

Caffe源码解析6: Neuron_Layer

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

NeuronLayer，顾名思义这里就是神经元，激活函数的相应层。我们知道在blob进入激活函数之前和之后他的size是不会变的，而且激活值也就是输出 y 只依赖于相应的输入 x 。在Caffe里面所有的layer的实现都放在src文件夹下的layer文件夹中，基本上很多文章里应用到的layer类型它都有cpu和cuda的实现。在caffe里面NeuronLayer比较多，在此罗列了一下

- AbsValLayer
- BNLLayer
- DropoutLayer
- ExpLayer
- LogLayer
- PowerLayer
- ReLULayer
- CuDNNReLULayer
- SigmoidLayer
- CuDNNSigmoidLayer
- TanHLayer
- CuDNNTanHLayer
- ThresholdLayer
- PReLULayer

Caffe里面的Neuron种类比较多方便人们使用，这里我们着重关注几个主要的Neuro_layer

ReLULayer

目前在激活层的函数中使用ReLU是非常普遍的，一般我们在看资料或者讲义中总是提到的是Sigmoid函数，它比Sigmoid有更快的收敛性，因为sigmoid在收敛的时候越靠近目标点收敛的速度会越慢，也是其函数的曲线形状决定的。而ReLU则相对收敛更快，具体可以看Krizhevsky 12年的那篇ImageNet CNN文章有更详细的介绍。

其计算的公式是：

$$y = \max(0, x)$$

如果有负斜率式子变为：

$$y = \max(0, x) + \nu \min(0, x)$$

反向传播的公式

$$\frac{\partial E}{\partial x} = \begin{cases} \nu \frac{\partial E}{\partial y} & \text{if } x \leq 0 \\ \frac{\partial E}{\partial y} & \text{if } x > 0 \end{cases}$$

公告

昵称：楼綵航的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

其在caffe中的forward和backward函数为

```
template <typename Dtype>
void ReLULayer<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 count = bottom[0]->count();
    Dtype negative_slope = this->layer_param_.relu_param().negative_slope();
    for (int i = 0; i < count; ++i) {
        top_data[i] = std::max(bottom_data[i], Dtype(0))
            + negative_slope * std::min(bottom_data[i], Dtype(0));
    }
}

template <typename Dtype>
void ReLULayer<Dtype>::Backward_cpu(const vector<Blob<Dtype>*>& top,
    const vector<bool>& propagate_down,
    const vector<Blob<Dtype>*>& bottom) {
    if (propagate_down[0]) {
        const Dtype* bottom_data = bottom[0]->cpu_data();
        const Dtype* top_diff = top[0]->cpu_diff();
        Dtype* bottom_diff = bottom[0]->mutable_cpu_diff();
        const int count = bottom[0]->count();
        Dtype negative_slope = this->layer_param_.relu_param().negative_slope();
        for (int i = 0; i < count; ++i) {
            bottom_diff[i] = top_diff[i] * ((bottom_data[i] > 0)
                + negative_slope * (bottom_data[i] <= 0));
        }
    }
}
```

SigmoidLayer

Sigmoid函数，也称为阶跃函数，函数曲线是一个优美的S形。目前使用Sigmoid函数已经不多见了，大多使用ReLU来代替，其对应的激活函数为：

$$y = (1 + \exp(-x))^{-1}$$

其反向传播时

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial y} y(1-y)$$

其相应的forward和backward的函数为

```
template <typename Dtype>
void SigmoidLayer<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 count = bottom[0]->count();
    for (int i = 0; i < count; ++i) {
        top_data[i] = sigmoid(bottom_data[i]);
    }
}

template <typename Dtype>
void SigmoidLayer<Dtype>::Backward_cpu(const vector<Blob<Dtype>*>& top,
    const vector<bool>& propagate_down,
    const vector<Blob<Dtype>*>& bottom) {
    if (propagate_down[0]) {
        const Dtype* top_data = top[0]->cpu_data();
        const Dtype* top_diff = top[0]->cpu_diff();
        Dtype* bottom_diff = bottom[0]->mutable_cpu_diff();
        const int count = bottom[0]->count();
        for (int i = 0; i < count; ++i) {
            const Dtype sigmoid_x = top_data[i];
            bottom_diff[i] = top_diff[i] * sigmoid_x * (1. - sigmoid_x);
        }
    }
}
```

DropoutLayer

DropoutLayer现在是非常常用的一种网络层，只用在训练阶段，一般用在网络的全连接层中，可以减少网络的过拟合问题。其思想是在训练过程中随机的将一部分输入x之置为0。

--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)

$$y_{\text{train}} = \begin{cases} \frac{x}{1-p} & \text{if } u > p \\ 0 & \text{otherwise} \end{cases}$$

其forward_cpu和backward_cpu为:

```
template <typename Dtype>
void DropoutLayer<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();
    unsigned int* mask = rand_vec_.mutable_cpu_data();
    const int count = bottom[0]->count();
    if (this->phase_ == TRAIN) {
        // Create random numbers构造随机数, 这里是通过向量掩码来和bottom的数据相乘, scale_是控制undropped的比例
        caffe_rng_bernoulli(count, 1. - threshold_, mask);
        for (int i = 0; i < count; ++i) {
            top_data[i] = bottom_data[i] * mask[i] * scale_;
        }
    } else {
        caffe_copy(bottom[0]->count(), bottom_data, top_data);
    }
}

template <typename Dtype>
void DropoutLayer<Dtype>::Backward_cpu(const vector<Blob<Dtype>*>& top,
    const vector<bool>& propagate_down,
    const vector<Blob<Dtype>*>& bottom) {
    if (propagate_down[0]) {
        const Dtype* top_diff = top[0]->cpu_diff();
        Dtype* bottom_diff = bottom[0]->mutable_cpu_diff();
        if (this->phase_ == TRAIN) {
            const unsigned int* mask = rand_vec_.cpu_data();
            const int count = bottom[0]->count();
            for (int i = 0; i < count; ++i) {
                bottom_diff[i] = top_diff[i] * mask[i] * scale_;
            }
        } else {
            caffe_copy(top[0]->count(), top_diff, bottom_diff);
        }
    }
}
```

好文要顶

关注我

收藏该文



楼楚航的blog

关注 - 1

粉丝 - 60

+加关注

0

0

(请您对文章做出评价)

« 上一篇: [Caffe源码解析5: Conv_Layer](#)
» 下一篇: [Caffe源码解析7: Pooling_Layer](#)

posted @ 2016-02-19 14:16 楼楚航的blog 阅读(344) 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



纯前端表格控件

流畅操作海量数据

了解SpreadJS

- **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...