU 切换
  - require(‘cutorch’)
  - require(‘cunn’)
  - 切换tensor类型

```lua
require 'torch'
require 'cutorch'
require 'nn'
require 'cunn'

-- Batch size, input dim, hidden dim, num classes
local N, D, H, C = 100, 1000, 100, 10

local dtype = 'torch.CudaTensor'

-- Build a one-layer ReLU network
local net = nn.Sequential()
net:add(nn.Linear(D, H))
net:add(nn.ReLU())
net:add(nn.Linear(H, C))
net:type(dtype)

-- Collect all weights and gradients in a single Tensor
local weights, grad_weights = net:getParameters()

-- Loss functions are called "criterions"
local crit = nn.CrossEntropyCriterion()  -- Softmax loss
crit:type(dtype)

-- Generate some random input data
local x = torch.randn(N, D):type(dtype)
local y = torch.Tensor(N):random(C):type(dtype)

-- Forward pass: Compute scores and loss
local scores = net:forward(x)
local loss = crit:forward(scores, y)

-- Backward pass: compute gradients
grad_weights:zero()
local dscores = crit:backward(scores, y)
local dx = net:backward(x, dscores)

-- Make a gradient step
local learning_rate = 1e-3
weights:add(-learning_rate, grad_weights)
```

- 利用optim模块

包含模块

迭代函数

sgd算法实现模块

```lua
require 'torch'
require 'nn'
require 'optim'

-- Batch size, input dim, hidden dim, num classes
local N, D, H, C = 100, 1000, 100, 10

-- Build a one-layer ReLU network
local net = nn.Sequential()
net:add(nn.Linear(D, H))
net:add(nn.ReLU())
net:add(nn.Linear(H, C))

-- Collect all weights and gradients in a single Tensor
local weights, grad_weights = net:getParameters()

-- Loss functions are called "criterions"
local crit = nn.CrossEntropyCriterion()  -- Softmax loss

-- Calback to interface with optim methods
local function f(w)
  assert(w == weights)

  -- Generate some random input data
  local x = torch.randn(N, D)
  local y = torch.Tensor(N):random(C)

  -- Forward pass: Compute scores and loss
  local scores = net:forward(x)
  local loss = crit:forward(scores, y)

  -- Backward pass: compute gradients
  grad_weights:zero()
  local dscores = crit:backward(scores, y)
  local dx = net:backward(x, dscores)

  return loss, grad_weights
end

-- Make a step using Adam
local state = {learningRate=1e-3}
optim.adam(f, weights, state)
```

Torch: Modules

Tons of built-in modules and loss functions

New ones all the time:

- Abs.lua
- AbsCriterion.lua
- Add.lua
- AddConstant.lua
- BCECriterion.lua
- BatchNormalization.lua
- Bilinear.lua
- CAddTable.lua
- CDivTable.lua
- CMakeLists.txt
- CMul.lua
- CMulTable.lua
- TemporalConvolution.lua
- TemporalMaxPooling.lua
- TemporalSubSampling.lua
- Threshold.lua
- Transpose.lua
- View.lua
- VolumetricAveragePooling.lua
- VolumetricConvolution.lua
- VolumetricDropout.lua
- VolumetricFullConvolution.lua
- VolumetricMaxPooling.lua
- VolumetricMaxUnpooling.lua
- WeightedEuclidean.lua
- WeightedMSECriterion.lua
- MarginCriterion.lua
- MarginRankingCriterion.lua
- Max.lua
- Mean.lua
- Min.lua
- MixtureTable.lua
- Module.lua
- Mul.lua
- MulConstant.lua
- MultiCriterion.lua
- MultiLabelMarginCriterion.lua
- MultiLabelSoftMarginCriterion.lua
- MultiMarginCriterion.lua
- Narrow.lua

Added 2/19/2016

Added 2/16/2016

- SparseLinear.lua
- SpatialAdaptiveMaxPooling.lua
- SpatialAveragePooling.lua
- SpatialBatchNormalization.lua
- SpatialContrastiveNormalization.lua
- SpatialConvolution.lua
- SpatialConvolutionLocal.lua
- SpatialConvolutionMM.lua
- SpatialConvolutionMap.lua
- SpatialCrossMapLRN.lua
- SpatialDivisiveNormalization.lua
- SpatialDropout.lua
- SpatialFractionalMaxPooling.lua
- SpatialFullConvolution.lua
- SpatialFullConvolutionMap.lua
- SpatialLPPooling.lua
- SpatialMaxPooling.lua
- SpatialMaxUnpooling.lua
- ClassSimplexCriterion.lua
- Concat.lua
- ConcatTable.lua
- Container.lua
- Contiguous.lua
- Copy.lua
- Cosine.lua
- CosineDistance.lua
- CosineEmbeddingCriterion.lua
- Criterion.lua
- CriterionTable.lua
- CrossEntropyCriterion.lua
- DepthConcat.lua
- DistKLDivCriterion.lua
- DotProduct.lua
- Dropout.lua
- ELU.lua

https://github.com/torch/nn

- 开发自己定义的模块

Module

`Module` is an abstract class which defines fundamental methods necessary for a training a neural network. Modules are serializable.

Modules contain two states variables: output and gradInput.

[output] forward(input)

Takes an `input` object, and computes the corresponding `output` of the module. In general `input` and `output` are Tensors. However, some special sub-classes like table layers might expect something else. Please, refer to each module specification for further information.

After a `forward()`, the output state variable should have been updated to the new value.

It is not advised to override this function. Instead, one should implement updateOutput(input) function. The forward module in the abstract parent class Module will call `updateOutput(input)`.

[gradInput] backward(input, gradOutput)

Performs a *backpropagation step* through the module, with respect to the given `input`. In general this method makes the assumption forward(input) has been called before, *with the same input*. This is necessary for optimization reasons. If you do not respect this rule, `backward()` will compute incorrect gradients.

In general `input` and `gradOutput` and `gradInput` are Tensors. However, some special sub-classes like table layers might expect something else. Please, refer to each module specification for further information.

A *backpropagation step* consist in computing two kind of gradients at `input` given `gradOutput` (gradients with respect to the output of the module). This function simply performs this task using two function calls:

Torch: Modules

Writing your own modules is easy!

TimesTwo.lua

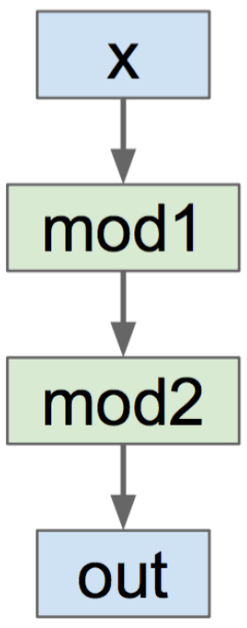
```lua
require 'nn'

local times_two, parent = torch.class('nn.TimesTwo', 'nn.Module')


function times_two:__init()
  parent.__init(self)
end


function times_two:updateOutput(input)
  self.output:mul(input, 2)
  return self.output
end


function times_two:updateGradInput(input, gradOutput)
  self.gradInput:mul(gradOutput, 2)
  return self.gradInput
end
```

times_two_example.lua

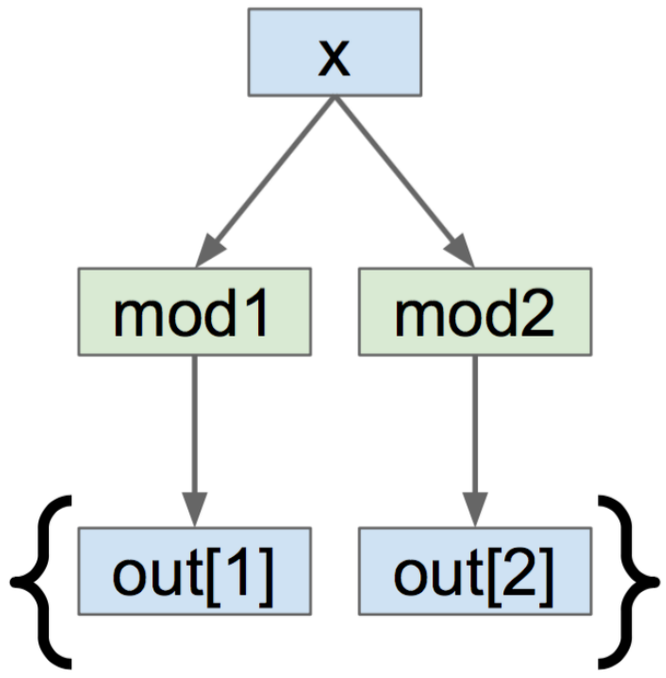
```lua
require 'nn'

require 'TimesTwo'

local times_two = nn.TimesTwo()

local input = torch.randn(4, 5)
local output = times_two:forward(input)

print('here is input:')
print(input)

print('here is output:')
print(output)

local gradOutput = torch.randn(4, 5)
local gradInput = times_two:backward(input, gradOutput)

print('here is gradOutput:')
print(gradOutput)

print('here is gradInput')
print(gradInput)
```

Container modules allow you to combine multiple modules

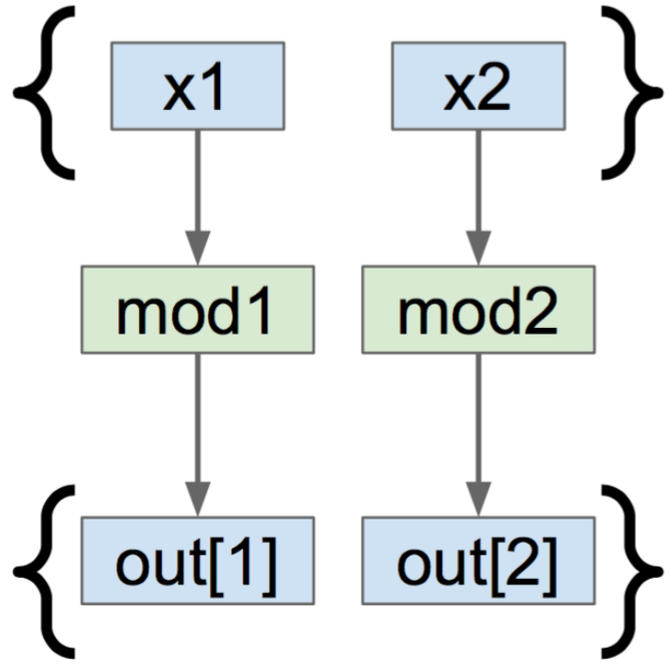
```
local seq = nn.Sequential()
seq:add(mod1)
seq:add(mod2)
local out = seq:forward(x)
```




```
local concat = nn.ConcatTable()
concat:add(mod1)
concat:add(mod2)
local out = concat:forward(x)
```



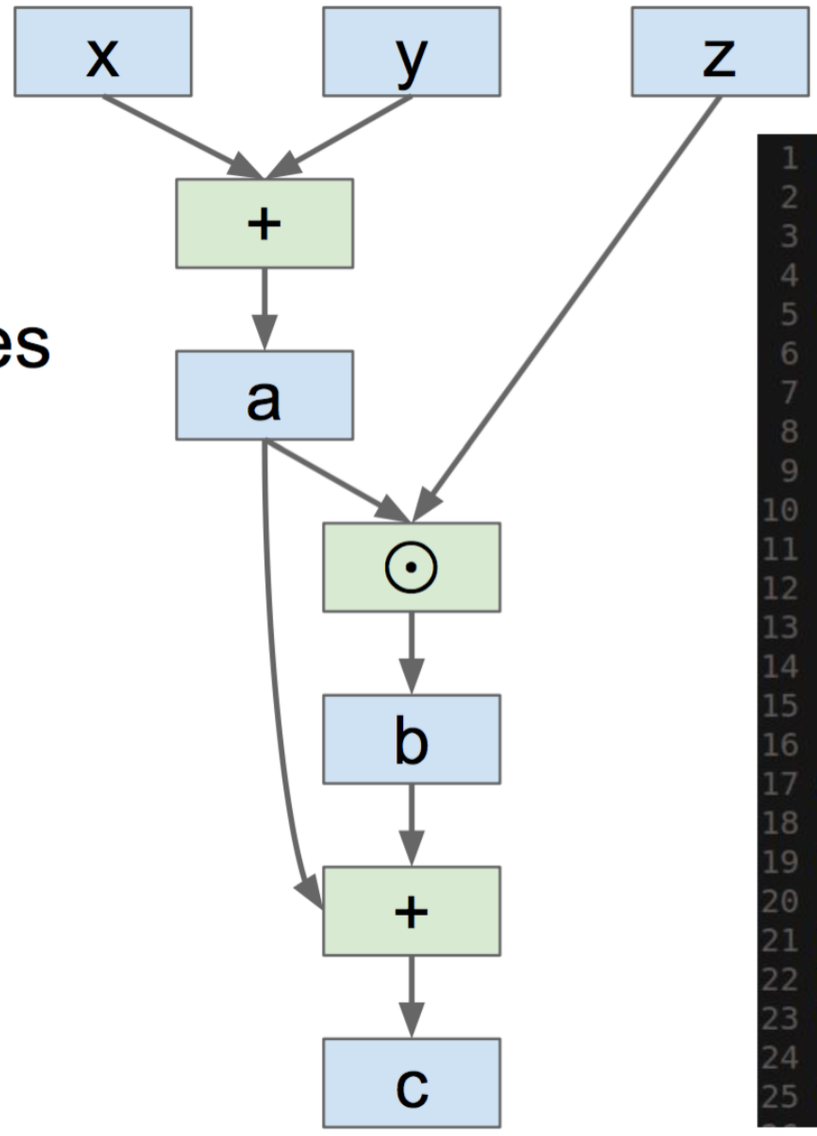

```
local parallel = nn.ParallelTable()
parallel:add(mod1)
parallel:add(mod2)
local out = parallel:forward({x1, x2})
```



Torch: nngraph

Use nngraph to build modules that combine their inputs in complex ways

Inputs: x, y, z
Outputs: c
a = x + y
b = a ⊙ z
c = a + b




```lua
require 'torch'
require 'nn'
require 'nngraph'

local function build_module()
  local x = nn.Identity()()
  local y = nn.Identity()()
  local z = nn.Identity()()

  local a = nn.CAddTable()({x, y})
  local b = nn.CMulTable()({a, z})
  local c = nn.CAddTable()({a, b})

  local inputs = {x, y, z}
  local outputs = {c}
  return nn.gModule(inputs, outputs)
end

local mod = build_module()

local x = torch.randn(4, 5)
local y = torch.randn(4, 5)
local z = torch.randn(4, 5)

local c = mod:forward({x, y, z})
```

主要特点

- 灵活的框架
  - 使用Lua语言做为控制应用层，彻底的模块化设计
- 友好的交互
  - 快速学习深度学习中各个组件的功能
- 全程可控的训练过程
  - 适合实验新的网络类型

实验室用的多，生产环境用的少

Torch: Pros / Cons

- (-) Lua
- (-) Less plug-and-play than Caffe
  - You usually write your own training code
- (+) Lots of modular pieces that are easy to combine
- (+) Easy to write your own layer types and run on GPU
- (+) Most of the library code is in Lua, easy to read
- (+) Lots of pretrained models!
- (-) Not great for RNNs

TensorFlow

https://www.tensorflow.org

- Google 研发/主推
- 基于computation graphs 计算模型
- Python/C++语言
- TensorBoard 可视化工具
- 多机多卡支持

- computation graphs vs layer base

- 首先构造好整个计算链路
  - 可以对链路进行优化
  - 分布式调度容易实现

- 基于层模型
  - 每个层的计算，固定实现 forward/backword
  - 必须手动指定目标GPU卡

```python
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

sess = tf.InteractiveSession()
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)

x = tf.placeholder(tf.float32, shape=[None, 784])
y_ = tf.placeholder(tf.float32, shape=[None, 10])

W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))

y = tf.nn.softmax(tf.matmul(x,W) + b)

cross_entropy = -tf.reduce_sum(y_*tf.log(y))

train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)


sess.run(tf.initialize_all_variables())

for i in xrange(5000):
    batch = mnist.train.next_batch(32)
    train_step.run(feed_dict={x: batch[0], y_: batch[1]})
    if i%100 == 0:
        loss = cross_entropy.eval(feed_dict={x: batch[0], y_: batch[1]})
        print ("====> %g" % loss)


correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(accuracy.eval(feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

输入节点

参数变量

运算节点

优化模块

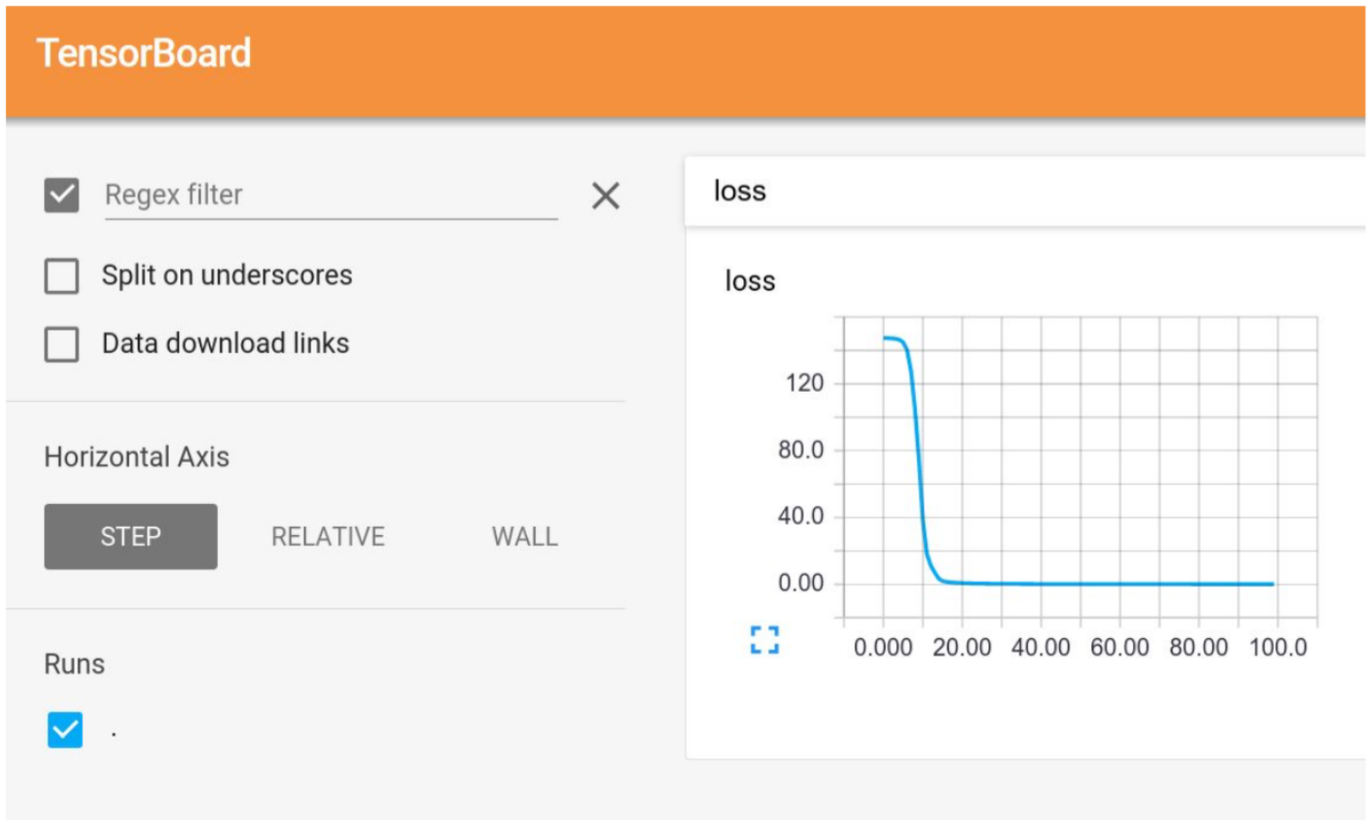
训练过程

```
 4 import tensorflow as tf
 5 from tensorflow.examples.tutorials.mnist import input_data
 6
 7 sess = tf.InteractiveSession()
 8 mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
 9
10 x = tf.placeholder(tf.float32, shape=[None, 784])
11 y_ = tf.placeholder(tf.float32, shape=[None, 10])
12
13 W = tf.Variable(tf.zeros([784,10]))
14 b = tf.Variable(tf.zeros([10]))
15
16 y = tf.nn.softmax(tf.matmul(x,W) + b)
17
18 cross_entropy = -tf.reduce_sum(y_*tf.log(y))
19
20 train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
21
22
23 sess.run(tf.initialize_all_variables())
24
25 for i in xrange(5000):
26     batch = mnist.train.next_batch(32)
27     train_step.run(feed_dict={x: batch[0], y_: batch[1]})
28     if i%100 == 0:
29         loss = cross_entropy.eval(feed_dict={x: batch[0], y_: batch[1]})
30         print ("====> %g" % loss)
31
32
33 correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
34 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
35 print(accuracy.eval(feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

- 利用ipython工具进行交互式调试

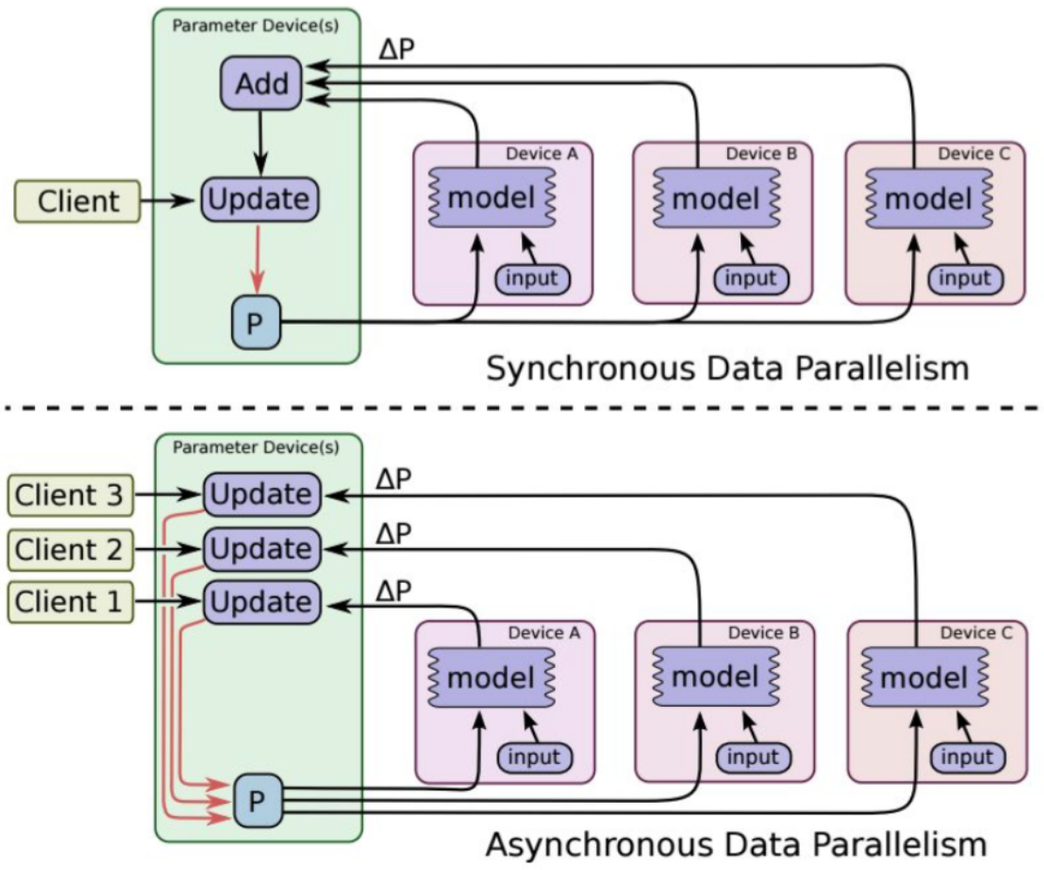
TensorFlow: Tensorboard

Start Tensorboard server, and we get graphs!

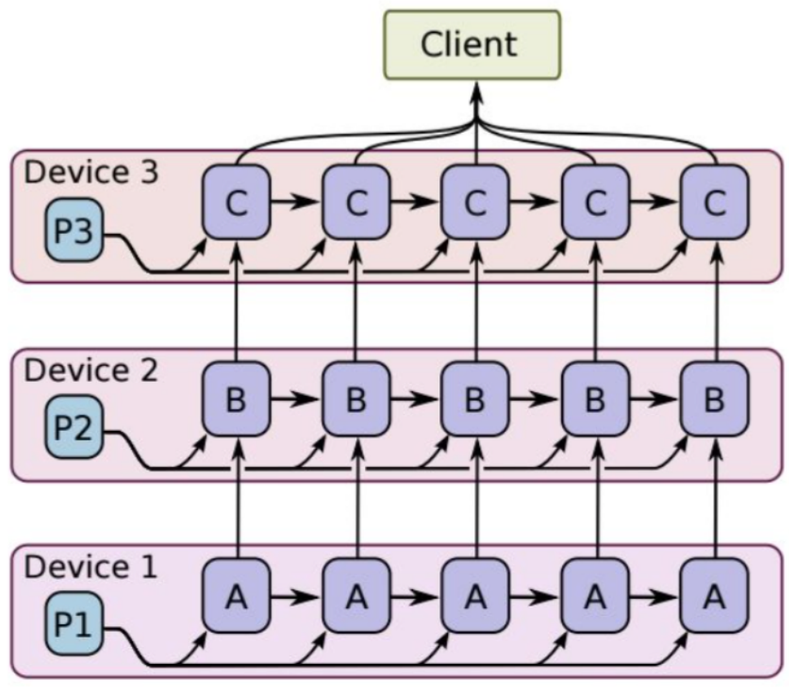


TensorFlow: Multi-GPU

Data parallelism:
synchronous or asynchronous

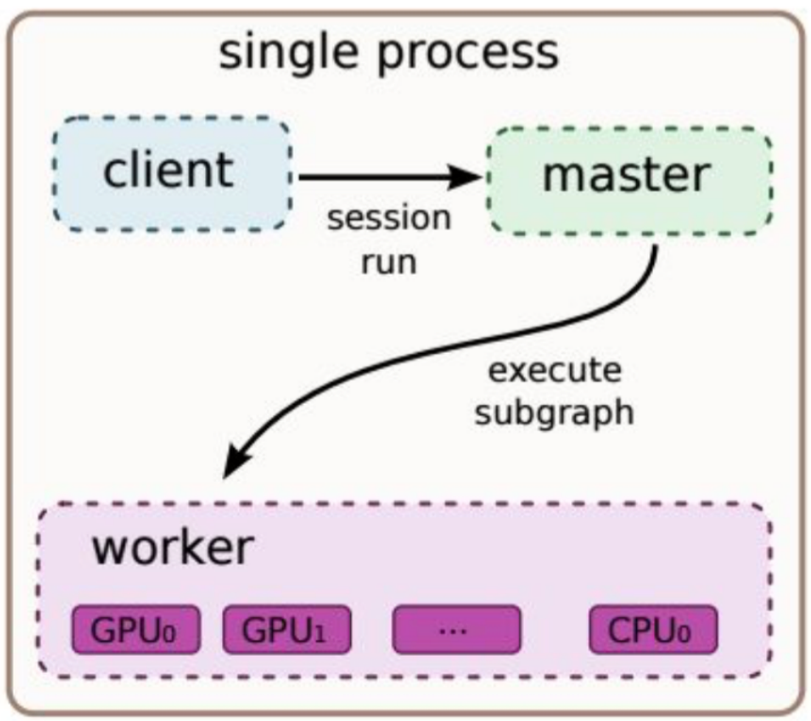


Model parallelism:
Split model across GPUs

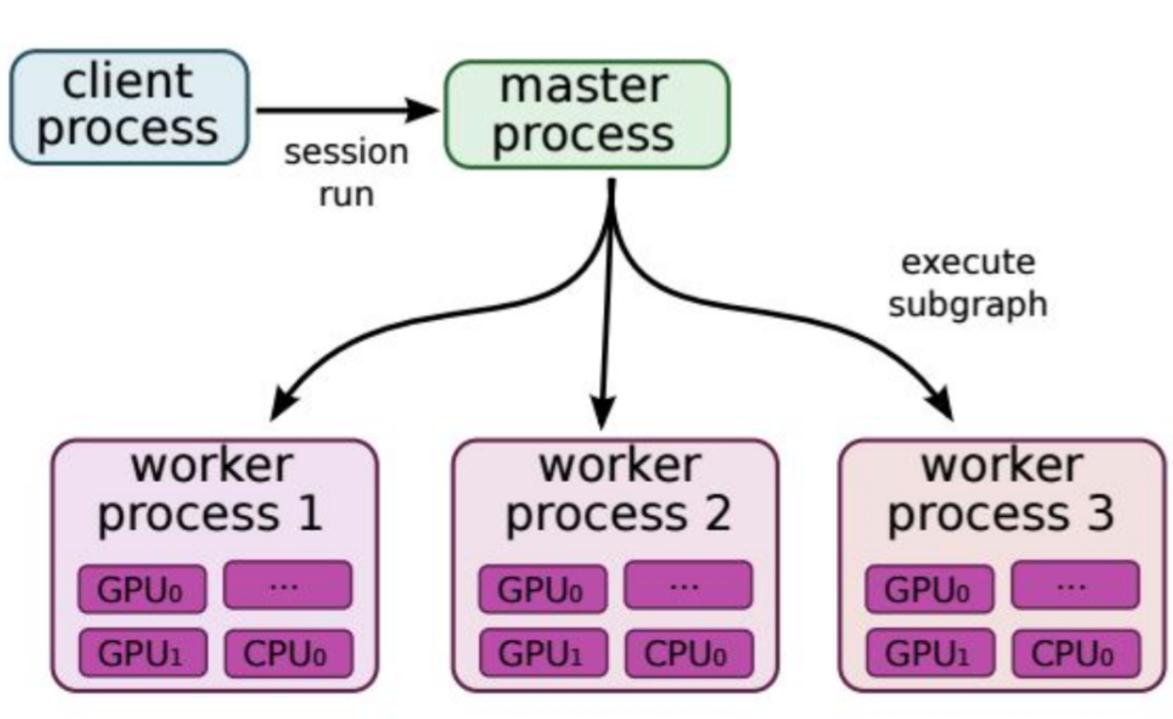


TensorFlow: Distributed

Single machine:
Like other frameworks



Many machines:
Not open source (yet) =(



- 优点/缺点

- (+) Python + numpy
- (+) Computational graph abstraction, like Theano; great for RNNs
- (+) Much faster compile times than Theano
- (+) Slightly more convenient than raw Theano?
- (+) TensorBoard for visualization
- (+) Data AND model parallelism; best of all frameworks
- (+/-) Distributed models, but not open-source yet
- (-) Slower than other frameworks right now
- (-) Much “fatter” than Torch; more magic
- (-) Not many pretrained models

MxNet

https://github.com/dmlc/mxnet

- 和TensorFlow类似，但在支持图计算的基础上，增加过程模型，即支持两种计算模型
- 比TensoFlow更为轻量

Flexible

Supports both imperative and symbolic programmings.

Portable

Runs on CPUs or GPUs, on clusters, servers, desktops, or mobile phones

Multiple Lanuages

Supports over 7 programming languages, including C++, Python, R, Scala, Julia, Matlab, and Javascripts.

Auto-Differentiation

Calculates the gradient automatically for training a model.

Distributed on Cloud

Supports distributed training on multiple CPU/GPU machines, including AWS, GCE, Azure, and Yarn clusters.

Performance

The well-optimized C++ backend engine parallelize both I/O and computations

```python
def get_mlp():
    """
    multi-layer perceptron
    """
    data = mx.symbol.Variable('data')
    fc1  = mx.symbol.FullyConnected(data = data, name='fc1', num_hidden=128)
    act1 = mx.symbol.Activation(data = fc1, name='relu1', act_type="relu")
    fc2  = mx.symbol.FullyConnected(data = act1, name = 'fc2', num_hidden = 64)
    act2 = mx.symbol.Activation(data = fc2, name='relu2', act_type="relu")
    fc3  = mx.symbol.FullyConnected(data = act2, name='fc3', num_hidden=10)
    mlp  = mx.symbol.SoftmaxOutput(data = fc3, name = 'softmax')
    return mlp
```

```
model = mx.model.FeedForward(
    ctx                = devs,
    symbol             = network,
    num_epoch          = args.num_epochs,
    learning_rate      = args.lr,
    momentum           = 0.9,
    wd                 = 0.00001,
    initializer        = mx.init.Xavier(factor_type="in", magnitude=2.34),
    **model_args)

eval_metrics = ['accuracy']
## TopKAccuracy only allows top_k > 1
for top_k in [5, 10, 20]:
    eval_metrics.append(mx.metric.create('top_k_accuracy', top_k = top_k))
```

```
if batch_end_callback is not None:
    if not isinstance(batch_end_callback, list):
        batch_end_callback = [batch_end_callback]
else:
    batch_end_callback = []
batch_end_callback.append(mx.callback.Speedometer(args.batch_size, 50))

model.fit(
    X                  = train,
    eval_data          = val,
    eval_metric        = eval_metrics,
    kvstore            = kv,
    batch_end_callback = batch_end_callback,
    epoch_end_callback = checkpoint)
```

总结一下

	Caffe	**Torch**	**TensorFlow**	**MxNet**
Language	C++, Python	Lua	Python	C++, Python
Pretrained	Yes ++	Yes ++	Inception	
Multi-GPU: Data parallel	Yes	Yes cunn. DataParallelTable	Yes	YES
Multi-GPU: Model parallel	No	Yes fbcunn.ModelParallel	Yes (best)	YES
Readable source code	Yes (C++)	Yes (Lua)	No	YES(C++ 11)
Good at RNN	No	Mediocre	Yes (best)	YES

- 从Torch入门，容易学习，只需要克服Lua语言
- 最应该关注TensorFlow
- Caffe目前还是产品化最多的库
- 可以了解一下MxNet