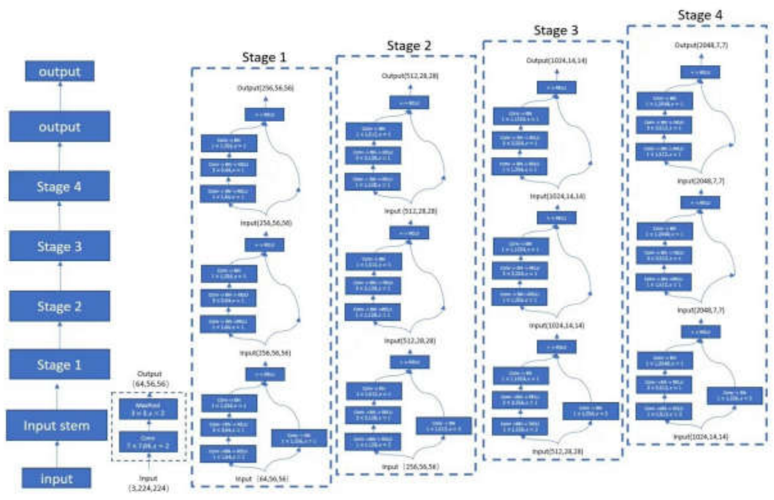
ResNet50复现笔记

零、复现参考图：

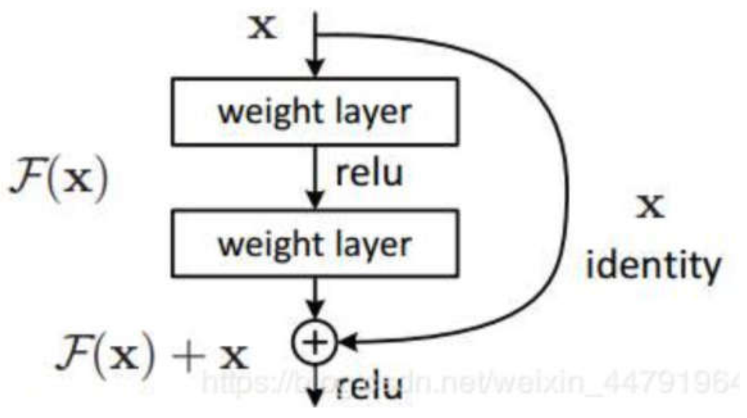


一、残差结构

Residual net(残差网络)

将靠前若干层的某一层数据输出直接跳过多层引入到后面数据层的输入部分。

意味着后面的特征层的内容会有一部分由其前面的某一层线性贡献。


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x})+\mathbf{x}$
relu
https://blog.csdn.net/weixin_4479196

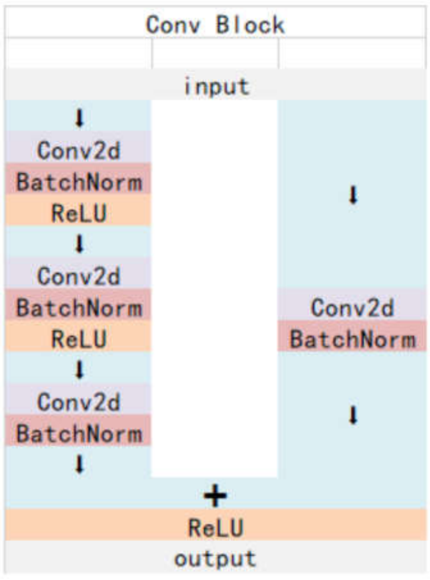
深度残差网络的设计是为了克服由于网络深度加深而产生的学习效率变低与准确率无法有效提升的问题。

二、ResNet50模型基本构成

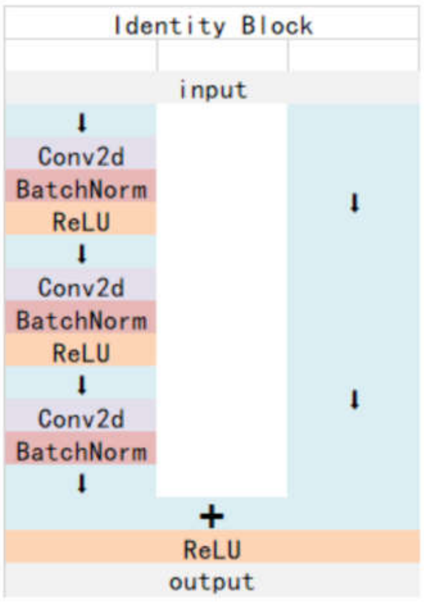
ResNet50有两个基本的块，分别名为**Conv Block**和**Identity Block**

Conv Block输入和输出的维度（通道数和size）是不一样的，所以不能连续串联，它的作用是改变网络的维度；

Identity Block输入维度和输出维度（通道数和size）相同，可以串联，用于加深网络的。

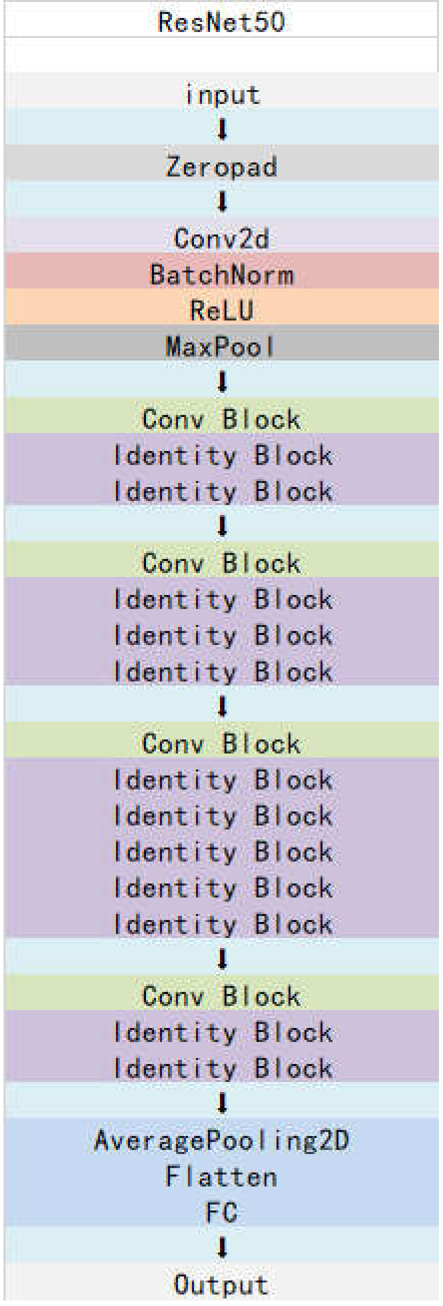


Conv Block结构



Identity Block的结构

三、总体的网络结构



四、代码复现

1.导库

```python
import torch
from torch import nn
```

2.写Block类

```python
'''
    Block的各个plane值：
        inplane：输出block的之前的通道数
        midplane：在block中间处理的时候的通道数（这个值是输出维度的1/4）
        midplane*self.extention：输出的维度
'''
```

```
class Bottleneck(nn.Module):

    #每个stage中维度拓展的倍数
    extention=4

    #定义初始化的网络和参数
    def __init__(self,inplane,midplane,stride,downsample=None):
        super(Bottleneck,self).__init__()

        self.conv1=nn.Conv2d(inplane,midplane,kernel_size=1,stride=stride,bias=False)
        self.bn1=nn.BatchNorm2d(midplane)
        self.conv2=nn.Conv2d(midplane,midplane,kernel_size=3,stride=1,padding=1,bias=Fa
        self.bn2=nn.BatchNorm2d(midplane)
        self.conv3=nn.Conv2d(midplane,midplane*self.extention,kernel_size=1,stride=1,bi
        self.bn3=nn.BatchNorm2d(midplane*self.extention)
        self.relu=nn.ReLU(inplace=False)

        self.downsample=downsample
        self.stride=stride


    def forward(self,x):
        #参差数据
        residual=x

        #卷积操作
        out=self.relu(self.bn1(self.conv1(x)))
        out=self.relu(self.bn2(self.conv2(out)))
        out=self.relu(self.bn3(self.conv3(out)))

        #是否直连（如果时Identity block就是直连；如果是Conv Block就需要对参差边进行卷积，改
        if(self.downsample!=None):
            residual=self.downsample(x)

        #将参差部分和卷积部分相加
        out+=residual
        out=self.relu(out)

        return out
```

3.写Resnet结构

```python

class ResNet(nn.Module):

    #初始化网络结构和参数
    def __init__(self,block,layers,num_classes=1000):
        #self.inplane为当前的fm的通道数
        self.inplane=64

        super(ResNet,self).__init__()

        #参数
        self.block=block
        self.layers=layers

        #stem的网络层
        self.conv1=nn.Conv2d(3,self.inplane,kernel_size=7,stride=2,padding=3,bias=False
        self.bn1=nn.BatchNorm2d(self.inplane)
        self.relu=nn.ReLU()
        self.maxpool=nn.MaxPool2d(kernel_size=3,padding=1,stride=2)

        #64，128，256，512是指扩大4倍之前的维度,即Identity Block的中间维度
        self.stage1=self.make_layer(self.block,64,self.layers[0],stride=1)
        self.stage2=self.make_layer(self.block,128,self.layers[1],stride=2)
        self.stage3=self.make_layer(self.block,256,self.layers[2],stride=2)
        self.stage4=self.make_layer(self.block,512,self.layers[3],stride=2)

        #后续的网络
        self.avgpool=nn.AvgPool2d(7)
        self.fc = nn.Linear(512 * block.extention, num_classes)





    def forward(self,x):

        #stem部分:conv+bn+relu+maxpool
        out=self.conv1(x)
        out=self.bn1(out)
        out=self.relu(out)
        out=self.maxpool(out)
```

```python

```

```python
        #block
        out=self.stage1(out)
        out=self.stage2(out)
        out=self.stage3(out)
        out=self.stage4(out)

        #分类
        out=self.avgpool(out)
        out = torch.flatten(out, 1)
        out=self.fc(out)


        return out

    def make_layer(self,block,midplane,block_num,stride=1):
        '''
            block:block模块
            midplane：每个模块中间运算的维度，一般等于输出维度/4
            block_num：重复次数
            stride：Conv Block的步长
        '''

        block_list=[]

        #先计算要不要加downsample模块
        downsample=None
        if(stride!=1 or self.inplane!=midplane*block.extention):
            downsample=nn.Sequential(
                nn.Conv2d(self.inplane,midplane*block.extention,stride=stride,kernel_s
                nn.BatchNorm2d(midplane*block.extention)
            )


        #Conv Block
        conv_block=block(self.inplane,midplane,stride=stride,downsample=downsample)
        block_list.append(conv_block)
        self.inplane=midplane*block.extention

        #Identity Block
        for i in range(1,block_num):
            block_list.append(block(self.inplane,midplane,stride=1))

        return nn.Sequential(*block_list)
```

4.调用

```
resnet = ResNet(Bottleneck, [3, 4, 6, 3])
x=torch.randn(1,3,224,224)
x=resnet(x)
print(x.shape)
```

参考：https://blog.csdn.net/weixin_44791964/article/details/102790260

resnet50.svg
421.65KB

myResnet.py
4.56KB