

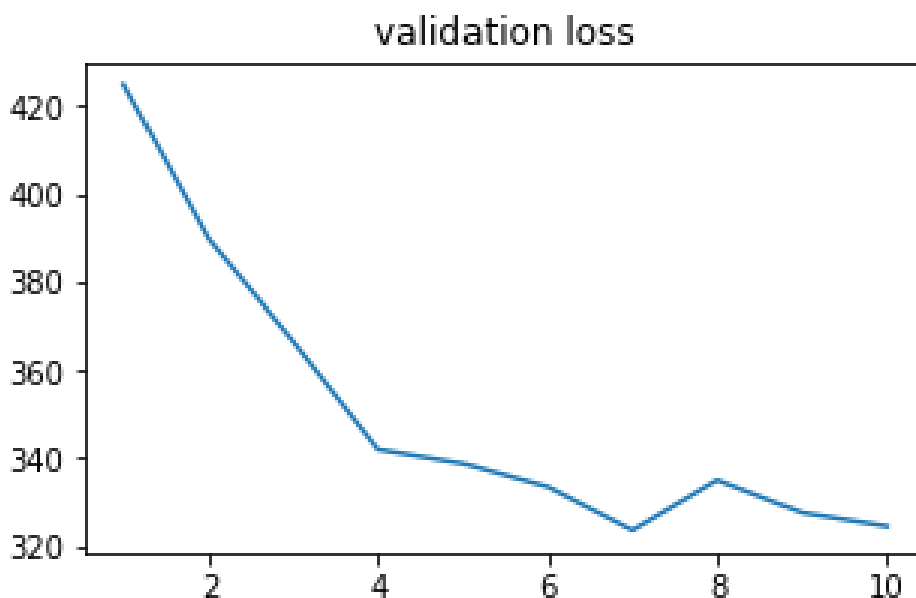
# 卷积神经网络实验报告

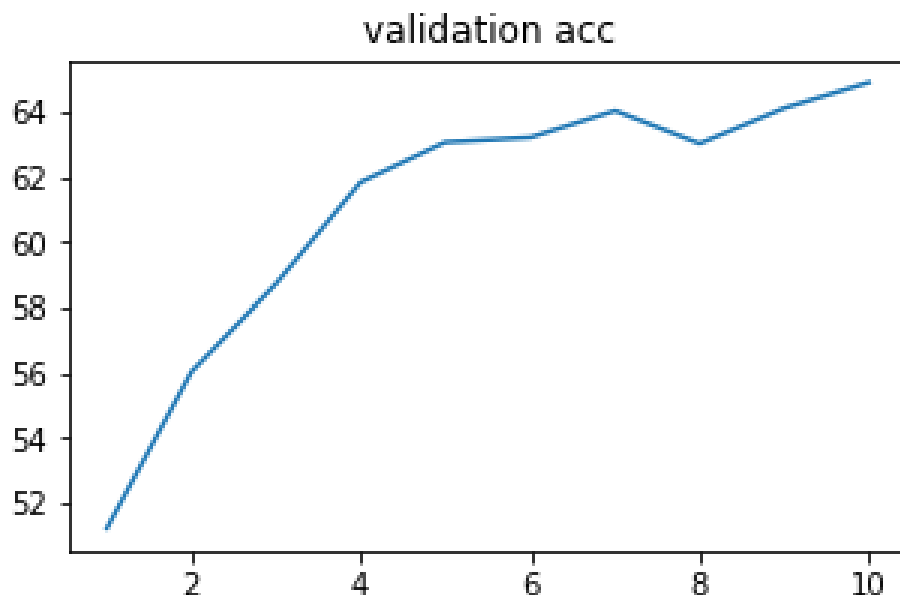
## 原始版本CNN网络

实验中给出的CNN网络结构十分简单，整体来说由两个卷积层和三个全连接层构成。输入的图像维度为[3,32,32]，首先通过第一个卷积层(Conv2d(3,6,5))，得到特征图大小为[6,28,28]，然后通过ReLU激活函数，在使用最大池化层进行下采样，得到特征图大小为[6,14,14]。然后通过第二个卷积层(Conv2d(6,16,5))，得到特征图大小为[16,10,10]，然后再通过最大池化层进行下采样，得到特征图[16,5,5]，将得到的特征展平得到一维向量，然后通过三个全连接层，参数分别为[400, 120],[120, 84],[84, 10]。具体网络结构如下。

```
1 Net(  
2   (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))  
3   (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,  
   ceil_mode=False)  
4   (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))  
5   (fc1): Linear(in_features=400, out_features=120, bias=True)  
6   (fc2): Linear(in_features=120, out_features=84, bias=True)  
7   (fc3): Linear(in_features=84, out_features=10, bias=True)  
8 )
```

在本次实验中，选用Adam作为优化器，学习率为0.001，训练了10个epoch，得到训练结果如图。

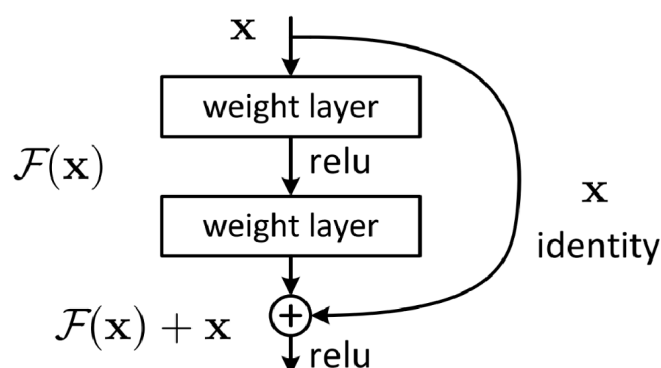




可以看到，仅仅是两层的卷积神经网络和层的全连接网络结构，在训练了10个epoch左右，也能够接近收敛，并且验证的准确率到了64%左右。虽然这个结果并不高，但是对于这样简单的网络结构和较少的训练论述而言，也算不错的结果。

## ResNet实验

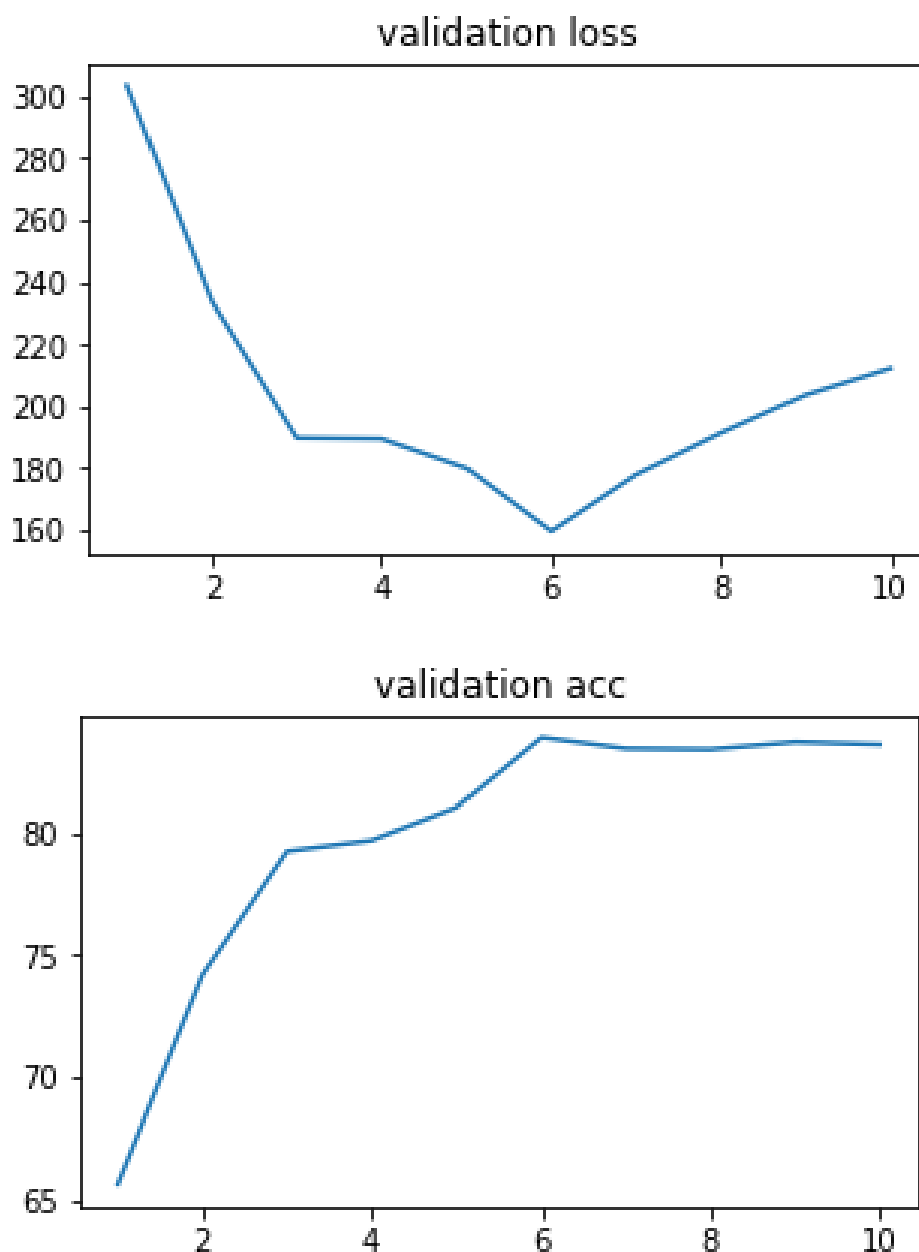
ResNet通过引入残差链接的方式，有效解决了模型层数过多的时候出现梯度消失的问题。ResNet中基本块的结构如下。



对于层数较少的ResNet，一般直接采用BasicBlock，其中包含两个  $3 \times 3$  卷积层，得到的输出结果和该block的输入相加，这就是残差链接。通过残差连接，能够保证该block在进行梯度回传的时候，有一项的值为1，不会出现梯度消失的现象，使得网络能够通过增加层数来提升性能。

在本次实验中，由于只是进行简单的十分类任务，因此并不需要较深的网络，在这里选择使用ResNet最少层数的版本ResNet18。ResNet18包含4层。每层中都有两个BasicBlock，每一层的通道数逐步翻倍，四层的通道数分别为[64, 128, 256, 512]。通过增加通道数，降低特征图的大小，提取到不同方面的特征。其结构见最后附录。

为了保证实验对比的公平性，我们使用Adam优化器，学习率为0.001，训练了10个epoch，得到实验结果如图所示。

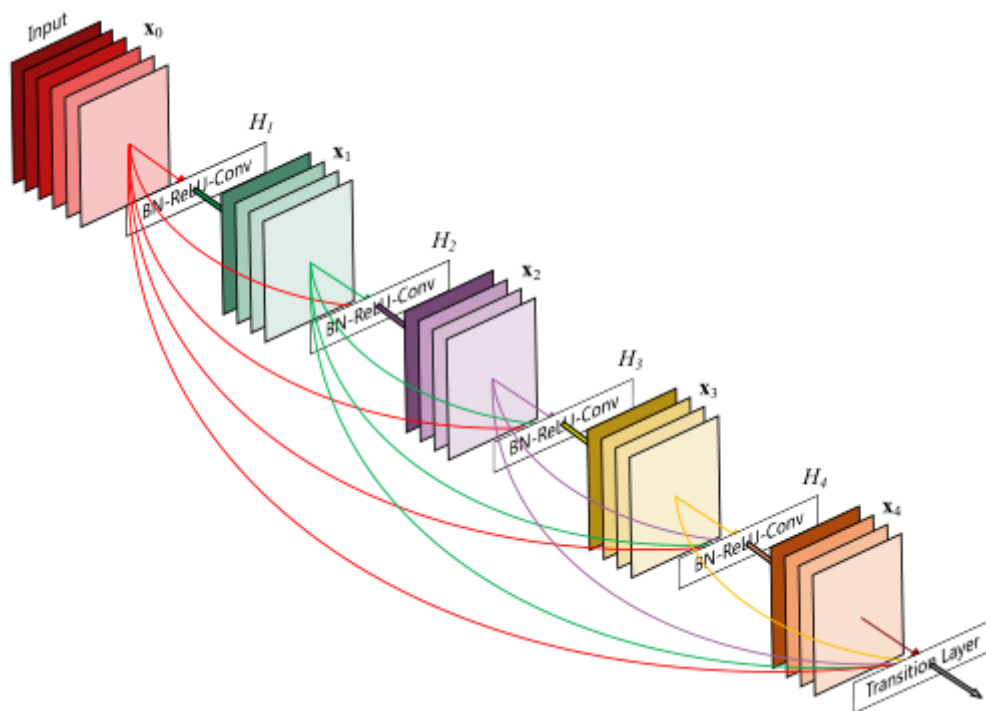


从验证集的loss和准确率来看，ResNet的效果要明显优于之前的简单CNN结构，一方面是ResNet中使用了更深的网络层数和更大的隐藏层宽度，另一方面是ResNet中增加了残差链接，使得模型在训练的过程中参数能够得到更好地更新。由于ResNet中参数较多，对于本次实验的任务来讲很容易出现过拟合的现象，从实验结果中也能够看到一些过拟合的端倪。在训练的最后几个epoch中，训练集上的准确率还在提升，但是验证集上的准确率基本上已经稳定了，并且loss还有上升的趋势。

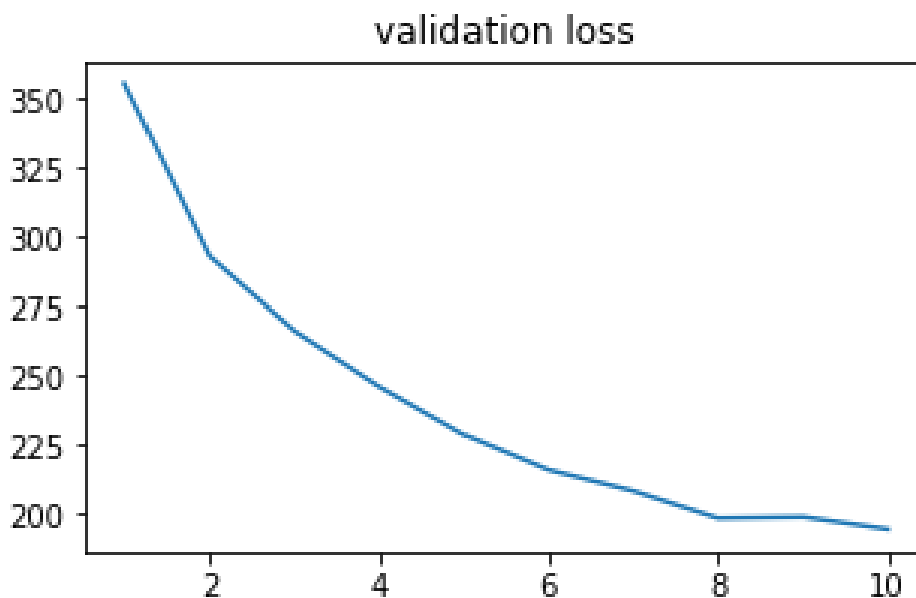
## DenseNet实验

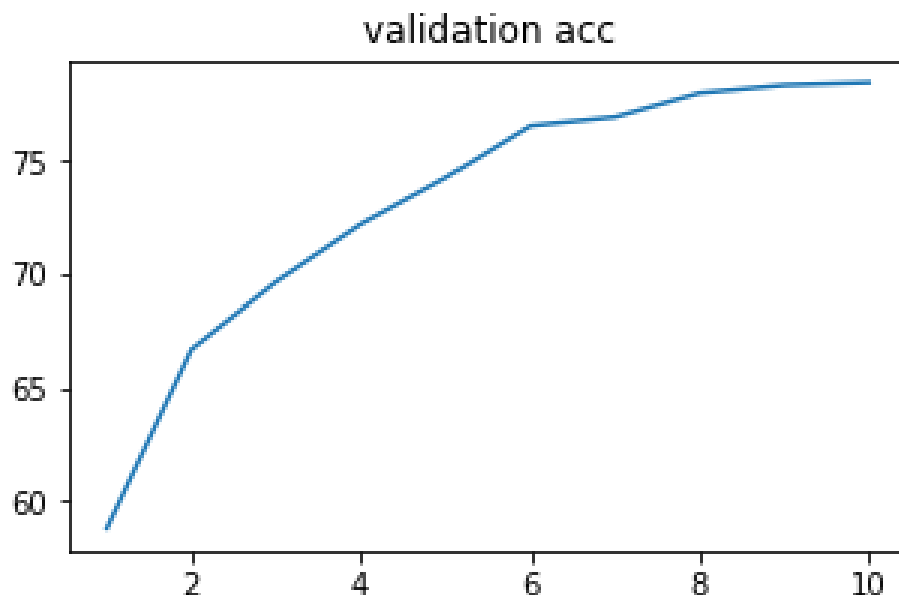
DenseNet也是一种深层神经网络的结构，它具有缓解梯度消失，增强特征传播、鼓励特征复用和大幅减少参数数量等优点。相较于ResNet在不同的Layer之间采用残差链接的方式，DenseNet采用了一种更为激进的策略，就是将之前Layer的输出全部拼接到一起作为本层的输入，这样就能够尽最大可能利用各层的特征，同样每一层也都有从最后一层可以直接回传的梯度，可以有效的缓解梯度消失的问题。

DenseNet的结构如下。其中每一个基本块的结构和ResNet十分相似，Transition Layer(过渡层): 采用 $1 \times 1$ Conv和 $2 \times 2$ 平均池化作为相邻Dense Block之间的转换层，减少feature map数和缩小feature map size, size指width\*height。在相邻Dense Block中输出的feature map size是相同的，以便它们能够很容易的连接在一起。



然后为了保证实验对比的公平，在实验中实现了一个22层的DenseNet，其结构见最后附录。训练过程都是采用了Adam优化器，学习率为0.01，训练了10个epoch，得到训练结果如下。



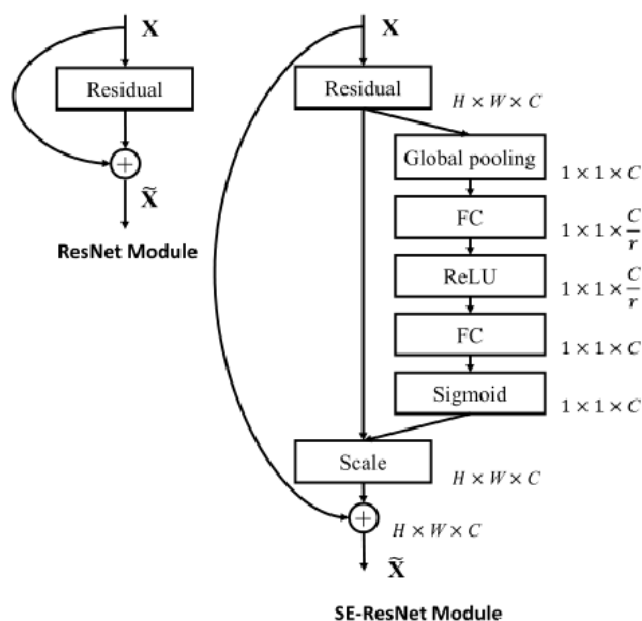


可以看到在本次实验中，DenseNet的性能要弱于ResNet，其原因大概是DenseNet的设计更加适合较深层的网络，而本次实现的DenseNet22相对来讲网络层数较浅，这样在进行特征拼接的时候，前几层的特征表示也不好，导致影响到了后续的特征表示。

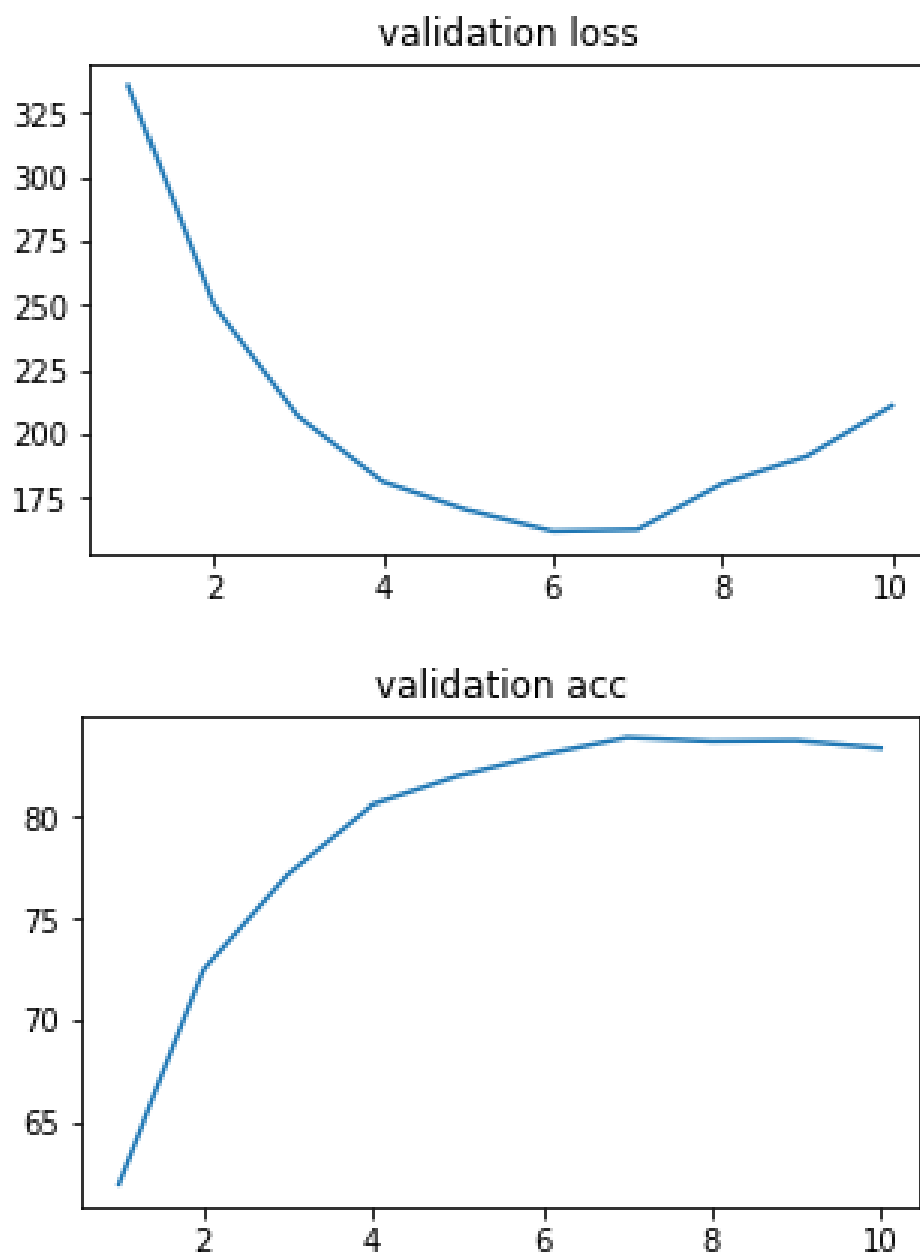
## SE-ResNet实验

带有SE模块的ResNet主要是解决了ResNet中通道数很多，而通道中可能提取到了一些干扰的因素，因此需要采用一种机制来筛选通道中的有效信息。SE模块就是这样的一种门控机制，通过通道注意力的方式来过滤出通道之间哪些通道是更为重要的。

具体而言，其实现方法是首先通过全局平均池化来获取到每一个通道中的信息，然后通过一个两层的MLP，在经过sigmoid计算出每一个通道的重要性，并通过这种门控机制，筛选更为有效的通道。其结构如下。

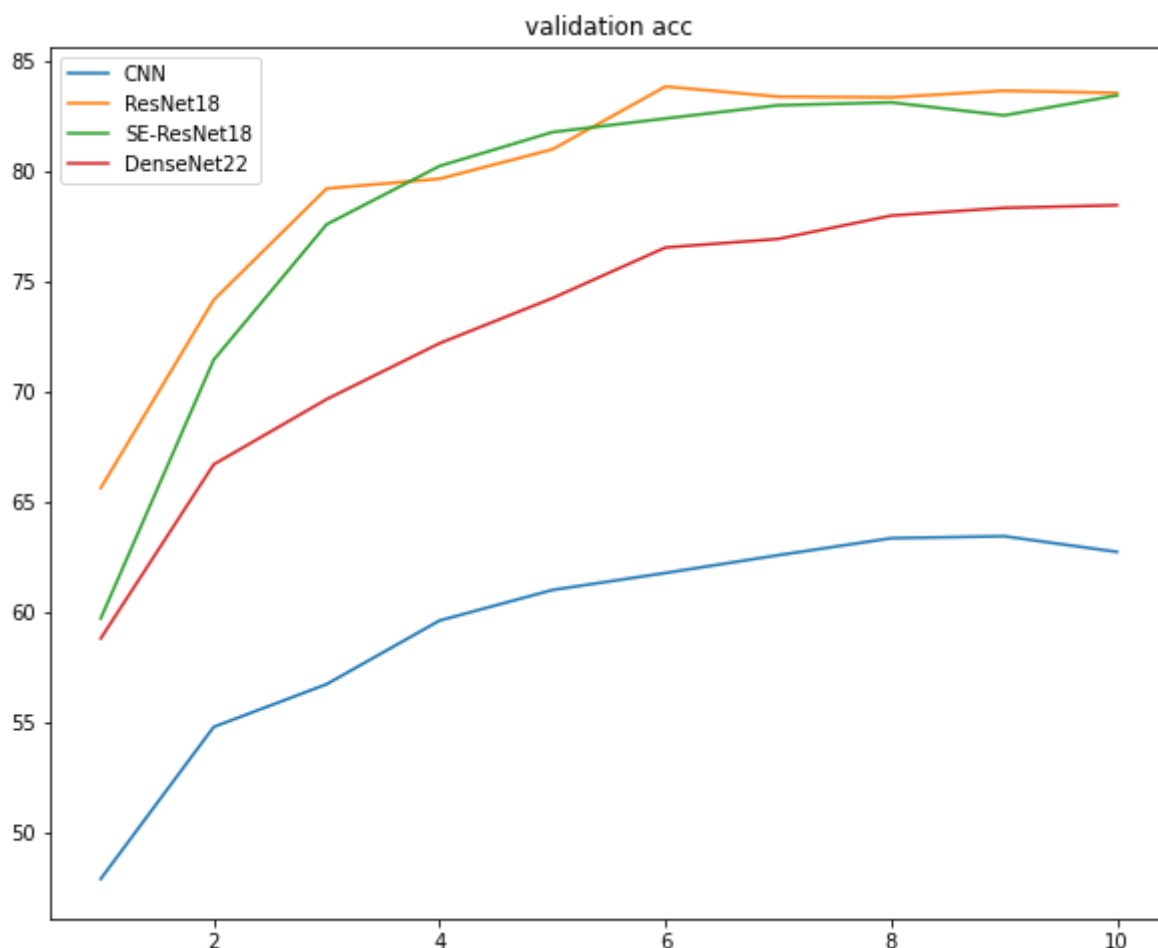


在对ResNet18进行改进之后，得到了Se-ResNet18，其结构见最后的附录。然后为了保证实验对比的公平，我们都是采用了Adam优化器，学习率为0.01，训练了10个epoch，得到训练结果如下。



## 实验对比

在本次实验中，总共实现了基础CNN、ResNet18、SE-ResNet18和DenseNet22网络，并且在相同的实验设定下进行测试，在训练阶段，全部采用Adam优化器，学习率为0.01，训练10个epoch。得到对比结果如下。



从实验结果上可以看出，普通的CNN结构效果最差，这里主要原因是因为CNN网络的层数相比较浅，对于特征的代表能力比较弱。

在增加了增加了残差链接之后，实现的ResNet18结构效果实现了明显的提升，主要原因是ResNet18大大加深了网络的层数，通过更深层的网络提高特征抽取能力。并且残差链接有效降低了深层网络中梯度消失的影响。这样可以看到ResNet18的效果在所有实现的方法中是最好的。

而增加了SE模块的ResNet18并没有取得性能提升，笔者分析可能是因为增加了SE模块后网络的参数增加了，在相同的轮数下还没有能够达到一个比较好的性能。从实验图像中也能够看出ResNet18基本上已经呈现出收敛，而增加了SE模块后仍有上升趋势。SE模块主要是通过筛选有效通道，增强通道中有效信息的提出，过滤掉噪声影响，来提升深层神经网络中多通道的特征表达能力，进而实现特征抽取效果的提升。

而DenseNet22并没有表现出其应有的优秀效果，笔者通过阅读论文发现，DenseNet适合更深层的网络结构，而在本次实验中实现的22层相对比较浅，DenseNet结构并不能很好地发挥作用。主要是因为DenseNet采用了更为激进的特征融合方式，后面的层会拼接前面层的输出，这就要求每一层的输出都要尽可能好，而较浅地网络层数会导致前面的层抽取的特征效果不加，进而影响了后续层的特征表示，因此影响了模型最终的性能。

## 附录

# ResNet18结构

```
1 ResNet_18(  
2     (conv1): Sequential(  
3         (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,  
4         1), bias=False)  
5         (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
6         track_running_stats=True)  
7         (2): ReLU()  
8     )  
9     (layer1): Sequential(  
10        (0): BasicBlock(  
11            (left): Sequential(  
12                (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),  
13                padding=(1, 1), bias=False)  
14                (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
15                track_running_stats=True)  
16                (2): ReLU(inplace=True)  
17                (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),  
18                padding=(1, 1), bias=False)  
19                (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
20                track_running_stats=True)  
21            )  
22            (shortcut): Sequential()  
23        )  
24        (1): BasicBlock(  
25            (left): Sequential(  
26                (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),  
27                padding=(1, 1), bias=False)  
28                (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
29                track_running_stats=True)  
30                (2): ReLU(inplace=True)  
31                (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),  
32                padding=(1, 1), bias=False)  
33                (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
34                track_running_stats=True)  
35            )  
36            (shortcut): Sequential()  
37        )  
38    )  
39    (layer2): Sequential(  
40        (0): BasicBlock(  
41            (left): Sequential(  
42                (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2),  
43                padding=(1, 1), bias=False)  
44                (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,  
45                track_running_stats=True)  
46                (2): ReLU(inplace=True)  
47                (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),  
48                padding=(1, 1), bias=False)
```



```

36         (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
37     )
38     (shortcut): Sequential(
39         (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2),
bias=False)
40         (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
41     )
42 )
43 (1): BasicBlock(
44     (left): Sequential(
45         (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
46         (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
47         (2): ReLU(inplace=True)
48         (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
49         (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
50     )
51     (shortcut): Sequential()
52 )
53 )
54 (layer3): Sequential(
55     (0): BasicBlock(
56         (left): Sequential(
57             (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
58             (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
59             (2): ReLU(inplace=True)
60             (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
61             (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
62         )
63         (shortcut): Sequential(
64             (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2),
bias=False)
65             (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
66         )
67     )
68     (1): BasicBlock(
69         (left): Sequential(
70             (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
71             (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
72             (2): ReLU(inplace=True)

```

```

73         (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
74         (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
75     )
76     (shortcut): Sequential()
77 )
78 )
79 (layer4): Sequential(
80     (0): BasicBlock(
81         (left): Sequential(
82             (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
83             (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
84             (2): ReLU(inplace=True)
85             (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
86             (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
87         )
88         (shortcut): Sequential(
89             (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2),
bias=False)
90             (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
91         )
92     )
93     (1): BasicBlock(
94         (left): Sequential(
95             (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
96             (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
97             (2): ReLU(inplace=True)
98             (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
99             (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
100         )
101         (shortcut): Sequential()
102     )
103 )
104 (fc): Linear(in_features=512, out_features=10, bias=True)
105 )
106

```

## SE-ResNet18结构

```
1 SE_ResNet_18(  
2     (conv1): Sequential(  
3         (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,  
4         1), bias=False)  
5         (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
6         track_running_stats=True)  
7         (2): ReLU()  
8     )  
9     (layer1): Sequential(  
10        (0): SEBasicBlock(  
11            (left): Sequential(  
12                (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),  
13                padding=(1, 1), bias=False)  
14                (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
15                track_running_stats=True)  
16                (2): ReLU(inplace=True)  
17                (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),  
18                padding=(1, 1), bias=False)  
19                (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
20                track_running_stats=True)  
21            )  
22            (shortcut): Sequential()  
23            (se): SEModule(  
24                (avg_pool): AdaptiveAvgPool2d(output_size=1)  
25                (fc1): Conv2d(64, 4, kernel_size=(1, 1), stride=(1, 1))  
26                (relu): ReLU(inplace=True)  
27                (fc2): Conv2d(4, 64, kernel_size=(1, 1), stride=(1, 1))  
28                (sigmoid): Sigmoid()  
29            )  
30        )  
31        (1): SEBasicBlock(  
32            (left): Sequential(  
33                (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),  
34                padding=(1, 1), bias=False)  
35                (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
36                track_running_stats=True)  
37                (2): ReLU(inplace=True)  
38                (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),  
39                padding=(1, 1), bias=False)  
40                (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
41                track_running_stats=True)  
42            )  
43            (shortcut): Sequential()  
44            (se): SEModule(  
45                (avg_pool): AdaptiveAvgPool2d(output_size=1)  
46                (fc1): Conv2d(64, 4, kernel_size=(1, 1), stride=(1, 1))  
47                (relu): ReLU(inplace=True)  
48                (fc2): Conv2d(4, 64, kernel_size=(1, 1), stride=(1, 1))  
49                (sigmoid): Sigmoid()  
50            )  
51        )  
52    )  
53 )
```

```

40     )
41 )
42 )
43 (layer2): Sequential(
44   (0): SEBasicBlock(
45     (left): Sequential(
46       (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
47       (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
48       (2): ReLU(inplace=True)
49       (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
50       (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
51     )
52     (shortcut): Sequential(
53       (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2),
bias=False)
54       (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
55     )
56     (se): SEModule(
57       (avg_pool): AdaptiveAvgPool2d(output_size=1)
58       (fc1): Conv2d(128, 8, kernel_size=(1, 1), stride=(1, 1))
59       (relu): ReLU(inplace=True)
60       (fc2): Conv2d(8, 128, kernel_size=(1, 1), stride=(1, 1))
61       (sigmoid): Sigmoid()
62     )
63   )
64   (1): SEBasicBlock(
65     (left): Sequential(
66       (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
67       (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
68       (2): ReLU(inplace=True)
69       (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
70       (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
71     )
72     (shortcut): Sequential()
73     (se): SEModule(
74       (avg_pool): AdaptiveAvgPool2d(output_size=1)
75       (fc1): Conv2d(128, 8, kernel_size=(1, 1), stride=(1, 1))
76       (relu): ReLU(inplace=True)
77       (fc2): Conv2d(8, 128, kernel_size=(1, 1), stride=(1, 1))
78       (sigmoid): Sigmoid()
79     )
80   )
81 )

```

```

82     (layer3): Sequential(
83         (0): SEBasicBlock(
84             (left): Sequential(
85                 (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
86                 (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
87                 (2): ReLU(inplace=True)
88                 (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
89                 (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
90             )
91             (shortcut): Sequential(
92                 (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2),
bias=False)
93                 (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
94             )
95             (se): SEModule(
96                 (avg_pool): AdaptiveAvgPool2d(output_size=1)
97                 (fc1): Conv2d(256, 16, kernel_size=(1, 1), stride=(1, 1))
98                 (relu): ReLU(inplace=True)
99                 (fc2): Conv2d(16, 256, kernel_size=(1, 1), stride=(1, 1))
100                (sigmoid): Sigmoid()
101            )
102        )
103        (1): SEBasicBlock(
104            (left): Sequential(
105                (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
106                (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
107                (2): ReLU(inplace=True)
108                (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
109                (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
110            )
111            (shortcut): Sequential()
112            (se): SEModule(
113                (avg_pool): AdaptiveAvgPool2d(output_size=1)
114                (fc1): Conv2d(256, 16, kernel_size=(1, 1), stride=(1, 1))
115                (relu): ReLU(inplace=True)
116                (fc2): Conv2d(16, 256, kernel_size=(1, 1), stride=(1, 1))
117                (sigmoid): Sigmoid()
118            )
119        )
120    )
121    (layer4): Sequential(
122        (0): SEBasicBlock(
123            (left): Sequential(

```

```

124         (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
125         (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
126         (2): ReLU(inplace=True)
127         (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
128         (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
129     )
130     (shortcut): Sequential(
131         (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2),
bias=False)
132         (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
133     )
134     (se): SEModule(
135         (avg_pool): AdaptiveAvgPool2d(output_size=1)
136         (fc1): Conv2d(512, 32, kernel_size=(1, 1), stride=(1, 1))
137         (relu): ReLU(inplace=True)
138         (fc2): Conv2d(32, 512, kernel_size=(1, 1), stride=(1, 1))
139         (sigmoid): Sigmoid()
140     )
141 )
142     (1): SEBasicBlock(
143         (left): Sequential(
144             (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
145             (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
146             (2): ReLU(inplace=True)
147             (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
148             (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
149         )
150         (shortcut): Sequential()
151         (se): SEModule(
152             (avg_pool): AdaptiveAvgPool2d(output_size=1)
153             (fc1): Conv2d(512, 32, kernel_size=(1, 1), stride=(1, 1))
154             (relu): ReLU(inplace=True)
155             (fc2): Conv2d(32, 512, kernel_size=(1, 1), stride=(1, 1))
156             (sigmoid): Sigmoid()
157         )
158     )
159 )
160     (fc): Linear(in_features=512, out_features=10, bias=True)
161 )
162

```

## DenseNet结构

```
1 DenseNet(  
2     (conv_1): Conv2d(3, 24, kernel_size=(3, 3), stride=(1, 1), padding=  
3         (1, 1), bias=False)  
4     (dense1): Sequential(  
5         (0): Bottleneck(  
6             (bn_1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True,  
7                 track_running_stats=True)  
8             (conv_1): Conv2d(24, 48, kernel_size=(1, 1), stride=(1, 1),  
9                 bias=False)  
10             (bn_2): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,  
11                 track_running_stats=True)  
12             (conv_2): Conv2d(48, 12, kernel_size=(3, 3), stride=(1, 1),  
13                 padding=(1, 1), bias=False)  
14         )  
15     (1): Bottleneck(  
16         (bn_1): BatchNorm2d(36, eps=1e-05, momentum=0.1, affine=True,  
17             track_running_stats=True)  
18         (conv_1): Conv2d(36, 48, kernel_size=(1, 1), stride=(1, 1),  
19             bias=False)  
20         (bn_2): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,  
21             track_running_stats=True)  
22         (conv_2): Conv2d(48, 12, kernel_size=(3, 3), stride=(1, 1),  
23             padding=(1, 1), bias=False)  
24     )  
25     (2): Bottleneck(  
26         (bn_1): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,  
27             track_running_stats=True)  
28         (conv_1): Conv2d(48, 48, kernel_size=(1, 1), stride=(1, 1),  
29             bias=False)  
30         (bn_2): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,  
31             track_running_stats=True)  
32         (conv_2): Conv2d(48, 12, kernel_size=(3, 3), stride=(1, 1),  
33             padding=(1, 1), bias=False)  
34     )  
35     (trans_1): Transition(  
36         (bn): BatchNorm2d(60, eps=1e-05, momentum=0.1, affine=True,  
37             track_running_stats=True)  
38         (conv): Conv2d(60, 30, kernel_size=(1, 1), stride=(1, 1),  
39             bias=False)  
40     )  
41     (dense2): Sequential(  
42         (0): Bottleneck(  
43             (bn_1): BatchNorm2d(30, eps=1e-05, momentum=0.1, affine=True,  
44                 track_running_stats=True)  
45             (conv_1): Conv2d(30, 48, kernel_size=(1, 1), stride=(1, 1),  
46                 bias=False)  
47             (bn_2): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,  
48                 track_running_stats=True)
```

```

32         (conv_2): Conv2d(48, 12, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
33     )
34     (1): Bottleneck(
35         (bn_1): BatchNorm2d(42, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
36         (conv_1): Conv2d(42, 48, kernel_size=(1, 1), stride=(1, 1),
bias=False)
37         (bn_2): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
38         (conv_2): Conv2d(48, 12, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
39     )
40     (2): Bottleneck(
41         (bn_1): BatchNorm2d(54, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
42         (conv_1): Conv2d(54, 48, kernel_size=(1, 1), stride=(1, 1),
bias=False)
43         (bn_2): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
44         (conv_2): Conv2d(48, 12, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
45     )
46 )
47 (trans_2): Transition(
48     (bn): BatchNorm2d(66, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
49     (conv): Conv2d(66, 33, kernel_size=(1, 1), stride=(1, 1),
bias=False)
50 )
51 (dense_3): Sequential(
52     (0): Bottleneck(
53         (bn_1): BatchNorm2d(33, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
54         (conv_1): Conv2d(33, 48, kernel_size=(1, 1), stride=(1, 1),
bias=False)
55         (bn_2): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
56         (conv_2): Conv2d(48, 12, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
57     )
58     (1): Bottleneck(
59         (bn_1): BatchNorm2d(45, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
60         (conv_1): Conv2d(45, 48, kernel_size=(1, 1), stride=(1, 1),
bias=False)
61         (bn_2): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
62         (conv_2): Conv2d(48, 12, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
63     )
64     (2): Bottleneck(

```



```
65         (bn_1): BatchNorm2d(57, eps=1e-05, momentum=0.1, affine=True,
66         track_running_stats=True)
67         (conv_1): Conv2d(57, 48, kernel_size=(1, 1), stride=(1, 1),
68         bias=False)
69         (bn_2): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,
70         track_running_stats=True)
71         (conv_2): Conv2d(48, 12, kernel_size=(3, 3), stride=(1, 1),
72         padding=(1, 1), bias=False)
73     )
74     )
75     (bn): BatchNorm2d(69, eps=1e-05, momentum=0.1, affine=True,
76     track_running_stats=True)
77     (fc): Linear(in_features=69, out_features=10, bias=True)
78 )
```