

## 强晓萃

----一个爱打球的程序猿

## Caffemodel解析

因为工作需要最近一直在琢磨Caffe，纯粹新手，写博客供以后查阅方便，请大神们批评指正！

Caffe中，数据的读取、运算、存储都是采用Google Protocol Buffer来进行的，所以首先来较为详细的介绍下Protocol Buffer(PB)。

PB是一种**轻便、高效的结构化数据存储格式**，可以用于结构化数据串行化，很适合做数据存储或 RPC 数据交换格式。它可用于通讯协议、数据存储等领域的**语言无关、平台无关、可扩展的序列化结构数据格式**。是一种效率和兼容性都很优秀的**二进制数据传输格式**，目前提供了C++、Java、Python 三种语言的 API。Caffe采用的是C++和Python的API。

接下来，我用一个简单的例子来说明一下。

使用PB和 C++ 编写一个十分简单的例子程序。该程序由两部分组成。第一部分被称为Writer，第二部分叫做Reader。Writer 负责将一些结构化的数据写入一个磁盘文件，Reader则负责从该磁盘文件中读取结构化数据并打印到屏幕上。准备用于演示的结构化数据是HelloWorld，它包含两个基本数据：

ID，为一个整数类型的数据；

Str，这是一个字符串。

首先我们需要编写一个proto文件，定义我们程序中需要处理的结构化数据，**Caffe是定义在caffe.proto文件中**。在PB的术语中，结构化数据被称为 Message。proto文件非常类似java或C语言的数据定义。代码清单 1 显示了例子应用中的proto文件内容。

## 清单 1. proto 文件

```
1 package lm;
2
3 message helloworld
4 {
5     {
6         required int32      id = 1;    // ID
7         required string     str = 2;    // str
8         optional int32      opt = 3;    // optional field
9     }
10 }
11
12
13 }
```

2017年9月						
<	日	一	二	三	四	五
	27	28	29	30	31	1
	3	4	5	6	7	8
	10	11	12	13	14	15
	17	18	19	20	21	22
	24	25	26	27	28	29
	1	2	3	4	5	6

## 导航

[博客园](#)[首页](#)[新随笔](#)[联系](#)[订阅 XML](#)[管理](#)

## 统计

[随笔 - 3](#)[文章 - 0](#)[评论 - 8](#)[引用 - 0](#)

## 公告

昵称：强晓萃

园龄：2年5个月

粉丝：7

关注：1

+加关注

## 搜索

## 常用链接

[我的随笔](#)[我的评论](#)[我的参与](#)[最新评论](#)[我的标签](#)

## 我的标签

[CNN\(1\)](#)[DeepID\(1\)](#)

一个比较好的习惯是认真对待proto文件的文件名。比如将命名规则定于如下：packageName.MessageName.proto

在上例中，package名字叫做lm，定义了一个消息helloworld，该消息有三个成员，类型为int32的id，另一个为类型为string的成员str。optional是一个可选的成员，即消息中可以不包含该成员，required表明是必须包含该成员。一般在定义中会出现如下三个字段属性：

对于required的字段而言，初值是必须要提供的，否则字段的便是未初始化的。在Debug模式的buffer库下编译的话，序列化话的时候可能会失败，而且在反序列化的时候对于该字段的解析会总是失败的。所以，对于修饰符为required的字段，请在序列化的时候务必给予初始化。

对于optional的字段而言，如果未进行初始化，那么一个默认值将赋予该字段，当然也可以指定默认值。

对于repeated的字段而言，该字段可以重复多个，谷歌提供的这个addressbook例子便有个很好的该修饰符的应用场景，即每个人可能有多多个电话号码。在高级语言里面，我们可以通过数组来实现，而在proto定义文件中可以使用repeated来修饰，从而达到相同目的。当然，出现0次也是包含在内的。

写好proto文件之后就可以用PB编译器(protoc)将该文件编译成目标语言了。本例中我们将使用C++。假设proto文件存放在\$SRC\_DIR下面，您也想把生成的文件放在同一个目录下，则可以使用如下命令：

```
1 protoc -I=$SRC_DIR --cpp_out=$DST_DIR $SRC_DIR/addressbook.proto
```

命令将生成两个文件：

lm.helloworld.pb.h，定义了C++类的头文件；**这个头文件十分重要,解析时主要关注它的类定义。**

lm.helloworld.pb.cc，C++类的实现文件。

在生成的头文件中，定义了一个C++类helloworld，后面的Writer和Reader将使用这个类来对消息进行操作。诸如对消息的成员进行赋值，将消息序列化等等都有相应的方法。

如前所述，Writer将把一个结构化数据写入磁盘，以便其他人来读取。假如我们不使用PB，其实也有许多的选择。一个可能的方法是将数据转换为字符串，然后将字符串写入磁盘。转换为字符串的方法可以使用sprintf()，这非常简单。数字123可以变成字符串"123"。这样做似乎没有什么不妥，但是仔细考虑一下就会发现，这样的做法对写Reader的那个人的要求比较高，Reader的作者必须了解Writer的细节。比如"123"可以是单个数字123，但也可以是三个数字1、2和3等等。这么说来，我们还必须让Writer定义一种分隔符一样的字符，以便Reader可以正确读取。但分隔符也许还会引起其他的什么问题。最后我们发现一个简单的Helloworld也需要写许多处理消息格式的代码。

如果使用PB，那么这些细节就可以不需要应用程序来考虑了。使用PB，Writer的工作很简单，需要处理的结构化数据由.proto文件描述，经过上一节中的编译过程后，该数据化结构对应了一个C++的类，并定义在lm.helloworld.pb.h中。对于本例，类名为lm::helloworld。

Writer需要include该头文件，然后便可以使用这个类了。现在，在Writer代码中，**将要存入磁盘的结构化数据由一个lm::helloworld类的对象表示**，它提供了一系列的get/set函数用来修改和读取结构化数据中的数据成员，或者叫field。

当我们需要将该结构化数据保存到磁盘上时，类lm::helloworld已经提供相应的方法把一个复杂的数据变成一个字节序列，我们可以将这个字节序列写入磁盘。

对于想要读取这个数据的程序来说，也只需要使用类lm::helloworld的相应反序列化方法来将这个字节序列重新转换成结构化数据。这同我们开始时那个"123"的想法类似，不过PB想的远远比我们那个粗糙的字符串转换要全面，因此，我们可以放心将这类事情交给PB吧。程序清单2演示了Writer的主要代码。

#### 清单 2. Writer 的主要代码

☐

DL(1)  
Face Verification(1)  
Object Detection(1)  
YOLO(1)

#### 随笔分类

DL(3)  
论文笔记(2)

#### 随笔档案

2016年1月 (1)  
2015年4月 (1)  
2015年3月 (1)

#### 最新评论

1. Re:Caffemodel解析  
@Hucley不好意思 隔这么久才回答你 我这种方式解析很繁琐 你可以通过ipython去解析 Caffemodel Caffe官网上有例子...

--强晓萃

2. Re:Caffemodel解析  
@强晓萃我还是有很多地方不明白，我今天训练了一下MNIST手写库，然后也同样出现对'caffe::NetParameter::NetParameter()'未定义的引用的错误，请问楼主你在使用的时候有.....

--Hucley

3. Re:Caffemodel解析  
@Hucley基本思路是一致的，你得根据自己的实际情况适当的改下...

--强晓萃

4. Re:Caffemodel解析  
@强晓萃解析的时候代码和楼主你举得MNIST手写库例子一样么（改动。caffemodel文件就可以），还是要根据自己的模型重新编写。我用你这上面的代码出现错误：错误：'length'在此作用域中尚未.....

--Hucley

5. Re:Caffemodel解析  
@Hucley恩恩，你先解析出来，保存成二进制文件，然后MATLAB去读取就可以了。。。...  
--强晓萃

#### 阅读排行榜

1. Caffemodel解析(7314)  
2. You Only Look Once:Unified, Real-Time Object Detection论文笔记(2364)

### 评论排行榜

1. Caffemodel解析(8)

### 推荐排行榜

1. Caffemodel解析(3)

```
1 #include "lm.helloworld.pb.h"
2
3 ...
4
5 int main(void)
6
7 {
8
9     lm::helloworld msg1;
10
11     msg1.set_id(101);           //设置id
12
13     msg1.set_str("hello");      //设置str
14
15     // 向磁盘中写入数据流fstream
16
17     fstream output("./log", ios::out | ios::trunc | ios::binary);
18
19     if (!msg1.SerializeToOstream(&output)) {
20
21         cerr << "Failed to write msg." << endl;
22
23         return -1;
24
25     }
26
27     return 0;
28
29 }
```

Msg1 是一个helloworld类的对象，set\_id()用来设置id的值。SerializeToOstream将对象序列化后写入一个fstream流。我们可以写出Reader代码，程序清单3列出了 reader 的主要代码。

#### 清单 3. Reader的主要代码

```
1 #include "lm.helloworld.pb.h"
2
3 ...
4
5 void ListMsg(const lm::helloworld & msg) {
6
7     cout << msg.id() << endl;
```

```

8
9  cout << msg.str() << endl;
10
11 }
12
13 int main(int argc, char* argv[]) {
14
15     lm::helloworld msg1;
16
17     {
18
19         fstream input("./log", ios::in | ios::binary);
20
21         if (!msg1.ParseFromIstream(&input)) {
22
23             cerr << "Failed to parse address book." << endl;
24
25             return -1;
26
27         }
28     }
29 }
30
31 ListMsg(msg1);
32
33 ...
34
35 }

```

同样，Reader 声明类helloworld的对象msg1，然后利用ParseFromIstream从一个fstream流中读取信息并反序列化。此后，ListMsg中采用get方法读取消息的内部信息，并进行打印输出操作。

运行Writer和Reader的结果如下：

```


>writer
>reader
101
Hello

```

Reader 读取文件 log 中的序列化信息并打印到屏幕上。这个例子本身并无意义，但只要稍加修改就可以将它变成更加有用的程序。比如将磁盘替换为网络 socket，那么就可以实现基于网络的数据交换任务。而存储和交换正是PB最有效的应用领域。

到这里为止，我们只给出了一个简单的没有任何用处的例子。在实际应用中，人们往往需要定义更加复杂的 Message。我们用“复杂”这个词，不仅仅是指从个数上说有更多的 fields 或者更多类型的 fields，而是指更加复杂的数据结构：**嵌套 Message**，Caffe.proto文件中定义了大量的嵌套Message。使得Message的表达能力增强很多。代码清单 4 给出一个嵌套 Message 的例子。

#### 清单 4. 嵌套 Message 的例子

 View Code

在 Message Person 中，定义了嵌套消息 PhoneNumber，并用来定义 Person 消息中的 phone 域。这使得人们可以定义更加复杂的数据结构。

以上部分参考网址：<http://www.ibm.com/developerworks/cn/linux/l-cn-gpb/>

在Caffe中也是类似于上例中的Writer和Reader去读写PB数据的。接下来，具体说明下Caffe中是如何存储Caffemodel的。在Caffe主目录下的 **solver.cpp** 文件中的一段代码展示了Caffe是如何存储Caffemodel的，代码清单5如下：

#### 清单 5. Caffemodel存储代码


```
1 template <typename Dtype>
2
3 void Solver<Dtype>::Snapshot() {
4
5     NetParameter net_param;    // NetParameter为网络参数类
6
7     // 为了中间结果，也会写入梯度值
8
9     net_ -> ToProto(&net_param, param_.snapshot_diff());
10
11     string filename(param_.snapshot_prefix());
12
13     string model_filename, snapshot_filename;
14
15     const int kBufferSize = 20;
16
17     char iter_str_buffer[kBufferSize];
18
19     // 每训练完1次，iter_就加1
20
21     snprintf(iter_str_buffer, kBufferSize, "_iter_%d", iter_ + 1);
22
23     filename += iter_str_buffer;
24
25     model_filename = filename + ".caffemodel"; //XX_iter_YY.caffemodel
26
27     LOG(INFO) << "Snapshotting to " << model_filename;
28
29     // 向磁盘写入网络参数
30
31     WriteProtoToBinaryFile(net_param, model_filename.c_str());
32
33     SolverState state;
34
```

```

35 SnapshotSolverState(&state);
36
37 state.set_iter(iter_ + 1);    //set
38
39 state.set_learned_net(model_filename);
40
41 state.set_current_step(current_step_);
42
43 snapshot_filename = filename + ".solverstate";
44
45 LOG(INFO) << "Snapshotting solver state to " << snapshot_filename;
46
47 // 向磁盘写入网络state
48
49 WriteProtoToBinaryFile(state, snapshot_filename.c_str());
50
51 }

```



在清单5代码中，我们可以看到，其实Caffemodel存储的数据也就是网络参数net\_param的PB，Caffe可以保存每一次训练完成后的网络参数，我们可以通过XX.prototxt文件来进行参数设置。在这里的 WriteProtoToBinaryFile函数与之前HelloWorld例子中的Writer函数类似，在这就不在贴出。那么我们只要弄清楚NetParameter类的组成，也就明白了Caffemodel的具体数据构成。在caffe.proto这个文件中定义了NetParameter类，如代码清单6所示。

#### 清单6. Caffemodel存储代码

```

1 message NetParameter {
2
3   optional string name = 1;    // 网络名称
4
5   repeated string input = 3;   // 网络输入input blobs
6
7   repeated BlobShape input_shape = 8; // The shape of the input blobs
8
9   // 输入维度blobs, 4维(num, channels, height and width)
10
11   repeated int32 input_dim = 4;
12
13   // 网络是否强制每层进行反馈操作开关
14
15   // 如果设置为False, 则会根据网络结构和学习率自动确定是否进行反馈操作
16
17   optional bool force_backward = 5 [default = false];
18
19   // 网络的state, 部分网络层依赖, 部分不依赖, 需要看具体网络

```

```
20
21 optional NetState state = 6;
22
23 // 是否打印debug log
24
25 optional bool debug_info = 7 [default = false];
26
27 // 网络层参数, Field Number 为100, 所以网络层参数在最后
28
29 repeated LayerParameter layer = 100;
30
31 // 弃用: 用 'layer' 代替
32
33 repeated V1LayerParameter layers = 2;
34
35 }
36
37 // Specifies the shape (dimensions) of a Blob.
38
39 message BlobShape {
40
41   repeated int64 dim = 1 [packed = true];
42
43 }
44
45 message BlobProto {
46
47   optional BlobShape shape = 7;
48
49   repeated float data = 5 [packed = true];
50
51   repeated float diff = 6 [packed = true];
52
53   optional int32 num = 1 [default = 0];
54
55   optional int32 channels = 2 [default = 0];
56
57   optional int32 height = 3 [default = 0];
58
59   optional int32 width = 4 [default = 0];
60
61 }
62
63
64
65 // The BlobProtoVector is simply a way to pass multiple blobproto instances
66
67 around.
```

```
68
69 message BlobProtoVector {
70
71   repeated BlobProto blobs = 1;
72
73 }
74
75 message NetState {
76
77   optional Phase phase = 1 [default = TEST];
78
79   optional int32 level = 2 [default = 0];
80
81   repeated string stage = 3;
82
83 }
84
85 message LayerParameter {
86
87   optional string name = 1; // the layer name
88
89   optional string type = 2; // the layer type
90
91   repeated string bottom = 3; // the name of each bottom blob
92
93   repeated string top = 4; // the name of each top blob
94
95   // The train/test phase for computation.
96
97   optional Phase phase = 10;
98
99   // Loss weight值: float
100
101   // 每一层为每一个top blob都分配了一个默认值，通常是0或1
102
103   repeated float loss_weight = 5;
104
105   // 指定的学习参数
106
107   repeated ParamSpec param = 6;
108
109   // The blobs containing the numeric parameters of the layer.
110
111   repeated BlobProto blobs = 7;
112
113   // included/excluded.
114
115   repeated NetStateRule include = 8;
```



```
116
117 repeated NetStateRule exclude = 9;
118
119 // Parameters for data pre-processing.
120
121 optional TransformationParameter transform_param = 100;
122
123 // Parameters shared by loss layers.
124
125 optional LossParameter loss_param = 101;
126
127 // 各种类型层参数
128
129 optional AccuracyParameter accuracy_param = 102;
130
131 optional ArgMaxParameter argmax_param = 103;
132
133 optional ConcatParameter concat_param = 104;
134
135 optional ContrastiveLossParameter contrastive_loss_param = 105;
136
137 optional ConvolutionParameter convolution_param = 106;
138
139 optional DataParameter data_param = 107;
140
141 optional DropoutParameter dropout_param = 108;
142
143 optional DummyDataParameter dummy_data_param = 109;
144
145 optional EltwiseParameter eltwise_param = 110;
146
147 optional ExpParameter exp_param = 111;
148
149 optional HDF5DataParameter hdf5_data_param = 112;
150
151 optional HDF5OutputParameter hdf5_output_param = 113;
152
153 optional HingeLossParameter hinge_loss_param = 114;
154
155 optional ImageDataParameter image_data_param = 115;
156
157 optional InfogainLossParameter infogain_loss_param = 116;
158
159 optional InnerProductParameter inner_product_param = 117;
160
161 optional LRNParameter lrn_param = 118;
162
163 optional MemoryDataParameter memory_data_param = 119;
```

```
164
165 optional MVNParameter mvn_param = 120;
166
167 optional PoolingParameter pooling_param = 121;
168
169 optional PowerParameter power_param = 122;
170
171 optional PythonParameter python_param = 130;
172
173 optional ReLUParameter relu_param = 123;
174
175 optional SigmoidParameter sigmoid_param = 124;
176
177 optional SoftmaxParameter softmax_param = 125;
178
179 optional SliceParameter slice_param = 126;
180
181 optional TanHParameter tanh_param = 127;
182
183 optional ThresholdParameter threshold_param = 128;
184
185 optional WindowDataParameter window_data_param = 129;
186
187 }
```



那么接下来的一段代码来演示如何解析Caffemodel，我解析用的model为MNIST手写库训练后的model，Lenet\_iter\_10000.caffemodel。

#### 清单7. Caffemodel解析代码

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

```
#include <stdio.h>
#include <string.h>
#include <fstream>
#include <iostream>
#include "proto/caffe.pb.h"

using namespace std;
using namespace caffe;

int main(int argc, char* argv[])
{
    caffe::NetParameter msg;
```

```
16  fstream input("lenet_iter_10000.caffemodel", ios::in | ios::binary);
17  if (!msg.ParseFromIstream(&input))
18  {
19      cerr << "Failed to parse address book." << endl;
20      return -1;
21  }
22
23  ::google::protobuf::RepeatedPtrField< LayerParameter >* layer = msg.mutable_layer();
24  ::google::protobuf::RepeatedPtrField< LayerParameter >::iterator it = layer->begin();
25  for (; it != layer->end(); ++it)
26  {
27      cout << it->name() << endl;
28      cout << it->type() << endl;
29      cout << it->convolution_param().weight_filler().max() << endl;
30  }
31
32  return 0;
33 }
```



参考网址：<http://www.cnblogs.com/stephen-liu74/archive/2013/01/04/2842533.html>

分类: DL

好文要顶

关注我

收藏该文



强晓萃

关注 - 1

粉丝 - 7

+加关注

3

0

« 上一篇：[Deep Learning Face Representation from Predicting 10,000 Classes论文笔记](#)

» 下一篇：[You Only Look Once:Unified, Real-Time Object Detection论文笔记](#)

posted on 2015-04-19 16:50 强晓萃 阅读(7314) 评论(8) 编辑 收藏

## 评论

#1楼 2015-05-14 23:09 天苍苍野茫茫

能结合更多实例就好了

支持(0) 反对(0)

## #2楼[楼主] 2015-05-15 08:33 强晓苹

@ 天苍苍野茫茫

恩恩，其实我觉得其他model都大同小异，所以仅仅举了一例。

支持(0) 反对(0)

## #3楼 2015-07-20 16:50 Hucley

楼主，要想知道.caffemodel中得数据，也是先要将caffemodel文件解析出来么？  
最近需要将.caffemodel中的数据提取出来转换成.mat文件，不知楼主能否指点一二

支持(0) 反对(0)

## #4楼[楼主] 2015-07-20 17:05 强晓苹

@ Hucley

恩恩，你先解析出来，保存成二进制文件，然后MATLAB去读取就可以了。。。

支持(0) 反对(0)

## #5楼 2015-07-20 22:14 Hucley

@ 强晓苹

解析的时候代码和楼主你举得MNIST手写库例子一样么（改动.caffemodel文件就可以），还是要根据自己的模型重新编写。我用你这上面的代码出现错误：错误：'length'在此作用域中尚未声明，如果把 printf("length = %d\n", length);这句去掉又会出现/tmp/ccntFjUs.o：在函数'main'中：

caffemodel.cpp(.text+0x33)：'caffe::NetParameter::NetParameter()'未定义的引用等一系列错误，楼主有时间能为我解答以下么？

支持(0) 反对(0)

## #6楼[楼主] 2015-07-21 08:24 强晓苹

@ Hucley

基本思路是一致的，你得根据自己的实际情况适当的改下

支持(0) 反对(0)

## #7楼 2015-07-21 09:08 Hucley

@ 强晓苹

我还是有很多地方不明白，我今天训练了一下MNIST手写库，然后也同样出现对'caffe::NetParameter::NetParameter()'未定义的引用的错误，  
请问楼主你在使用的时候有没有出现同样的问题，能否指教一下？

支持(0) 反对(0)

## #8楼[楼主] 2016-01-12 17:02 强晓苹

@ Hucley

不好意思 隔这么久才回答你 我这种方式解析很繁琐 你可以通过ipython去解析Caffemodel Caffe官网上有例子

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【推荐】腾讯云上实验室 1小时搭建人工智能应用

【推荐】可嵌入您系统的“在线Excel”！SpreadJS 纯前端表格控件

【推荐】阿里云“全民云计算”优惠升级



#### 最新IT新闻:

- HTC难兄难弟：华硕利润创七年新低后也要追求利润优先
  - 顺丰将在湖北鄂州建国际物流机场 预计年内开建
  - 微博新协议对大V动粗，可是得罪了大V真的不影响赚钱吗？
  - 停止对人类的迷恋！AI完全模仿人类大脑是在浪费时间
  - 阿里文娱成立现场娱乐事业群 张宇兼任CEO
- » 更多新闻...



#### 最新知识库文章:

- Google 及其云智慧
  - 做到这一点，你也可以成为优秀的程序员
  - 写给立志做码农的大学生
  - 架构腐化之谜
  - 学会思考，而不只是编程
- » 更多知识库文章...

---

Powered by:  
博客园  
Copyright © 强晓萃