

Caffe源码解析1: Blob

转载请注明出处，楼燚(yì)航的blog，<http://www.cnblogs.com/louyihang-loves-baiyan/>

首先看到的是Blob这个类，Blob是作为Caffe中数据流通的一个基本类，网络各层之间的数据是通过Blob来传递的。这里整个代码是非常规范的，基本上条件编译，命名空间，模板类，各种不太经常看到的关键字如explicit,inline等等。

首先提一下explicit关键字的作用是禁止单参数构造函数的隐式转换，具体含义谷歌即可。还有inline的作用，inline主要是将代码进行复制，扩充，会使代码总量上升，好处就是可以节省调用的开销，能提高执行效率。

1主要变量

```
shared_ptr<SyncedMemory> data_;
shared_ptr<SyncedMemory> diff_;
shared_ptr<SyncedMemory> shape_data_;
vector<int> shape_;
int count_;
int capacity_;
```

BLOB只是一个基本的数据结构，因此内部的变量相对较少，首先是data_ 指针，指针类型是shared_ptr，属于boost库的一个智能指针，这一部分主要用来申请内存存储data，data主要是正向传播的时候用的。同理，diff_ 主要用来存储偏差，update data，shape_data_ 和 shape_ 都是存储Blob的形状，一个是老版本一个是新版本。count_ 表示Blob中的元素个数，也就是 个数*通道数*高度*宽度，capacity_ 表示当前的元素个数，因为Blob可能会reshape。

2主要函数

```
template <typename Dtype>
class Blob {
public:
    Blob()
        : data_(), diff_(), count_(0), capacity_(0) {}

    /// @brief Deprecated; use <code>Blob(const vector<int>& shape)</code>.
    explicit Blob(const int num, const int channels, const int height,
        const int width);
    explicit Blob(const vector<int>& shape);

    /// @brief Deprecated; use <code>Reshape(const vector<int>& shape)</code>.
    void Reshape(const int num, const int channels, const int height,
        const int width);
```

其中Blob作为一个最基础的类，其中构造函数开辟一个内存空间来存储数据，Reshape 函数在Layer中的reshape或者forward操作中来adjust dimension。同时在改变Blob大小时，内存将会被重新分配如果内存大小不够了，并且额外的内存将不会被释放。对input的blob进行reshape,如果立马调用 Net::Backward 是会出错的，因为reshape之后，要么 Net::forward 或者 Net::Reshape 就会被调用来将新的input shape 传播到高层

Blob类里面有重载很多个 count() 函数，主要还是为了统计Blob的容量（volume），或者是某一片（slice），从某个axis到具体某个axis的shape乘积。

```
inline int count(int start_axis, int end_axis)
```

并且Blob的Index是可以从负坐标开始读的，这一点跟Python好像

```
inline int CanonicalAxisIndex(int axis_index)
```

对于Blob中的4个基本变量 num, channel, height, width 可以直接通过 shape(0), shape(1), shape(2), shape(3) 来访问。

公告

昵称：楼燚航的blog

园龄：1年5个月

粉丝：60

关注：1

+加关注

< 2016年4月 >						
日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

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

更多链接

随笔档案

2016年3月(2)

2016年2月(2)

2016年1月(7)

2015年12月(2)

2015年11月(5)

2015年10月(5)

2015年9月(1)

2015年8月(2)

2015年7月(1)

2015年6月(1)

2015年5月(1)

2015年4月(3)

2014年12月(1)

最新评论

1. Re:opencv 3.0 DPM Cascade 检测（附带TBB和openMP加速）

好了，问题解决了，我用的opencv版本是2.412，所以导致那么慢，换成310就没有这问题了

计算offset

```
inline int offset(const int n, const int c = 0, const int h = 0, const int w = 0)
inline int offset(const vector<int>& indices)
```

offset计算的方式也支持两种方式，一种直接指定n,c,h,w或者放到一个vector中进行计算，偏差是根据对应的n,c,h,w，返回的offset是 $((n * channels() + c) * height() + h) * width() + w$

其实里面稍加留意可以看到有很多的

```
CHECK_GE
CHECK_LE
CHECK_EQ
....
```

等等看意思就知道了，肯定是在做比较Geater or Equal这样的意思。这其实是GLOG，谷歌的一个日志库，Caffe里面用用了大量这样的宏，看起来也比较直观

```
void CopyFrom(const Blob<Dtype>& source, bool copy_diff = false, bool reshape = false);
```

从一个blob中copy数据，通过开关控制是否copy_diff,如果是False则copy data。reshape控制是否需要reshape。好我们接着往下看

```
inline Dtype data_at(const int n, const int c, const int h, const int w)
inline Dtype diff_at(const int n, const int c, const int h, const int w)
inline Dtype data_at(const vector<int>& index)
inline Dtype diff_at(const vector<int>& index)
inline const shared_ptr<SyncedMemory>& data()
inline const shared_ptr<SyncedMemory>& diff()
```

这一部分函数主要通过给定的位置访问数据，根据位置计算与数据起始的偏差offset，在通过cpu_data*指针获得地址。下面几个函数都是获得

```
const Dtype* cpu_data() const;
void set_cpu_data(Dtype* data);
const int* gpu_shape() const;
const Dtype* gpu_data() const;
const Dtype* cpu_diff() const;
const Dtype* gpu_diff() const;
Dtype* mutable_cpu_data();
Dtype* mutable_gpu_data();
Dtype* mutable_cpu_diff();
Dtype* mutable_gpu_diff();
```

可以看到这里有data和diff两类数据，而这个diff就是我们所熟知的偏差，前者主要存储前向传递的数据，而后者存储的是反向传播中的梯度

```
void Update();
```

看到update里面调用了

```
caffe_axpy<float>(const int N, const float alpha, const float* X, float* Y)
{ cblas_saxpy(N, alpha, X, 1, Y, 1); }
```

这个函数在caffe的util下面的match-functions.cpp里面，主要是负责了线性代数库的调用，实现的功能是

$$Y = \alpha * X + \beta * Y$$

也就是blob里面的data部分减去diff部分

```
void FromProto(const BlobProto& proto, bool reshape = true);
void ToProto(BlobProto* proto, bool write_diff = false) const;
```

这两个函数主要是将数据序列化，存储到BlobProto，这里说到Proto是谷歌的一个数据序列化的存储格式，可以实现语言、平台无关、可扩展的序列化结构数据格式。Caffe里面数据的存储都采用这一结构，这里就不深入展开，具体可以参照这篇文章，对于proto的序列化和反序列都讲解的非常详细

<http://www.w2bc.com/Article/34963>

```
Dtype asum_data() const; //计算data的L1范数
Dtype asum_diff() const; //计算diff的L1范数
Dtype sumsq_data() const; //计算data的L2范数
Dtype sumsq_diff() const; //计算diff的L2范数
void scale_data(Dtype scale_factor); //将data部分乘以一个因子
void scale_diff(Dtype scale_factor); //将diff部分乘以一个因子
```

这几个函数是一些零散的功能，一看就懂。

--gaosi123

2. Re:opencv 3.0 DPM Cascade 检测 (附带TBB和openMP加速)

BTW, 我电脑是i7 4790k + 16GB内存，所以硬件设备应该不会是限制。不知道问题出在哪里

--gaosi123

3. Re:opencv 3.0 DPM Cascade 检测 (附带TBB和openMP加速)

如果可以，欢迎留个email

--gaosi123

4. Re:opencv 3.0 DPM Cascade 检测 (附带TBB和openMP加速)

你好，我也是直接把DPM代码拷贝到工程里，但是想你这样直接拷进去不会报错吗？我直接拷贝进去按照你的来，报错信息如下：Error 4 error C2039: 'dpm': is not a memb.....

--gaosi123

5. Re:Fast RCNN 训练自己数据集 (2修改数据读取接口)

@楼臻航的blog楼主你好！我在EdgeBoxes提取OP的时候也是直接用的默认参数，并且将坐标[x y w h]变成了左上右下的形式，但是发现检测车的时候效果并没有Selective Search好，.....

—JustJay

阅读排行榜

1. Fast RCNN 训练自己数据集 (2修改数据读取接口)(3946)
2. Fast RCNN 训练自己数据集 (1编译配置)(3166)
3. Fast RCNN 训练自己的数据集 (3训练和检测)(3097)
4. RCNN (Regions with CNN) 目标物检测 Fast RCNN的基础(2096)
5. Hog SVM 车辆 行人检测(979)


评论排行榜

1. Fast RCNN 训练自己数据集 (2修改数据读取接口)(22)
2. Fast RCNN 训练自己数据集 (1编译配置)(21)
3. Fast RCNN 训练自己的数据集 (3训练和检测)(5)
4. opencv 3.0 DPM Cascade 检测 (附带TBB和openMP加速)(4)
5. DPM检测模型 训练自己的数据集 读取接口修改(2)

推荐排行榜

1. Fast RCNN 训练自己数据集 (2修改数据读取接口)(5)
2. 车脸检测 Adaboost 检测过程(3)
3. Caffe 抽取CNN网络特征 Python(2)
4. DPM检测模型 训练自己的数据集 读取接口修改(2)
5. RCNN (Regions with CNN) 目标物检测 Fast RCNN的基础(2)

```
void ShareData(const Blob& other);  
void ShareData(const Blob& other);
```

这两个函数看名字就知道了一个是共享data，一个是共享diff，具体就是将别的blob的data和响应的diff指针给这个Blob，实现数据的共享。同时需要注意的是这个操作会引起这个Blob里面的SyncedMemory被释放，因为shared_ptr指针被用  重置的时候回调响应的析构器。

```
bool ShapeEquals(const BlobProto& other);
```

这函数就不用说了，比较两个Blob形状是否相同

好了，基本上Blob的主要参数功能基本就涵盖在里面了，以上只是我的拙见，如有纰漏，还望指出，万分感谢。



楼继航的blog

关注 - 1

粉丝 - 60

+加关注

0

0

(请您对文章做出评价)

« 上一篇: [梯度下降、随机梯度下降和批量梯度下降](#)

» 下一篇: [Caffe源码解析2: SyncedMem](#)

posted @ 2016-01-21 21:24 楼继航的blog 阅读(692) 评论(0) 编辑 收藏

[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】融云即时通讯云—豆果美食、Faceu等亿级APP都在用

【推荐】百度开放云—三月超低价促销



最新IT新闻:

- 女性玩家崛起 研发女性游戏要注意什么
- 投资人才! Google将免费培训100万非洲年轻人
- 三次改变世界后, 硅谷奇才Sean Parker的下一个使命是攻克癌症难题
- 微软甲骨文亚马逊们都热抢的SaaS服务, 也要面临选择分歧啦
- Medium和Twitter创始人将社交媒体形容为垃圾食品

» 更多新闻...



最新知识库文章:

- 我是一个线程
- 为什么未来是全栈工程师的世界?
- 程序bug导致了天大的损失, 要枪毙程序猿吗?
- 如何运维千台以上游戏云服务器
- 架构漫谈 (一): 什么是架构?

» 更多知识库文章...