

Harder&&Better

目录视图

摘要视图

RSS 订阅

个人资料



航子_harder

访问：2079次

积分：170

等级：BLOG > 2

排名：千里之外

原创：14篇 转载：4篇

译文：0篇 评论：0条

文章搜索

异步赠书：Kotlin领衔10本好书 SDCC 2017之区块链技术实战线上峰会 程序员9月书讯 每周荐书：Java Web、Python极客编程
(评论送书)

图片集

标签：图片

2017-09-06 20:03

21人阅读

评论

分类：图像处理(2)

版权声明：本文为博主原创文章，未经博主允许不得转载。

常用数据集记录

数据库	图像数	类别数	应用场景	图像大小	难度
Mnist	60000	10	字符识别	28*28	容易
Cifar10	60000	10	物体分类	32*32	中等
PASCAL VOC 20010-2012	9963	20	分类检测	470*380	很难
ImageNet	1400万	10万	物体分类	500*400	很难
KITTI	15000	-	车辆分类检测	-	中等
CompCars	30000	281	车辆分类	-	中等

关闭

文章分类

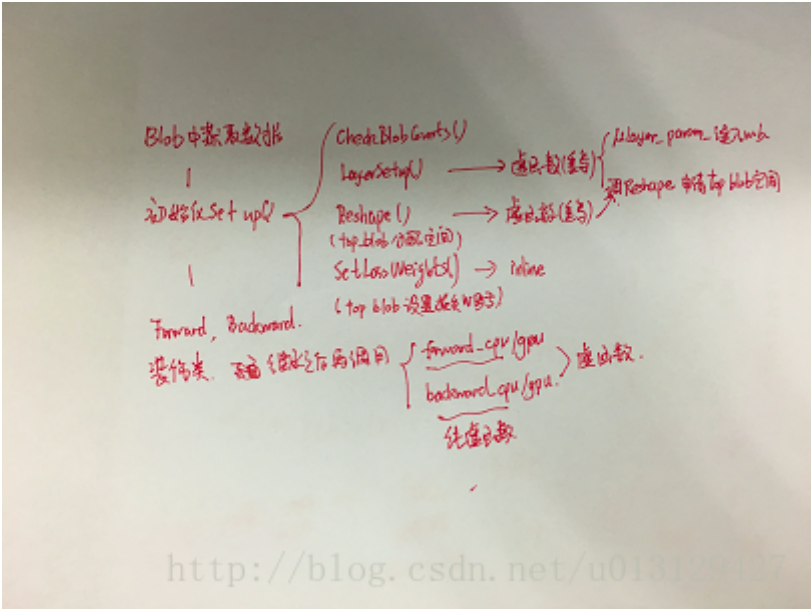
- 深度学习 (7)
- 图像处理 (3)
- C++深入 (1)
- 机器学习 (0)
- Python深入 (5)
- 刷题总结 (1)

文章存档

- 2017年09月 (1)
- 2017年07月 (3)
- 2017年06月 (1)
- 2017年05月 (13)

阅读排行

- rcnn-fast-rcnn--faster-rcr (211)
- kaggle--mnist--knn 比赛, (199)
- python-numpy小结3 (187)
- spyder--no module name (183)
- Fast_rcnn+Caffe配置以及 (162)
- caffe环境问题以及ubuntu (144)
- ubuntu14.04 + cuda 7.5 (134)
- 传统物体检测 (122)
- Faster-Rcnn demo.py解 (100)
- Python-Numpy小结1 (97)



可以根据自己的需要，从已有的类中选择自己继承的类，这样可以省却很多麻烦。一般可以继承的类包括

层	描述
Layer	如果你要定义的层和已有的层没有什么重叠，那么可以选择直接继承Layer
DataLayer	自定义网络输入层时，可以考虑继承它，内部的load_batch可能是你要重写的函数
NeuronLayer	自定义神经层，也就是中间的进行运算的层
LossLayer	如果现有的损失函数层不能满足需求，可以继承它

<http://blog.csdn.net/u013129427>

关闭

评论排行

图片集	(0)
Jupyter--闪退 解决办法	(0)
python-numpy小结3	(0)
Leetcode--Path Sum I,II,	(0)
python-numpy小结2	(0)
Python-Numpy小结1	(0)
传统物体检测	(0)
Fast_rcnn+Caffe配置以及	(0)
欢迎使用CSDN-markdov	(0)
caffe环境问题以及ubuntu	(0)

推荐文章

- * CSDN日报20170828——《4个方法快速打造你的阅读清单》
- * Android检查更新下载安装
- * 动手打造史上最简单的Recycleview 侧滑菜单
- * TCP网络通讯如何解决分包粘包问题
- * SDCC 2017之区块链技术实战线上峰会
- * 快速集成一个视频直播功能

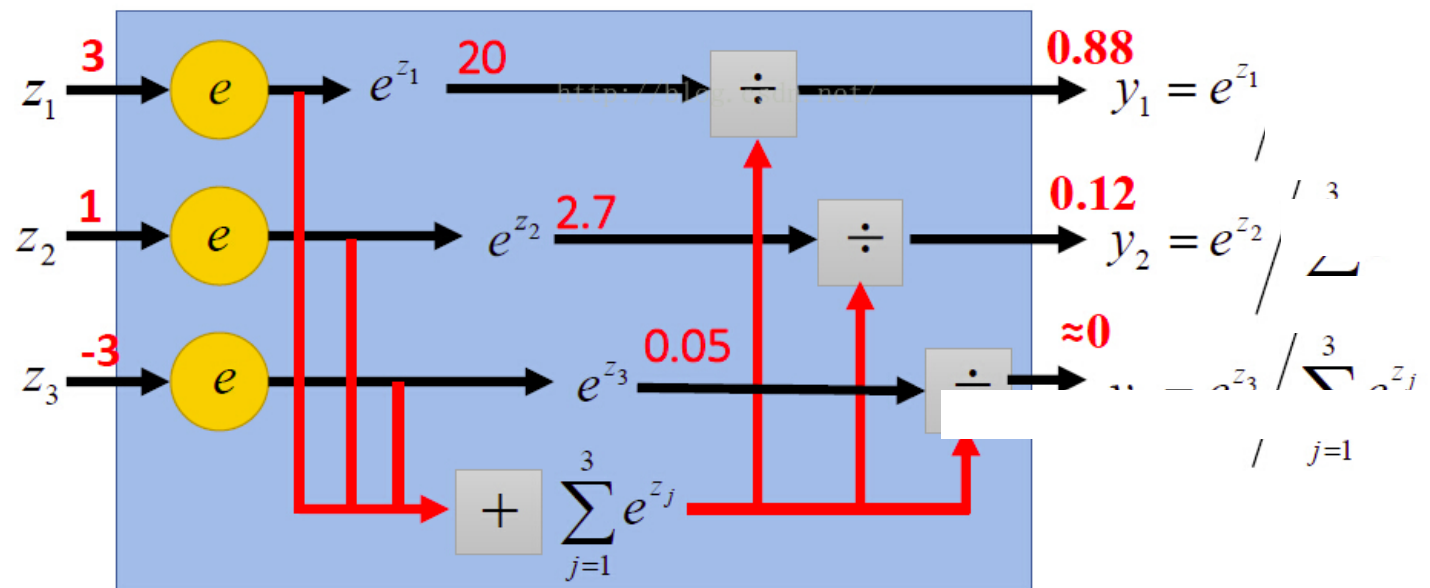
Output Layer (Option)

- Softmax layer as the output layer

Probability:

- $1 > y_i > 0$
- $\sum_i y_i = 1$

Softmax Layer



<http://blog.csdn.net/u013129427>

关闭

$$Z_4 = W_{41}x_1 + W_{42}x_2 + W_{43}x_3$$

$$Z_5 = W_{51}x_1 + W_{52}x_2 + W_{53}x_3$$

$$Z_6 = W_{61}x_1 + W_{62}x_2 + W_{63}x_3$$

$$a_4 = \frac{e^{Z_4}}{e^{Z_4} + e^{Z_5} + e^{Z_6}} = \sigma(Z_4) \quad \text{Loss} = -\sum_j y_j \ln a_j - (1 - y_j) \ln (1 - a_j)$$

$$\text{if } j=i \quad \text{Loss} = -\ln a_i$$

$$\frac{\partial L}{\partial W_{41}} = \left[\frac{\partial L}{\partial a_4} \right] \cdot \left[\frac{\partial a_4}{\partial Z_4} \right] \cdot \left[\frac{\partial Z_4}{\partial W_{41}} \right]$$

$$j=i \quad \frac{\partial L}{\partial Z_4} = \frac{\partial}{\partial Z_4} \left(\frac{e^{Z_4}}{\sum_k e^{Z_k}} \right) = \frac{(e^{Z_4})' \cdot \sum_k e^{Z_k} - e^{Z_4} \cdot e^{Z_4}}{(\sum_k e^{Z_k})^2}$$

$$= \frac{e^{Z_4}}{\sum_k e^{Z_k}} - \frac{e^{Z_4}}{\sum_k e^{Z_k}} \cdot \frac{e^{Z_4}}{e^{Z_k}}$$

$$= a_j \cdot (1 - a_j)$$

$$\frac{\partial L}{\partial W_{41}} = -\frac{1}{a_j} \cdot a_j \cdot (1 - a_j) = -(1 - a_j)$$

$$\begin{aligned}
 & \left(\frac{e^x}{\sum_k e^{z_k}} \right)' \quad \left(\frac{x}{x+y} \right)' \\
 & = \frac{(e^x)' \cdot \sum_k e^{z_k} + e^x \cdot 0}{(\sum_k e^{z_k})^2} \\
 & \text{if } j \neq i \quad \frac{\partial a_j}{\partial z_i} = \frac{\partial}{\partial z_i} \left(\frac{e^{z_j}}{\sum_k e^{z_k}} \right) \\
 & \text{④} \Rightarrow \text{⑤} \quad = \frac{0 \cdot \sum_k e^{z_k} - e^{z_j} \cdot e^{z_i}}{(\sum_k e^{z_k})^2} \\
 & = -a_j \cdot a_i \\
 & \frac{\partial L}{\partial w_{ki}} = \boxed{a_i} \cdot 0
 \end{aligned}$$

理想的分类器应当是除了真实标签的概率为 1，其余标签概率均为 0。这样计算得到其损失函数为 $-\ln(1) = 0$ 。损失函数越大，说明该分类器在真实标签上分类得越差，性能也就越差。一个非常差的分类器，可能在真实标签上的分类概率接近于 0，那么损失函数就接近于正无穷，我们称为训练发散，需要调小学习速率。在 ImageNet-1000 分类问题中，初始状态为均匀分布，每个类别的分类概率均为 0.001，故此时计算损失函数值为 $-\ln(0.001) = \ln(1000) = 6.907755\dots$ 。经常有同学问，“我的 loss 为什么总是在 6.9 左右（该现象被称为 6.9 高原反应），训练了好久都不下降呢？”说明还都没有训练收敛的迹象，尝试调大学习速率，或者修改权值初始化方式。

<http://blog.csdn.net/u013129427>

1.

关闭

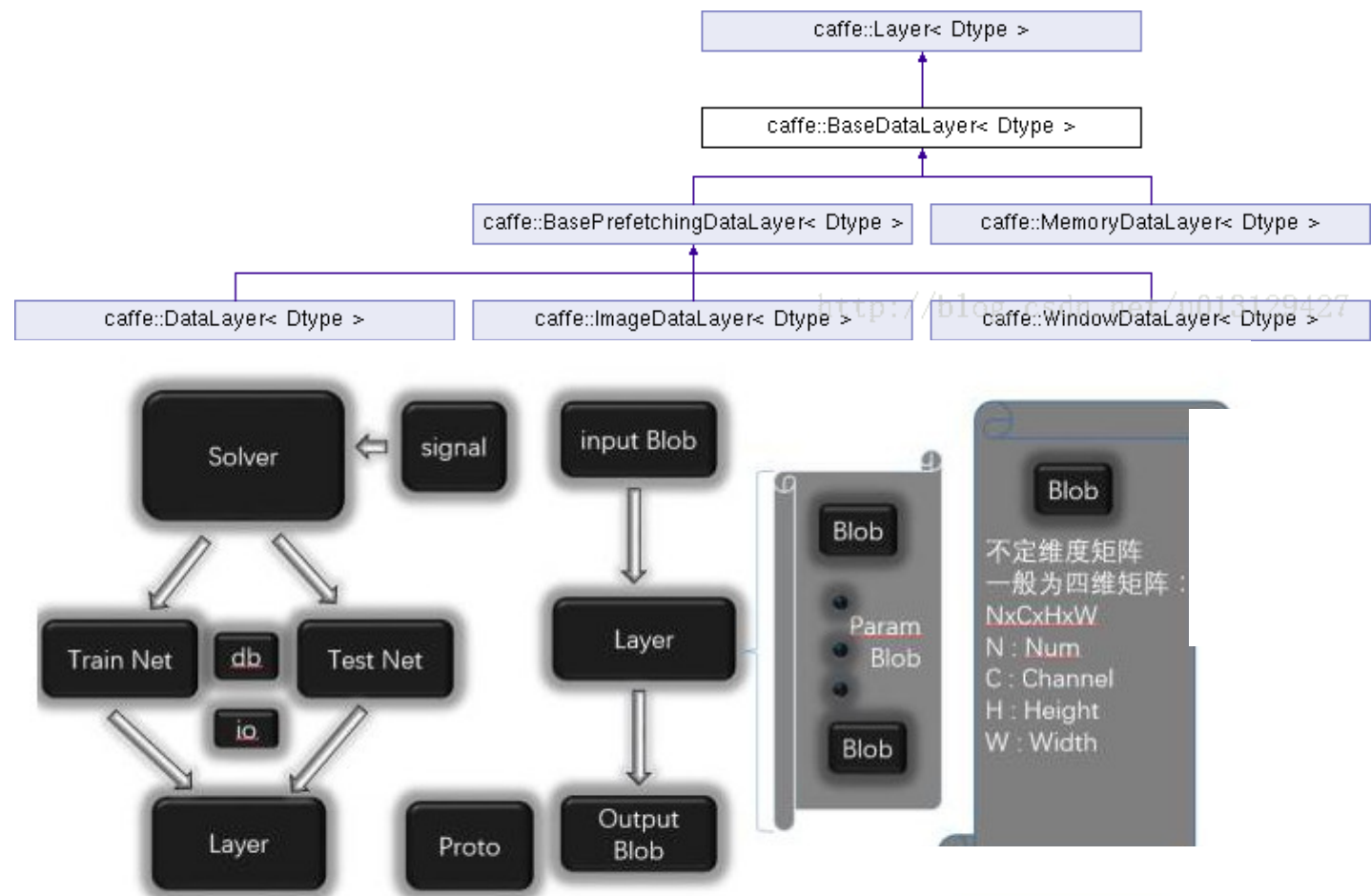


图1. Caffe源码总体架构图

关闭

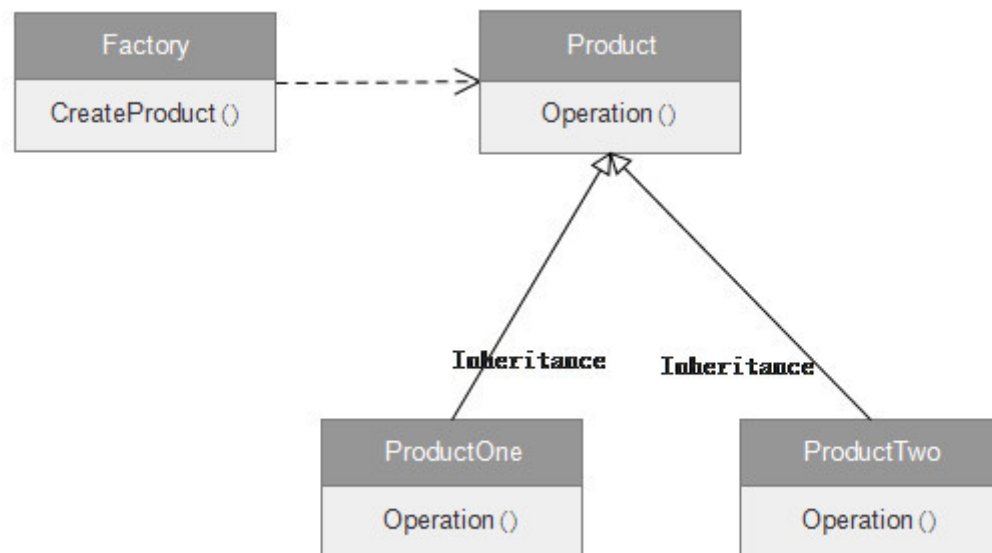


图2. 工厂模式UML类图

Stochastic Gradient Descent (type: "SGD")

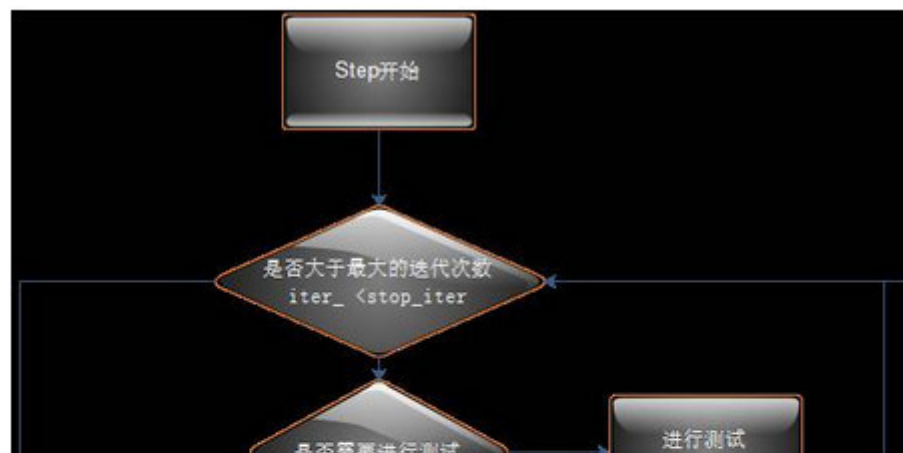
AdaDelta (type: "AdaDelta")

Adaptive Gradient (type: "AdaGrad")

Adam (type: "Adam")

Nesterov's Accelerated Gradient (type: "Nesterov")

RMSprop (type: "RMSProp")



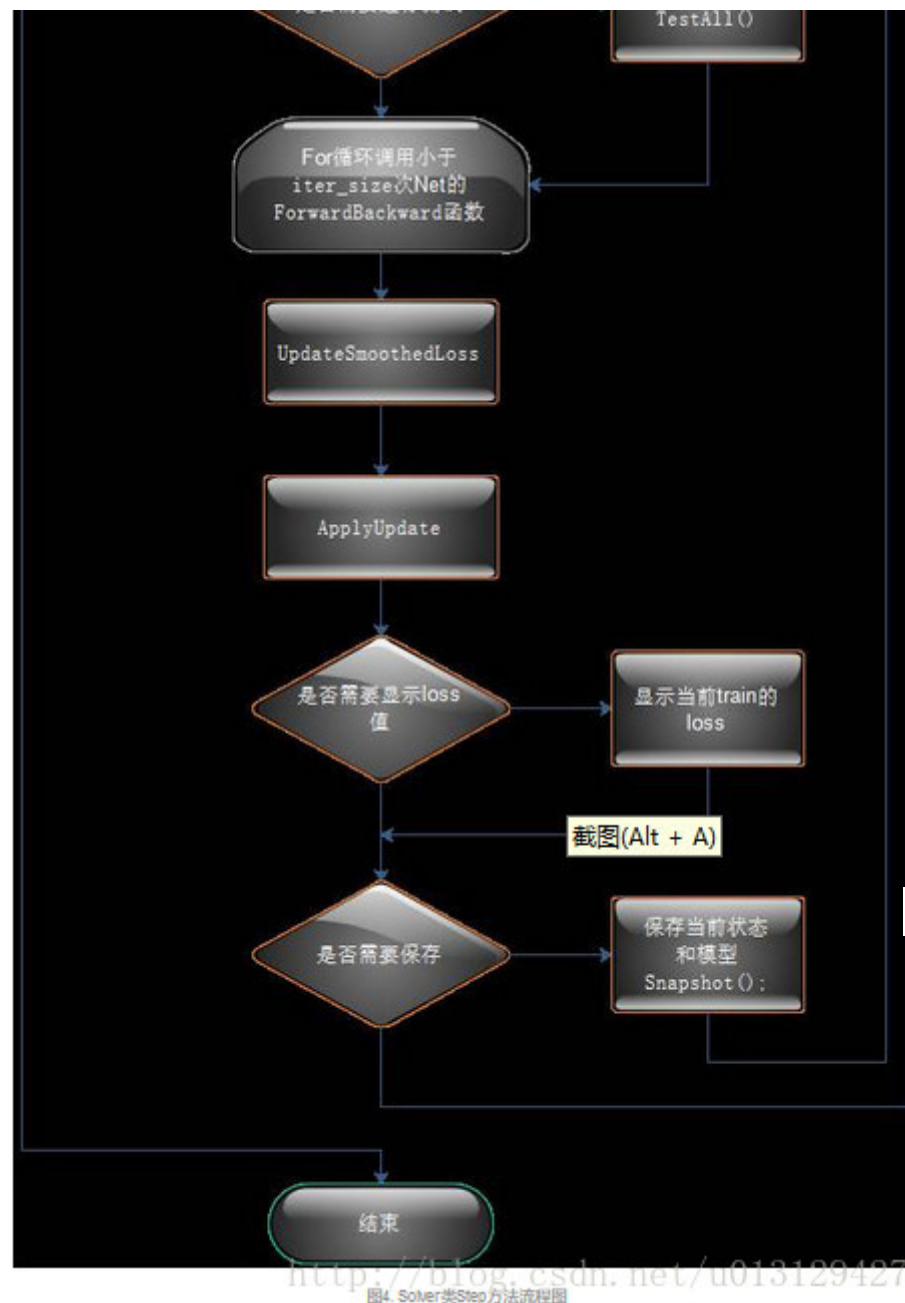


图4. Solver类Step方法流程图

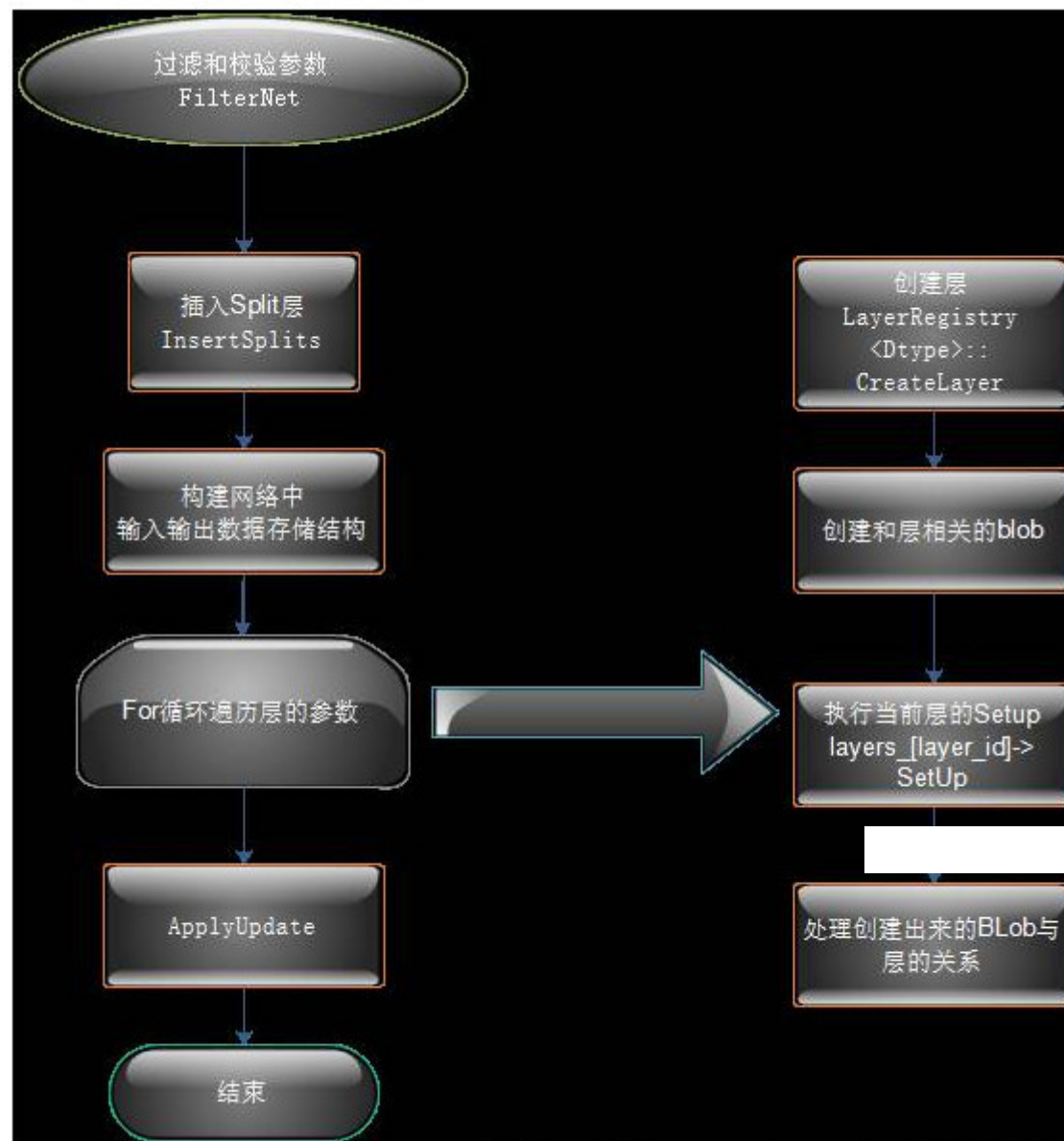
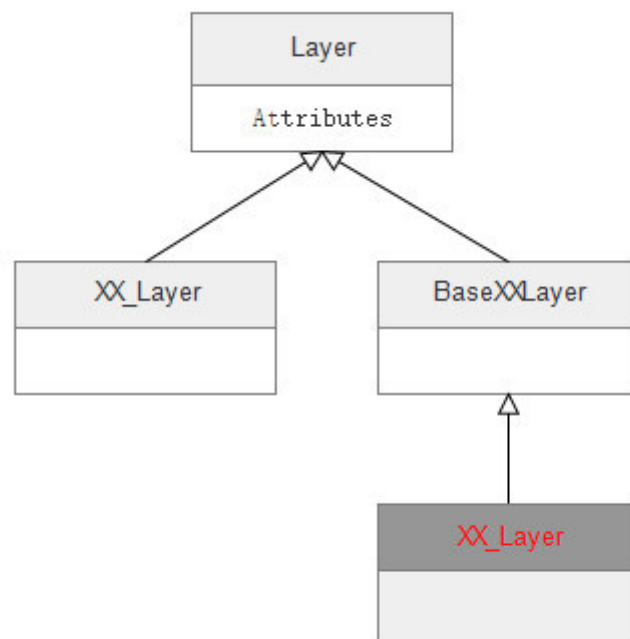


图5. Net类NetInit方法流程图

关闭



<http://blog.csdn.net/u013129427>

图8. Layer层的继承结构

```

116 template <typename Dtype>
117 class LayerRegisterer {
118 public:
119   LayerRegisterer(const string& type,
120                   shared_ptr<Layer<Dtype> > (*creator)(const LayerParameter&)) {
121     // LOG(INFO) << "Registering layer type: " << type;
122     LayerRegistry<Dtype>::AddCreator(type, creator);
123   }
124 };
125
126
127 #define REGISTER_LAYER_CREATOR(type, creator) \
128   static LayerRegisterer<float> g_creator_f_##type(#type, creator<float>); \
129   static LayerRegisterer<double> g_creator_d_##type(#type, creator<double>); \
130
  
```

关闭

```

130
131 #define REGISTER_LAYER_CLASS(type)
132     template <typename Dtype>
133     shared_ptr<Layer<Dtype> > Creator_##type##Layer(const LayerParameter& param)
134     {
135         return shared_ptr<Layer<Dtype> >(new type##Layer<Dtype>(param));
136     }
137 REGISTER_LAYER_CREATOR(type, Creator_##type##Layer)
138

```

```

55
56 INSTANTIATE_CLASS(SplitLayer);
57 REGISTER_LAYER_CLASS(Split);
58

```

```

template <typename Dtype>
shared_ptr<Layer<Dtype> > Creator_ SplitLayer(const LayerParameter& param)
{
    return shared_ptr<Layer<Dtype> >(new SplitLayer<Dtype>(param));
}

```

```

78
79 INSTANTIATE_CLASS(ConvolutionLayer);
80

```

```

36 // Get convolution layer according to engine.
37 template <typename Dtype>
38 shared_ptr<Layer<Dtype> > GetConvolutionLayer(
39     const LayerParameter& param) {
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73 REGISTER_LAYER_CREATOR(Convolution, GetConvolutionLayer);
74

```

关闭

```

*/
void SetUp(const vector<Blob<Dtype>*>& bottom,
           const vector<Blob<Dtype>*>& top) {
    InitMutex();
    CheckBlobCounts(bottom, top);
    LayerSetUp(bottom, top);
    Reshape(bottom, top);
    SetLossWeights(top);
}

```

<http://blog.csdn.net/u013129427>

feature.proto - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```

message Imagefeature{
    required int32 dwFeatSize = 1;
    repeated float pfFeat = 2;
}

```

<http://blog.csdn.net/u013129427>

feature.pb.cc	2016/10/20 10:54	C++ Source	14 KB
feature.pb.h	2016/10/20 10:54	C/C++ Header	7 KB
feature.proto	2016/10/20 10:51	PROTO 文件	1 KB

<http://blog.csdn.net/u013129427>

关闭

(1)c/c++---->常规---->附加包含目录---->

(\$your protobuffer include path)\protobuffer

(2)c/c++---->链接器-->常规-->附加库目录-->

(\$your protobuffer lib path)\protobuffer

(3) c/c++---->链接器-->输入---->附加依赖项-->

libprotobufd.lib;(带d的为debug模式)
或libprotobuf.lib; (不带d,为release模式)

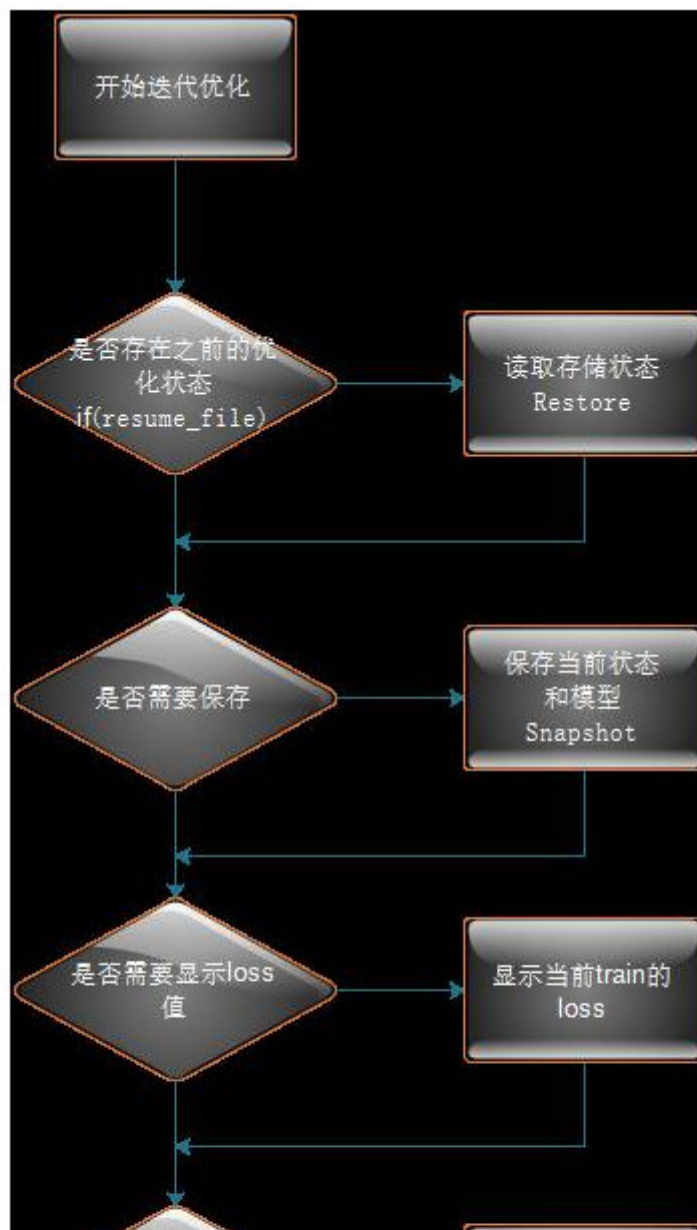
```
std::string write_out_string;
Feature imagefeature;
imagefeature.set_dwfacefeatsize(ffeatsize);
for (int i = 0; i < ffeatsize; i++)
{
    float value = MyCaffeFeat[i];
    imagefeature.add_pffeat(value);
}

std::fstream output(myfile_name, ios::out | ios::binary);
imagefeature.SerializeToString(&write_out_string); //序列化到string*类型write_out_string
中
int64_t length = write_out_string.size();
output.write((char*)&length, sizeof(int64_t));
output.write(write_out_string.data(), length);
output.close();
```

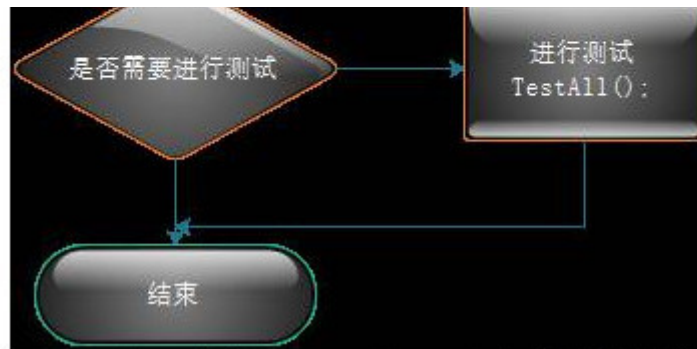
关闭


```
void WriteProtoToBinaryFile(const Message& proto, const char* filename) {  
    fstream output(filename, ios::out | ios::trunc | ios::binary);  
    CHECK(proto.SerializeToOstream(&output));  
}
```

<http://blog.csdn.net/u013129427>



关闭



<http://blog.csdn.net/u013129427>

图3. Solver类Solve方法流程图

顶

0

踩

0

[上一篇](#) [Opencv访问图像像素](#)

关闭

相关文章推荐

- 03crawler02 爬取贴吧排名, 制作图片集
- 用tensorflow训练自己的图片集-用TFRecords将代...
- 【免费】深入理解Docker内部原理及网络配置--王...
- Android入门实战
- 数字图像处理：附录-程序实例、参考文献、标准图..
- CSS+JS实现图片集展示（二）
- SDCC 2017之区块链技术实战线上峰会--蔡栋
- 5天搞定深度学习框架Caffe
- Caffe训练自己的数据之图片集的处理（根据前辈们..
- python检查data图片集和label标签集是否相同
- php零基础到项目实战
- 使用itextsharp创建PDF文档——图片集合

- CSS+JS实现图片集展示
- C语言及程序设计入门指导
- Android 控件之Gallery图片集
- caffe学习系列：训练自己的图片集（超详细教程）


查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 | 江苏乐知信息技术有限公司
京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved 

关闭