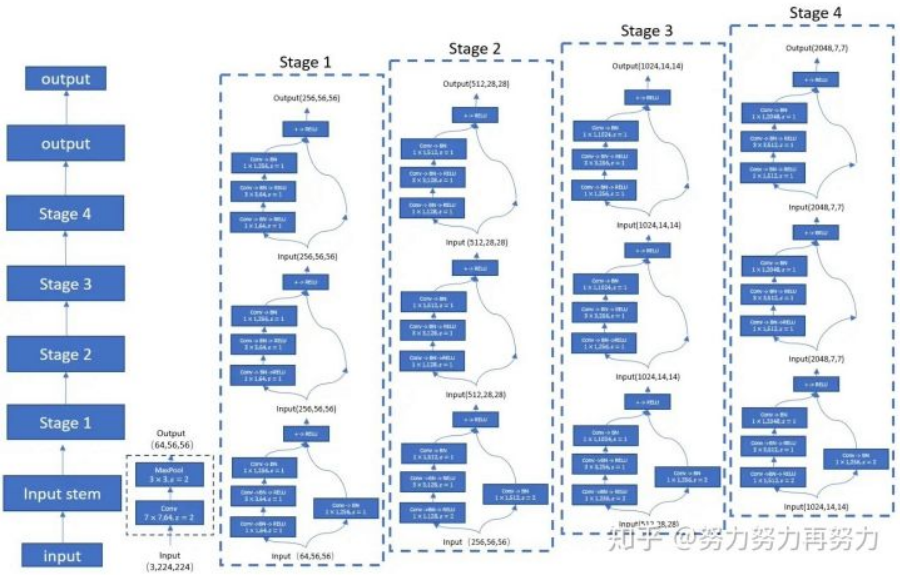
ResNet50复现笔记

Fighting... 已关注
厦门大学 计算机科学与技术硕士在读

视频讲解：b23.tv/DDCDCp

（代码在评论区置顶）

零、复现参考图：

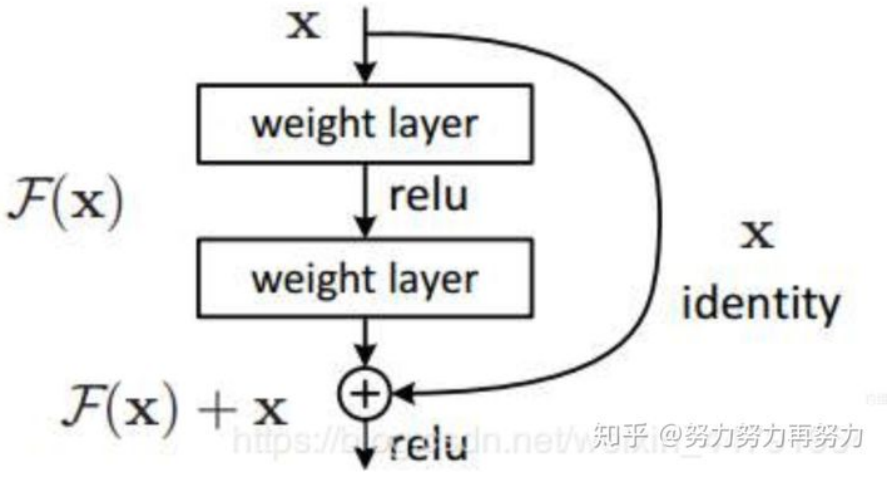


一、残差结构

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。



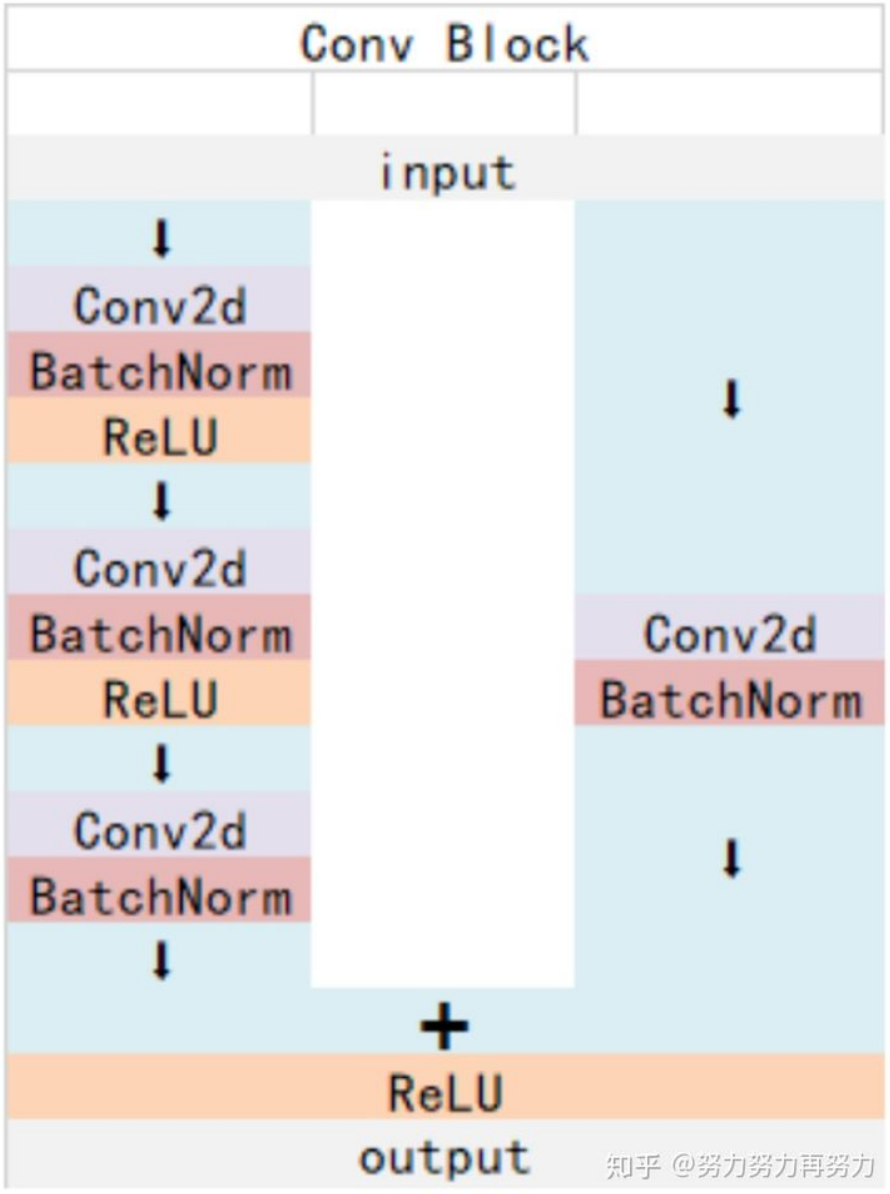
深度残差网络的设计是为了克服由于网络深度加深而产生的学习效率变低与准确率无法有效提升的

二、ResNet50模型基本构成

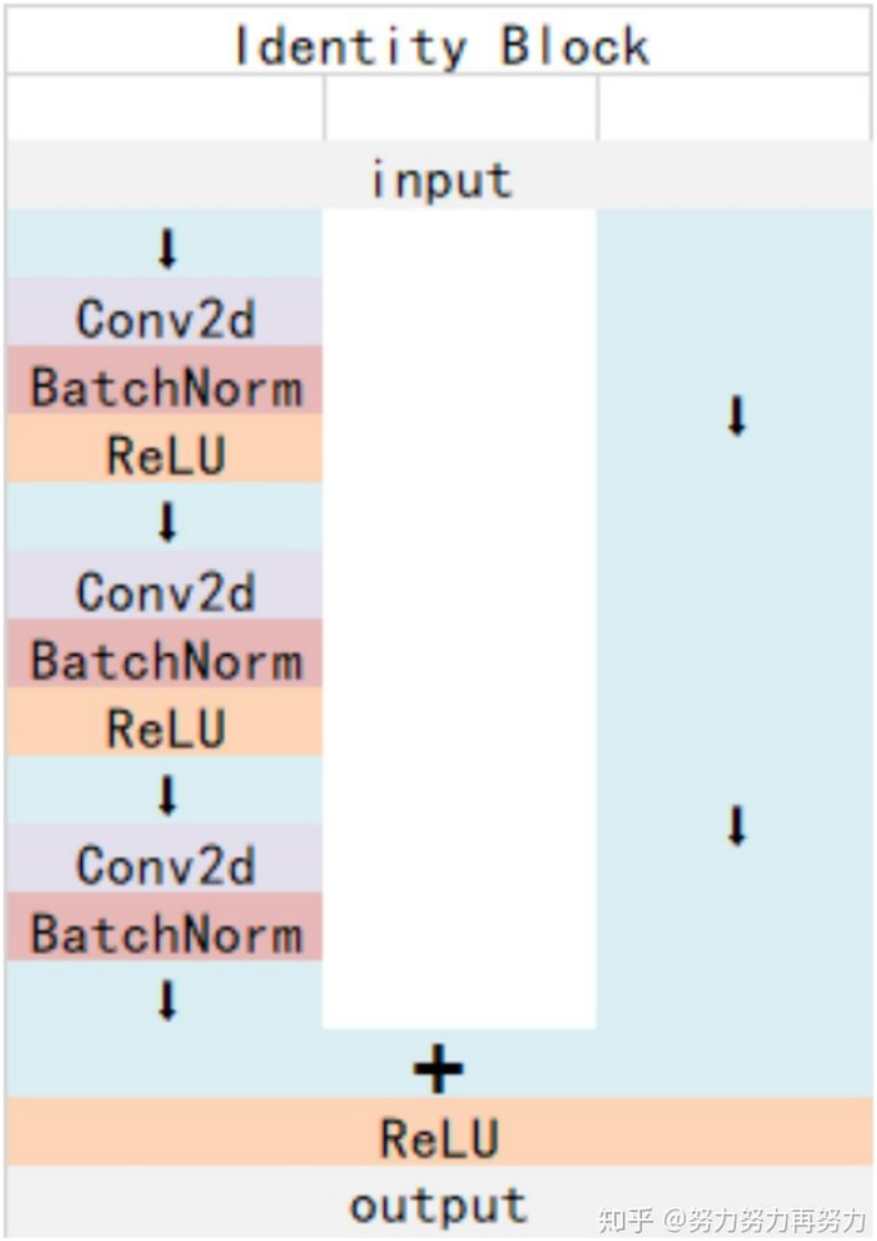
ResNet50有两个基本的块，分别名为**Conv Block**和**Identity Block**

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。

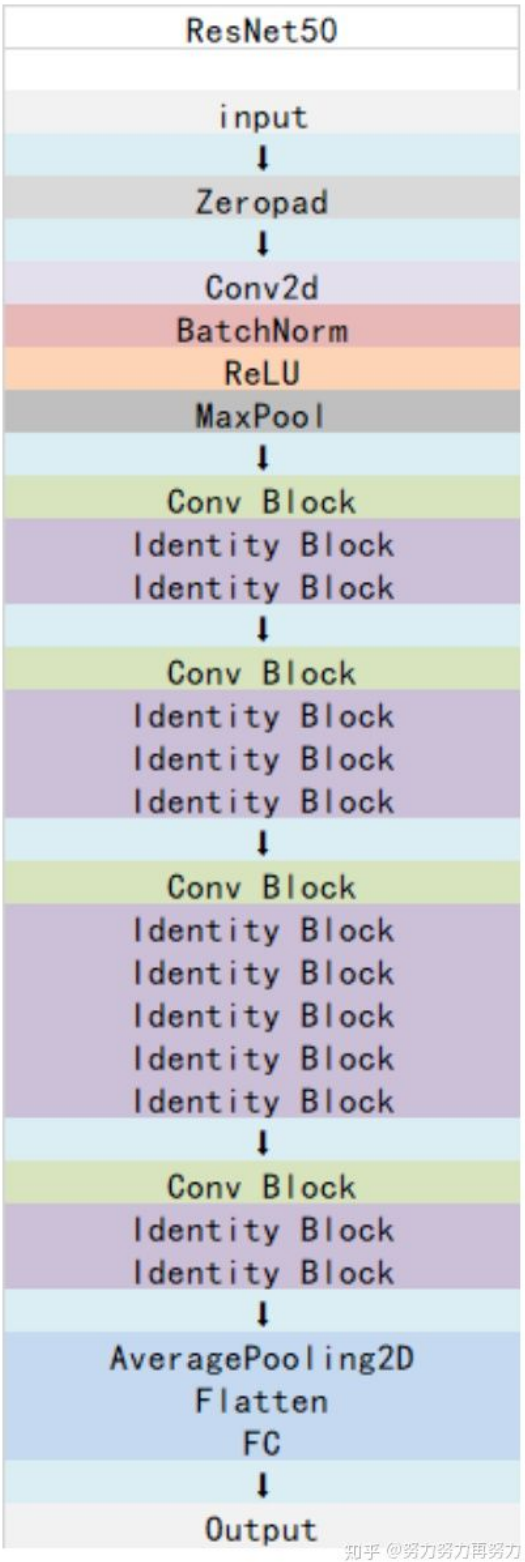


Conv Block结构



Identity Block的结构

三、总体的网络结构



四、代码复现

1.导库

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

```
'''
    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=F
        self.bn2=nn.BatchNorm2d(midplane)
        self.conv3=nn.Conv2d(midplane,midplane*self.extention,kernel_size=1,stride=1,b
        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，改变
        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

```
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=Fals
        self.bn1=nn.BatchNorm2d(self.inplane)
```

```
        #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)

        #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：blog.csdn.net/weixin_44...

复现之后resnet的图：

链接: pan.baidu.com/s/1pazTBD... 密码: ps9c

复现之后resnet的代码：

链接: pan.baidu.com/s/1SKfgCn... 密码: fi1w

编辑于 2021-03-08 08:40

ResNet　深度学习（Deep Learning）　PyTorch

文章被以下专栏收录

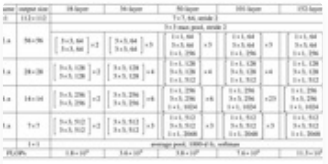
CV算法笔记
代码算法的流程笔记

推荐阅读

ResNet结构解析及pytorch代码

标签： pytorch ResNet是恺明大神提出来的一种结构，近些年的一些结构变种，很多也是基于ResNet做的一些改进，可以说ResNet开创了更深的网络的先河，并且在很多计算机视觉学习上都取得了不...

小新蜡笔

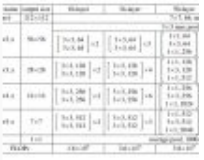


通过Pytorch实现ResNet18

samcw　发表于深度学习

完整学习 ResNet 家族 ResNext, SEResNet代码实...

史蒂芬方



ResNet网络结构详 Pytorch版本模型的

代辰

7 条评论　切换为时间排序

写下你的评论...

醉酒林　03-11

"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。"
您好，作者。 想问一下这两句话有参考文献吗

赞

04-06

▲ 赞同 35　▼　7 条评论　分享　喜欢　收藏　申请转载　…

赞

萌萌懂懂 2021-03-18

我想问一下，resnet里那个num_classes为什么设置1000?

赞

FightingCV (作者) 回复 萌萌懂懂 2021-03-18

imagenet是1000类，数据集有几类就设置成多少

赞

萌萌懂懂 回复 FightingCV (作者) 2021-03-18

我把背景也加进去了，不要紧吧?

赞

南柯一梦 2021-03-07

好文

赞

WinOneKey 2020-10-28

感谢感谢，正好用上

赞