

Tensorflow反卷积（DeConv）实现原理+手写python代码实现反卷积（DeConv）



huachao1001 (/u/0a7e42698e4b) [+ 关注](#)

2018.01.22 19:46 字数 691 阅读 1958 评论 6 喜欢 5

(/u/0a7e42698e4b)

上一篇文章已经介绍过卷积的实现，这篇文章我们学习反卷积原理，同样，在了解反卷积原理后，在后面手写python代码实现反卷积。

1 反卷积原理

反卷积原理不太好用文字描述，这里直接以一个简单例子描述反卷积过程。

假设输入如下：

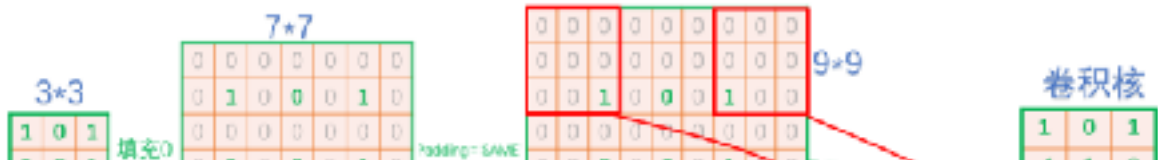
```
[[1,0,1],  
 [0,2,1],  
 [1,1,0]]
```

反卷积卷积核如下：

```
[[ 1, 0, 1],  
 [-1, 1, 0],  
 [ 0,-1, 0]]
```

现在通过 `stride=2` 来进行反卷积，使得尺寸由原来的 3×3 变为 6×6 。那么在Tensorflow框架中，反卷积的过程如下(不同框架在裁剪这步可能不一样)：





反卷积实现例子

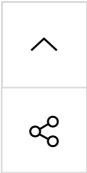
(/apps/
utm_sc
banner

其实通过我绘制的这张图，就已经把原理讲的很清楚了。大致步奏就是，先填充0，然后进行卷积，卷积过程跟上一篇文章讲述的一致。最后一步还要进行裁剪。好了，原理讲完了，(##)....

2 代码实现

上一篇文章我们只针对了输出通道数为1进行代码实现，在这篇文章中，反卷积我们将输出通道设置为多个，这样更符合实际场景。

先定义输入和卷积核：



```

input_data=[
    [[1,0,1],
     [0,2,1],
     [1,1,0]],

    [[2,0,2],
     [0,1,0],
     [1,0,0]],

    [[1,1,1],
     [2,2,0],
     [1,1,1]],

    [[1,1,2],
     [1,0,1],
     [0,2,2]]

]
weights_data=[
    [[[ 1, 0, 1],
      [-1, 1, 0],
      [ 0,-1, 0]],
     [[-1, 0, 1],
      [ 0, 0, 1],
      [ 1, 1, 1]],
     [[ 0, 1, 1],
      [ 2, 0, 1],
      [ 1, 2, 1]],
     [[ 1, 1, 1],
      [ 0, 2, 1],
      [ 1, 0, 1]]],

    [[[ 1, 0, 2],
      [-2, 1, 1],
      [ 1,-1, 0]],
     [[-1, 0, 1],
      [-1, 2, 1],
      [ 1, 1, 1]],
     [[ 0, 0, 0],
      [ 2, 2, 1],
      [ 1,-1, 1]],
     [[ 2, 1, 1],
      [ 0,-1, 1],
      [ 1, 1, 1]]]
]

```

(/apps/
utm_sc
banner

上面定义的输入和卷积核，在接下的运算过程如下图所示：



执行过程

可以看到实际上，反卷积和卷积基本一致，差别在于，反卷积需要填充过程，并在最后一步需要裁剪。具体实现代码如下：

(/apps/
utm_sc
banner



(/apps/
utm_sc
banner

```
# 根据输入map([h,w]) 和卷积核([k,k]), 计算卷积后的feature map
import numpy as np
def compute_conv(fm, kernel):
    [h,w]=fm.shape
    [k,_]=kernel.shape
    r=int(k/2)
    # 定义边界填充0后的map
    padding_fm=np.zeros([h+2,w+2],np.float32)
    # 保存计算结果
    rs=np.zeros([h,w],np.float32)
    # 将输入在指定该区域赋值, 即除了4个边界后, 剩下的区域
    padding_fm[1:h+1,1:w+1]=fm
    # 对每个点为中心的区域遍历
    for i in range(1,h+1):
        for j in range(1,w+1):
            # 取出当前点为中心的k*k区域
            roi=padding_fm[i-r:i+r+1,j-r:j+r+1]
            # 计算当前点的卷积, 对k*k个点乘后求和
            rs[i-1][j-1]=np.sum(roi*kernel)

    return rs

# 填充0
def fill_zeros(input):
    [c,h,w]=input.shape
    rs=np.zeros([c,h*2+1,w*2+1],np.float32)

    for i in range(c):
        for j in range(h):
            for k in range(w):
                rs[i,2*j+1,2*k+1]=input[i,j,k]

    return rs

def my_deconv(input, weights):
    # weights shape=[out_c, in_c, h, w]
    [out_c, in_c, h, w]=weights.shape
    out_h=h*2
    out_w=w*2
    rs=[]
    for i in range(out_c):
        w=weights[i]
        tmp=np.zeros([out_h, out_w], np.float32)
        for j in range(in_c):
            conv=compute_conv(input[j], w[j])
            # 注意裁剪, 最后一行和最后一列去掉
            tmp=tmp+conv[0:out_h, 0:out_w]
        rs.append(tmp)

    return rs

def main():
    input=np.asarray(input_data, np.float32)
    input= fill_zeros(input)
    weights=np.asarray(weights_data, np.float32)
    deconv=my_deconv(input, weights)

    print(np.asarray(deconv))

if __name__=='__main__':
    main()
```

计算卷积代码, 跟上一篇文章一致。代码直接看注释, 不再解释。运行结果如下:



```
[[[ 4.  3.  6.  2.  7.  3.]  
 [ 4.  3.  3.  2.  7.  5.]  
 [ 8.  6.  8.  5. 11.  2.]  
 [ 3.  2.  7.  2.  3.  3.]  
 [ 5.  5. 11.  3.  9.  3.]  
 [ 2.  1.  4.  5.  4.  4.]]  
  
[[ 4.  1.  7.  0.  7.  2.]  
 [ 5.  6.  0.  1.  8.  5.]  
 [ 8.  0.  8. -2. 14.  2.]  
 [ 3.  3.  9.  8.  1.  0.]  
 [ 3.  0. 13.  0. 11.  2.]  
 [ 3.  5.  3.  1.  3.  0.]]]
```

(/apps/
utm_sc
banner

为了验证实现的代码的正确性，我们使用tensorflow的conv2d_transpose函数执行相同的输入和卷积核，看看结果是否一致。验证代码如下：



```

import tensorflow as tf
import numpy as np
def tf_conv2d_transpose(input,weights):
    #input_shape=[n,height,width,channel]
    input_shape = input.get_shape().as_list()
    #weights_shape=[height,width,out_c,in_c]
    weights_shape=weights.get_shape().as_list()
    output_shape=[input_shape[0], input_shape[1]*2 , input_shape[2]*2 , weights_shape[2]]

    print("output_shape:",output_shape)

    deconv=tf.nn.conv2d_transpose(input,weights,output_shape=output_shape,
        strides=[1, 2, 2, 1], padding='SAME')
    return deconv

def main():
    weights_np=np.asarray(weights_data,np.float32)
    #将输入的每个卷积核旋转180°
    weights_np=np.rot90(weights_np,2,(2,3))

    const_input = tf.constant(input_data , tf.float32)
    const_weights = tf.constant(weights_np , tf.float32 )

    input = tf.Variable(const_input,name="input")
    #[c,h,w]----->[h,w,c]
    input=tf.transpose(input,perm=(1,2,0))
    #[h,w,c]----->[n,h,w,c]
    input=tf.expand_dims(input,0)

    #weights_shape=[out_c,in_c,h,w]
    weights = tf.Variable(const_weights,name="weights")
    #[out_c,in_c,h,w]----->[h,w,out_c,in_c]
    weights=tf.transpose(weights,perm=(2,3,0,1))

    #执行tensorflow的反卷积
    deconv=tf_conv2d_transpose(input,weights)

    init=tf.global_variables_initializer()
    sess=tf.Session()
    sess.run(init)

    deconv_val = sess.run(deconv)

    hwc=deconv_val[0]
    print(hwc)

if __name__=='__main__':
    main()

```

上面代码中，有几点需要注意：

1. 每个卷积核需要旋转180°后，再传入tf.nn.conv2d_transpose函数中，因为tf.nn.conv2d_transpose内部会旋转180°，所以提前旋转，再经过内部旋转后，能保证卷积核跟我们所使用的卷积核的数据排列一致。
2. 我们定义的输入的shape为[c,h,w]需要转为tensorflow所使用的[n,h,w,c]。
3. 我们定义的卷积核shape为[out_c,in_c,h,w],需要转为tensorflow反卷积中所使用的[h,w,out_c,in_c]



执行上面代码后，执行结果如下：

```
[[ 4.  3.  6.  2.  7.  3.]
 [ 4.  3.  3.  2.  7.  5.]
 [ 8.  6.  8.  5. 11.  2.]
 [ 3.  2.  7.  2.  3.  3.]
 [ 5.  5. 11.  3.  9.  3.]
 [ 2.  1.  4.  5.  4.  4.]]
[[ 4.  1.  7.  0.  7.  2.]
 [ 5.  6.  0.  1.  8.  5.]
 [ 8.  0.  8. -2. 14.  2.]
 [ 3.  3.  9.  8.  1.  0.]
 [ 3.  0. 13.  0. 11.  2.]
 [ 3.  5.  3.  1.  3.  0.]]
```

(/apps/
utm_sc
banner

对比结果可以看到，数据是一致的，证明前面手写的python实现的反卷积代码是正确的。

小礼物走一走，来简书关注我

赞赏支持

 Android开发 (/nb/4440436)

举报文章 © 著作权归作者所有



huachao1001 (/u/0a7e42698e4b) 

写了 95017 字，被 2389 人关注，获得了 2819 个喜欢
(/u/0a7e42698e4b)

+ 关注

武汉大学计算机学院硕士

喜欢 | 5



更多分享

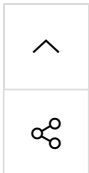


下载简书 App ▶




随时随地发现和创作内容



(/apps/redirect?utm_source=note-bottom-click)

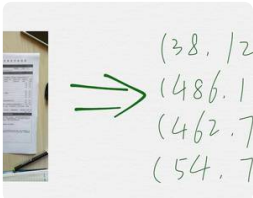


被以下专题收入，发现更多相似内容

-  人工智能 (/c/d800b9b79a22?utm_source=desktop&utm_medium=notes-included-collection)
-  神经网络 理解篇 (/c/aa908bd6079b?utm_source=desktop&utm_medium=notes-included-collection)
-  深度学习 (/c/70e9c4784491?utm_source=desktop&utm_medium=notes-included-collection)

(/apps/
utm_sc
banner


(/p/5ae69f175379?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation

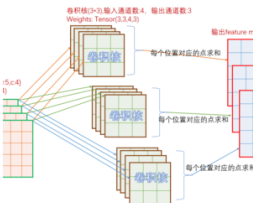
[转载]基于 TensorFlow 和 OpenCV 实现文档检测功能 (/p/5ae69f175379?...

文章来源: github 作者: fengjian 前言 本文不是神经网络或机器学习的入门教学，而是通过一个真实的产品案例，展示了在手机客户端上运行一个神经网络的关键技术点 在卷积神经网络适用的领域里，已经出现了...

 dopami (/u/66640ecf0a46?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation


(/p/abb7d9b82e2a?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation

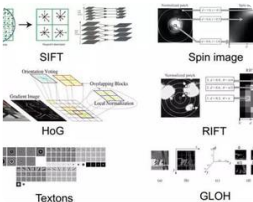
Tensorflow卷积实现原理+手写python代码实现卷积 (/p/abb7d9b82e2a?utm...

从一个通道的图片进行卷积生成新的单通道图的过程很容易理解，对于多个通道卷积后生成多个通道的图理解起来有点抽象。本文以通俗易懂的方式讲述卷积，并辅以图片解释，能快速理解卷积的实现原理。最后...

 huachao1001 (/u/0a7e42698e4b?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation


(/p/f0f574317f49?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation

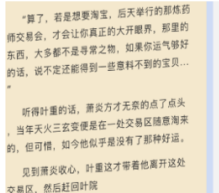
如何使用TensorFlow实现卷积神经网络 (/p/f0f574317f49?utm_campaign=...

姓名: 尤学强 学号: 17101223374 转载自: http://mp.weixin.qq.com/s/C6clDCGMr9t7BcCrY_5uBw 【嵌牛导读】: 深度学习 【嵌牛鼻子】: 时间序列信号, 音频信号, 文本数据 【嵌牛提问】: 卷积神经网络的优...

 51fb659a6d6f (/u/51fb659a6d6f?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation





utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation
UIPageViewController使用 分享笔记 (/p/a676899d9b70?utm_campaign=...

前言 由于公司要开发一款小说类阅读APP，其中体验上非常重要的一点便是翻页效果。为了实现翻页效果，我查询了很多资料后选择使用了UIPageViewController。原因很简单，使用方便，功能强大，开发速度快...

 巫师学徒 (/u/f636fd87e354?)


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation

(/p/b6659e3e50bb?)



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation
好的家分体式集成灶：灶台上的记忆，重温旧日的温暖！ (/p/b6659e3e50bb?)

记忆里，总会浮现出这样的场景：母亲在灶台上忙着切菜，炒菜，做饭。锅碗瓢盆碰撞出叮叮当当的响声。父亲则坐在灶膛前，往灶膛里一把一把的塞柴禾，红红的火光在父亲的脸上闪烁跳跃，忽明忽暗。还记得...

 家居资讯一点通 (/u/8f5d4159a3a0?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation

