[首页] [新闻] [文章]



首页 Web开发 Windows开发 编程语言 数据库技术 移动平台 系统服务 微信 布布扣 其他 数据分析

首页 > 其他 > 详细

caffe 的layer层组织结构

时间: 2015-07-03 12:26:24 阅读: 100 评论: 0 收藏: 0 [点我收藏+]

标签:深度学习 caffe 软件架构

caffe的layer层是靠INSTANTIATE_CLASS和REGISTER_LAYER_CLASS来进行对各个层类进行处理的。每一个layer都是继承基类Layer的,其中BaseDataLayer是输入数据的基类,从这个类继承的主要是:

Layer < Dtype > 🗸 🚄 派生类型 ▶ ★ ArgMaxLayer < Dtype > BaseConvolutionLayer < Dtype > 🔺 🔩 BaseDataLayer<Dtype> ▷ 🔳 基类型 🗸 偏 派生类型 http. BasePrefetchingDataLayer < Dtype > ▷ 🔳 基类型 ▲ 🕌 派生类型 ♪ ★ ImageDataLayer < Dtype > WindowDataLayer<Dtype> b ♣ DummvDataLaver<Dtvne>

这个图可以很好的看到这些layer的继承关系,整个layer层的结构非常清晰,使用抽象工厂的方式构建整个 layer层,然后刚才的宏将layer注册成抽象的服务类,然后再使用的时候再向服务器提供者申请调用各个类, 这样就可以把proto文件作为个各类的配置文件进行调用,这点和java的spring框架很像,使用的是SOA方 式。

当你了解这一结构之后,进行对caffe layer层进行分析的时候相信大多数人会清晰很多,只不过大多数传统的 c++或者java使用这种动态创建类的时候使用的是xml语言进行描述的,而caffe这个东西使用proto这种自定义 的非主流的方式进行了描述。在进行整个caffe剖析的时候要将layer和其他结构独立出来进行分析。(废话:看来多学习学习设计模式等还是非常有用的。)这种设计方式的有点和缺点都很明显,可以让编译型语言在运行时有一部分灵活性,但是依然无法彻底解决灵活性问题,不过对于算法来说这个灵活性够了,也类似 Strategy模式,在选择进行数据变换的时候这个数据可以有多种变换方式。而这种变换方式刚好是一种变换策略,因为网上很多人分析caffe的层,每种层的梯度如何下降,这样下降的优缺点如何太多了,我这里就不班门弄斧了,不说这些被很多人说过的东西,我只说caffe的整个工程架构。以及我们可以如何进行修改来满足我们自己的需求。

这样我们顺便看看caffe的是如何进行结构组织的,io方面使用的opency leveldb lmdb来进行数据读取和存储。Solver这个类是最终执行所有层组合出求解过程的组合。

整个caffe的代码核心部分就在layer里面,每个layer对应一个函数的梯度下降方式,理解这些函数需要数学和传统机器学习的基础,没法短时间速成所以这里我也就不进行胡扯了。这个留作以后慢慢给大家分享,争取每周分享一个层的剖析。

这个就是整个架构的初始化各个类的宏模板函数。

```
#define INSTANTIATE_CLASS(classname) char gInstantiationGuard##classname; template class classname<float>;
 template class classname < double >
这个下面就是典型的SOA方式注册每个class,以及存放这些class的容器,<img src="http://img.blog.csdn.net/20150703111
208565? watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvY3BhcmVudA = = /font/5a6L5L2T/fontsize/400/fill/I0JBQkartermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvY3BhcmVudA = -/font/5a6L5L2T/fontsize/400/fill/I0JBQkartermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvY3BhcmVudA = -/font/5a6L5L2T/fontsize/400/fill/I0JBQkartermark/2/text/aHR0cDovL2T/fontsize/400/fill/I0JBQkartermark/2/text/aHR0cDovL2T/fontsize/400/fill/I0JBQkartermark/2/text/aHR0cDovL2T/fontsize/400/fill/I0JBQkartermark/2/text/aHr0cDovL2T/fontsize/400/fill/I0JBQkartermark/2/text/aHr0cDovL2T/fontsize/400/fill/I0JBQkartermark/2/text/aHr0cDovL2T/fontsize/400/fill/I0JBQkartermark/2/text/aHr0cDovL2T/fontsize/400/fill/I0JBQkartermark/2/text/aHr0cDovL2T/fontsize/400/fill/I0JBQkartermark/2/text/aHr0cDovL2T/fontsize/
FCMA==/dissolve/70/gravity/Center" alt="" /> 当然这是一种简单粗暴的写法。再加上一堆模板和宏搞的整个系统看起来很高
大上,其实只要对c++模板和宏熟悉,这些利器会让你的开发更轻松。
template <typename Dtype>
class LayerRegisterer {
 public:
  LayerRegisterer(const string& type,
                       shared_ptr<Layer<Dtype> > (*creator)(const LayerParameter&)) {
     // LOG(INFO) << "Registering layer type: " << type;
     LayerRegistry < Dtype > :: AddCreator(type, creator);
};
#define REGISTER_LAYER_CREATOR(type, creator)
                                                                                                                                                     static LayerRegisterer<float> g_creator_f_##ty
                                                              static LayerRegisterer<double> g_creator_d_##type(#type, creator<double>)
pe(#type, creator<float>);
#define REGISTER_LAYER_CLASS(type)
                                                                                                                                            template <typename Dtype>
                         shared_ptr<Layer<Dtype> > Creator_##type##Layer(const LayerParameter& param) {
                                                               return shared_ptr<Layer<Dtype> >(new type##Layer<Dtype>(param));
                                                                                     REGISTER\_LAYER\_CREATOR(type, Creator\_\#\#type\#\#Layer)
} // namespace caffe
看完这些相信大家对caffe的源码分析定位可以快速入手。
版权声明:本文为博主原创文章,未经博主允许不得转载。
caffe 的layer层组织结构
标签:深度学习 caffe 软件架构
                                                                                                                                                                                                                                       踩
                                                                                                                                                                                                           (0)
                                                                                                                                                                                                                                        (0)
        评论
                                ─句话评论(0)
                                                                                                                    共0条
```

登录后才能评论!

登录