caffe的网络构建过程

训练网络的构建过程

第一步

在tools/caffe.cpp的train()函数中创建求解器，这是创建整个网络的入口。

```
229   shared_ptr<caffe::Solver<float> >
230       solver(caffe::SolverRegistry<float>::CreateSolver(solver_param));
```

这里我们以最基本的SGDSolver求解器为例来分析。

第二步

通过工厂模式的机制调用回调函数来生成SGDSolver的一个对象，关于caffe的工厂模式和回调机制我在前面已经分析过了，这里就不展开分析了。

(include/caffe/solver_factory.hpp)

```
73   static Solver<Dtype>* CreateSolver(const SolverParameter& param) {
74     const string& type = param.type();
75     CreatorRegistry& registry = Registry();
76     CHECK_EQ(registry.count(type), 1) << "Unknown solver type: " << type
77         << " (known types: " << SolverTypeListString() << ")";
78     return registry[type](param);   //通过回调创建一个实例
```

第三步

我们知道SGDSolver以及其派生类的基类都是继承自Solver类，创建SGDSolver类对象之前先会调用Solver类的构造函数，那我们先看看Solver类的构造函数吧，我们看到它调用了Init函数，那我们继续瞅瞅看吧。

(src/caffe/solver.cpp)

```
29 template <typename Dtype>
30 Solver<Dtype>::Solver(const SolverParameter& param)
31     : net_(), callbacks_(), requested_early_exit_(false) {
32   Init(param);
33 }
```

第四步

(src/caffe/solver.cpp)

```
44 void Solver<Dtype>::Init(const SolverParameter& param) {
45   LOG_IF(INFO, Caffe::root_solver()) << "Initializing solver from parameters: "
46     << std::endl << param.DebugString();
47   param_ = param;
48   CHECK_GE(param_.average_loss(), 1) << "average_loss should be non-negative.";
```

```
49   CheckSnapshotWritePermissions();
50   if (param_.random_seed() >= 0) {
51     Caffe::set_random_seed(param_.random_seed() + Caffe::solver_rank());
52   }
53   // Scaffolding code
54   InitTrainNet();            //初始化训练网络
55   InitTestNets();            //初始化测试网络
56   if (Caffe::root_solver()) {
57     LOG(INFO) << "Solver scaffolding done.";
58   }
59   iter_ = 0;
60   current_step_ = 0;
61 }
```

在Init函数中，最关键的是调用了InitTrainNet()函数和InitTestNet()函数，等等为什么我们选择的是训练模式这里会有测试网络呢？这个后面再分析，我们先沿着InitTrainNet()函数往下走。

第五步

(src/caffe/solver.cpp)

```
110   NetState net_state;
111   net_state.set_phase(TRAIN);
112   net_state.MergeFrom(net_param.state());
113   net_state.MergeFrom(param_.train_state());
114   net_param.mutable_state()->CopyFrom(net_state);
115   net_.reset(new Net<Dtype>(net_param));       //看到它创建网络结构了
116   for (int w_idx = 0; w_idx < param_.weights_size(); ++w_idx) {
117     LoadNetWeights(net_, param_.weights(w_idx));
118   }
```

InitTrainNet函数中还有一些其他的判断信息，这里只列出了最关键的代码。我们看到再115行这边创建了一根指向Net类的指针，即创建了一个Net实例。终于看到了网络骨架的影子，那我们看看Net类的构造函数吧。

第六步

(src/caffe/net.cpp)

```
25 Net<Dtype>::Net(const NetParameter& param) {
26   Init(param);
27 }
```

构造函数相当简单，调用了Init函数，但是这个Init函数就非常复杂了，涵盖了大量的信息。那下面我们一点一点的看这个Init函数具体做了写什么。

第七步

筛选层，因为我们定义的prototxt训练文件中有些层是专门拿来测试用的，而我们这里是创建训练网络，所以需要筛选一下层。

(src/caffe/net.cpp)

```
51   NetParameter filtered_param;
52   FilterNet(in_param, &filtered_param);
```

比如我们这里使用的mnist的lenet训练网络，则我们可以打印看一下哪些层被筛选进来了。

```
筛选之前的层
mnist mnist conv1 pool1 bn1 conv2 pool2 ip1 relu1 ip2 accuracy loss
筛选之后的层
mnist conv1 pool1 bn1 conv2 pool2 ip1 relu1 ip2 loss
```

这里有两个层被扔掉了，一个mnist数据层，一个是accuracy层，这两个都是测试的时候用到的层，所以不要了，如果想看具体是如何筛选的则可以看FilterNet这个函数。

第八步

设置一些容器的大小和一些参数，因为Net类中有很多成员变量，我不能一下子全部介绍，只有用到一点介绍一点。

(src/caffe/net.cpp)

```
61   map<string, int> blob_name_to_idx;    //每个blob对应的index
62   set<string> available_blobs;      //存放blob
63   memory_used_ = 0; //记录内存的使用量
64   // For each layer, set up its input and output
65   bottom_vecs_.resize(param.layer_size());
66   top_vecs_.resize(param.layer_size());
67   bottom_id_vecs_.resize(param.layer_size());
68   param_id_vecs_.resize(param.layer_size());
69   top_id_vecs_.resize(param.layer_size());
70   bottom_need_backward_.resize(param.layer_size());
```

caffe的in-place计算

对于一些层，其输入bottom blob和输出的top blob可以是同一个blob，比如ReLU层，BN层，这些层的输入blob和输出的blob的维度是一样的，所以我们可以使用相同的blob。所以我们使用了in-place计算可以节省一定的内存空间。但是同时也不是所有的层都是可以采用in-place计算的，比如conv层、pooling层、dnn层。因为这些层的输入和输出的blob的维度是不一样的，所以很显然就不能采用in-place计算，假设我们在BN层使用了in-place计算，则其日志打印为：

```
I0522 20:45:44.213629 15667 net.cpp:112] Creating Layer bn1
I0522 20:45:44.213635 15667 net.cpp:474] bn1 <- pool1
I0522 20:45:44.213649 15667 net.cpp:435] bn1 -> pool1 (in-place)
I0522 20:45:44.213925 15667 net.cpp:155] Setting up bn1
```