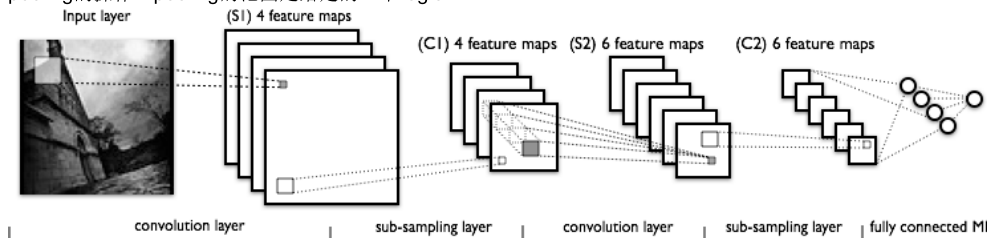


Caffe源码解析7: Pooling_Layer

转载请注明出处, 楼焱(yi)航的blog, <http://home.cnblogs.com/louyihang-loves-baiyan/>

Pooling 层一般在网络中是跟在Conv卷积层之后, 做采样操作, 其实是为了进一步缩小feature map, 同时也能增大神经元的视野。在Caffe中, pooling层属于vision_layer的一部分, 其相关的定义也在vision_layer.hpp的头文件中。Pooling层的相关操作比较少, 在Caffe的自带模式下只有Max pooling和Average pooling两种

下图是一个LeNet的网络结构图, 全连接之前主要有2个卷积层, 2个池化层, 其中sub_sampling layer就是pooling的操作。pooling的范围是给定的一个region。



PoolingLayer

caffe中Pooling的操作相对比较少, 结构也简单, 首先看它的Forward_cpu函数, 在forward的时候根据相应的Pooling_method选择相应的pooling方法

forward_cpu

```
void PoolingLayer<Dtype>::Forward_cpu(const vector<Blob<Dtype>*>& bottom,
    const vector<Blob<Dtype>*>& top) {
    const Dtype* bottom_data = bottom[0]->cpu_data();
    Dtype* top_data = top[0]->mutable_cpu_data();
    const int top_count = top[0]->count();
    //将mask信息输出到top[1],如果top大于1
    const bool use_top_mask = top.size() > 1;
    int* mask = NULL; // suppress warnings about uninitialized variables
    Dtype* top_mask = NULL;
    switch (this->layer_param_.pooling_param().pool()) {
    case PoolingParameter_PoolMethod_MAX://这里的case主要是实现max pooling的方法
        // Initialize
        if (use_top_mask) {
            top_mask = top[1]->mutable_cpu_data();
            caffe_set(top_count, Dtype(-1), top_mask);
        } else {
            mask = max_idx_.mutable_cpu_data();
            caffe_set(top_count, -1, mask);
        }
        caffe_set(top_count, Dtype(-FLT_MAX), top_data);
        // The main loop
        for (int n = 0; n < bottom[0]->num(); ++n) {
            for (int c = 0; c < channels_; ++c) {
                for (int ph = 0; ph < pooled_height_; ++ph) {
                    for (int pw = 0; pw < pooled_width_; ++pw) {
                        int hstart = ph * stride_h_ - pad_h_;//这里的hstart, wstart,hend,wend指的是pooling窗口
                        //在特征图中的坐标, 对应左上右下即x1 y1 x2 y2
                        int wstart = pw * stride_w_ - pad_w_;
                        int hend = min(hstart + kernel_h_, height_);
                        int wend = min(wstart + kernel_w_, width_);
                        hstart = max(hstart, 0);
                        wstart = max(wstart, 0);
                        const int pool_index = ph * pooled_width_ + pw;
                        for (int h = hstart; h < hend; ++h) {
                            for (int w = wstart; w < wend; ++w) {
                                const int index = h * width_ + w;//记录index偏差
                                if (bottom_data[index] > top_data[pool_index]) { //不停迭代
```

公告

昵称: 楼焱航的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

```

        top_data[pool_index] = bottom_data[index];
        if (use_top_mask) {
            top_mask[pool_index] = static_cast<Dtype>(index); //记录当前最大值的坐标索引
        } else {
            mask[pool_index] = index;
        }
    }
}
}
}
}
// 计算偏移量, 进入下一张图的index起始地址
bottom_data += bottom[0]->offset(0, 1);
top_data += top[0]->offset(0, 1);
if (use_top_mask) {
    top_mask += top[0]->offset(0, 1);
} else {
    mask += top[0]->offset(0, 1);
}
}
}
break;
case PoolingParameter_PoolMethod_AVE://average_pooling
for (int i = 0; i < top_count; ++i) {
    top_data[i] = 0;
}
// The main loop
for (int n = 0; n < bottom[0]->num(); ++n) { //同样是主循环
    for (int c = 0; c < channels_; ++c) {
        for (int ph = 0; ph < pooled_height_; ++ph) {
            for (int pw = 0; pw < pooled_width_; ++pw) {
                int hstart = ph * stride_h_ - pad_h_;
                int wstart = pw * stride_w_ - pad_w_;
                int hend = min(hstart + kernel_h_, height_ + pad_h_);
                int wend = min(wstart + kernel_w_, width_ + pad_w_);
                int pool_size = (hend - hstart) * (wend - wstart);
                hstart = max(hstart, 0);
                wstart = max(wstart, 0);
                hend = min(hend, height_);
                wend = min(wend, width_);
                for (int h = hstart; h < hend; ++h) {
                    for (int w = wstart; w < wend; ++w) {
                        top_data[ph * pooled_width_ + pw] +=
                            bottom_data[h * width_ + w];
                    }
                }
                top_data[ph * pooled_width_ + pw] /= pool_size; //获得相应的平均值
            }
        }
        // compute offset同理计算下一个图的起始地址
        bottom_data += bottom[0]->offset(0, 1);
        top_data += top[0]->offset(0, 1);
    }
}
break;
case PoolingParameter_PoolMethod_STOCHASTIC:
    NOT_IMPLEMENTED;
    break;
default:
    LOG(FATAL) << "Unknown pooling method.";
}
}

```

backward_cpu

对于误差的反向传导

对于pooling层的误差传到, 根据下式

$$\delta_j^l = \text{upsample}(\delta_j^{l+1}) \cdot h(a_j^l)'$$

这里的Upsample具体可以根据相应的pooling方法来进行上采样, upsample的基本思想也是将误差进行的平摊到各个采样的对应点上。在这里pooling因为是线性的所以h这一项其实是可以省略的。

具体的计算推导过程请结合<http://www.cnblogs.com/tornadomeet/p/3468450.html>有详细的推导过程, 结合代码中主循环中的最里项会更清晰的明白

```

template <typename Dtype>
void PoolingLayer<Dtype>::Backward_cpu(const vector<Blob<Dtype>*>& top,
    const vector<bool>& propagate_down, const vector<Blob<Dtype>*>& bottom) {
    if (!propagate_down[0]) {
        return;
    }
}

```

--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)

@楼焱航的blog楼主你好! 我在EdgeBoxe s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)

```

}
const Dtype* top_diff = top[0]->cpu_diff();//首先获得上层top_blob的diff
Dtype* bottom_diff = bottom[0]->mutable_cpu_diff();
caffe_set(bottom[0]->count(), Dtype(0), bottom_diff);
// We'll output the mask to top[1] if it's of size >1.
const bool use_top_mask = top.size() > 1;
const int* mask = NULL; // suppress warnings about uninitialized variables
const Dtype* top_mask = NULL;
switch (this->layer_param_.pooling_param().pool()) {
case PoolingParameter_PoolMethod_MAX:
    // The main loop
    if (use_top_mask) {
        top_mask = top[1]->cpu_data();
    } else {
        mask = max_idx_.cpu_data();
    }
    for (int n = 0; n < top[0]->num(); ++n) {
        for (int c = 0; c < channels_; ++c) {
            for (int ph = 0; ph < pooled_height_; ++ph) {
                for (int pw = 0; pw < pooled_width_; ++pw) {
                    const int index = ph * pooled_width_ + pw;
                    const int bottom_index =
                        use_top_mask ? top_mask[index] : mask[index]; //根据max pooling记录的mask位置, 进行
误差反转
                    bottom_diff[bottom_index] += top_diff[index];
                }
            }
            bottom_diff += bottom[0]->offset(0, 1);
            top_diff += top[0]->offset(0, 1);
            if (use_top_mask) {
                top_mask += top[0]->offset(0, 1);
            } else {
                mask += top[0]->offset(0, 1);
            }
        }
    }
    break;
case PoolingParameter_PoolMethod_AVE:
    // The main loop
    for (int n = 0; n < top[0]->num(); ++n) {
        for (int c = 0; c < channels_; ++c) {
            for (int ph = 0; ph < pooled_height_; ++ph) {
                for (int pw = 0; pw < pooled_width_; ++pw) {
                    int hstart = ph * stride_h_ - pad_h_;
                    int wstart = pw * stride_w_ - pad_w_;
                    int hend = min(hstart + kernel_h_, height_ + pad_h_);
                    int wend = min(wstart + kernel_w_, width_ + pad_w_);
                    int pool_size = (hend - hstart) * (wend - wstart);
                    hstart = max(hstart, 0);
                    wstart = max(wstart, 0);
                    hend = min(hend, height_);
                    wend = min(wend, width_);
                    for (int h = hstart; h < hend; ++h) {
                        for (int w = wstart; w < wend; ++w) {
                            bottom_diff[h * width_ + w] +=
                                top_diff[ph * pooled_width_ + pw] / pool_size; //mean_pooling中, bottom的误差值按
pooling窗口中的大小计算, 从上一层进行填充后, 再除窗口大小
                        }
                    }
                }
            }
            // offset
            bottom_diff += bottom[0]->offset(0, 1);
            top_diff += top[0]->offset(0, 1);
        }
    }
    break;
case PoolingParameter_PoolMethod_STOCHASTIC:
    NOT_IMPLEMENTED;
    break;
default:
    LOG(FATAL) << "Unknown pooling method.";
}
}

```



楼臻航的blog

关注 - 1

粉丝 - 60

[+加关注](#)

« 上一篇: [Caffe源码解析6: Neuron Layer](#)

» 下一篇: [Caffe 单独测试添加的layer](#)

posted @ 2016-02-23 21:31 楼臻航的blog 阅读(487) 评论(0) 编辑 收藏

(请您对文章做出评价)

[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:

- LG确认：开发Friends模块设备需要取得授权并协同开发
 - 触角越来越广 华为能成为中国的三星吗？
 - 微软认知服务：人工智能的技术拼图
 - 知己知彼，百战不殆：一篇文章看懂隐藏在阿尔法狗背后的深度学习
 - 女性玩家崛起 研发女性游戏要注意什么
- » 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，要枪毙程序员猿吗？
 - 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...