卷积神经网络实验报告

姓名：靳乐卿　　学号：2012159

实验要求：

- 掌握卷积的基本原理
- 学会使用 PyTorch 搭建简单的 CNN 实现 Cifar10 数据集分类
- 学会使用 PyTorch 搭建简单的 ResNet 实现 Cifar10 数据集分类
- 学会使用 PyTorch 搭建简单的 DenseNet 实现 Cifar10 数据集分类
- 学会使用 PyTorch 搭建简单的 SE-ResNet 实现 Cifar10 数据集分类

报告内容：

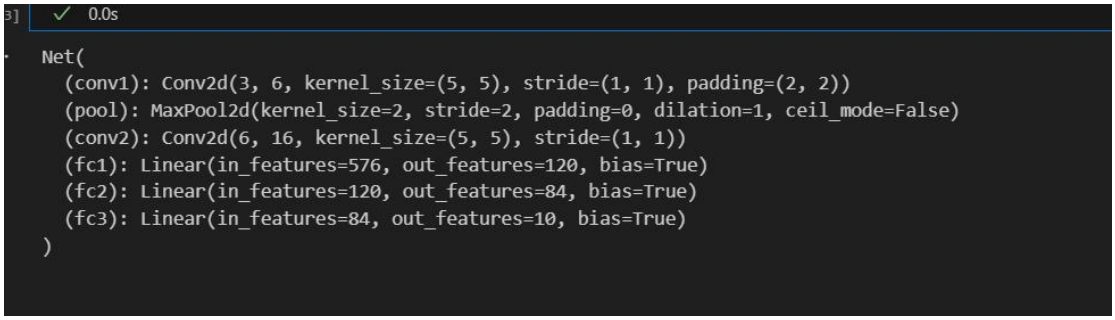
- 老师提供的原始版本 CNN 网络结构（可用 print(net)打印，复制文字或截图皆可）、在 Cifar10 验证集上的训练 loss 曲线、准确度曲线图
- 个人实现的 ResNet 网络结构在上述验证集上的训练 loss 曲线、准确度曲线图
- 个人实现的 DenseNet 网络结构在上述验证集上的训练 loss 曲线、准确度曲线图
- 个人实现的带有 SE 模块（Squeeze-and-Excitation Networks）的 ResNet 网络结构在上述验证集上的训练 loss 曲线、准确度曲线图
- 解释没有跳跃连接的卷积网络、ResNet、DenseNet、SE-ResNet 在训练过程中有什么不同（重点部分）
- 格式不限

作业提交：

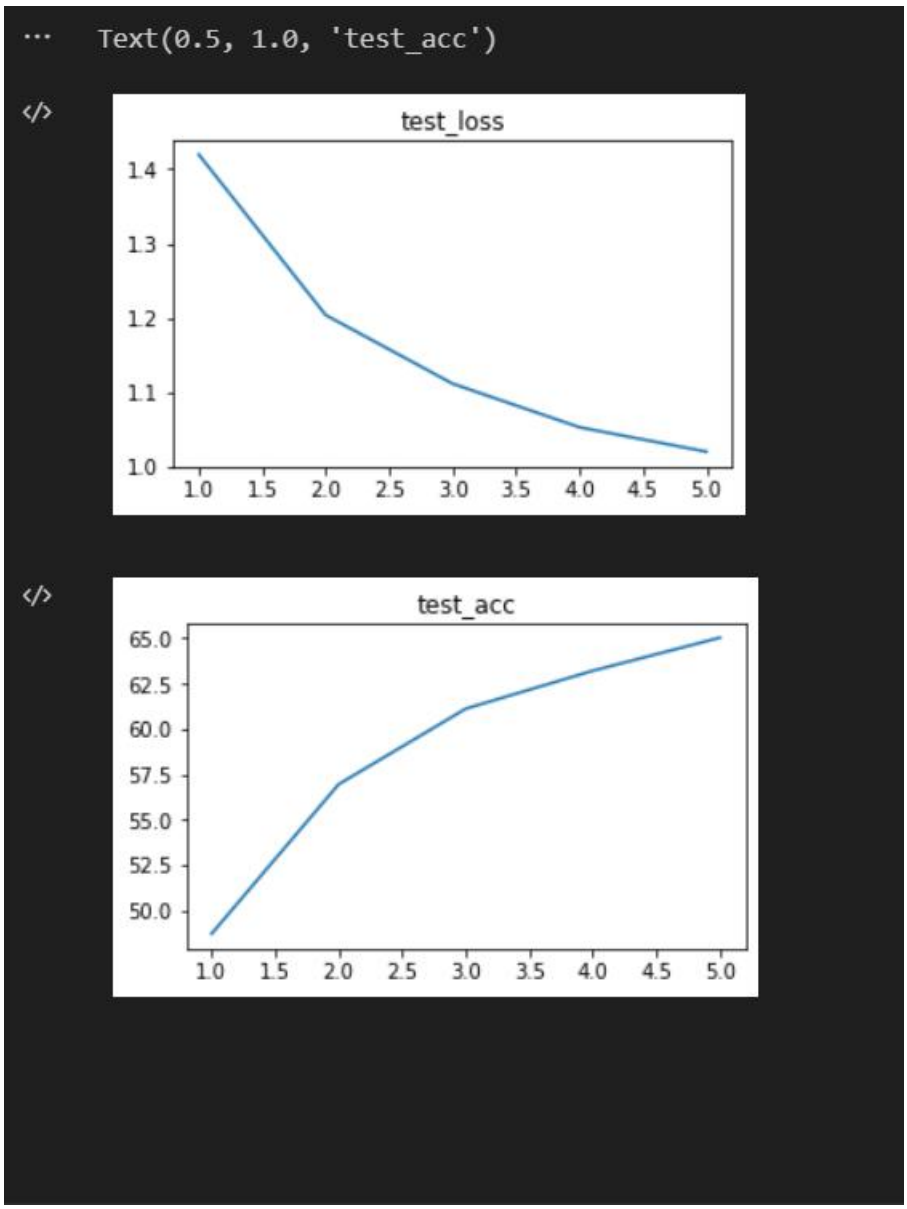
- 期末前将报告和代码（可将 jupyter notebook 里代码复制到一个 xxx.py 文件中）打包（学号+姓名.zip），提交方式另行通知
- 实验报告内容应工整

原始版本 CNN

网络结构



训练 5 轮的曲线



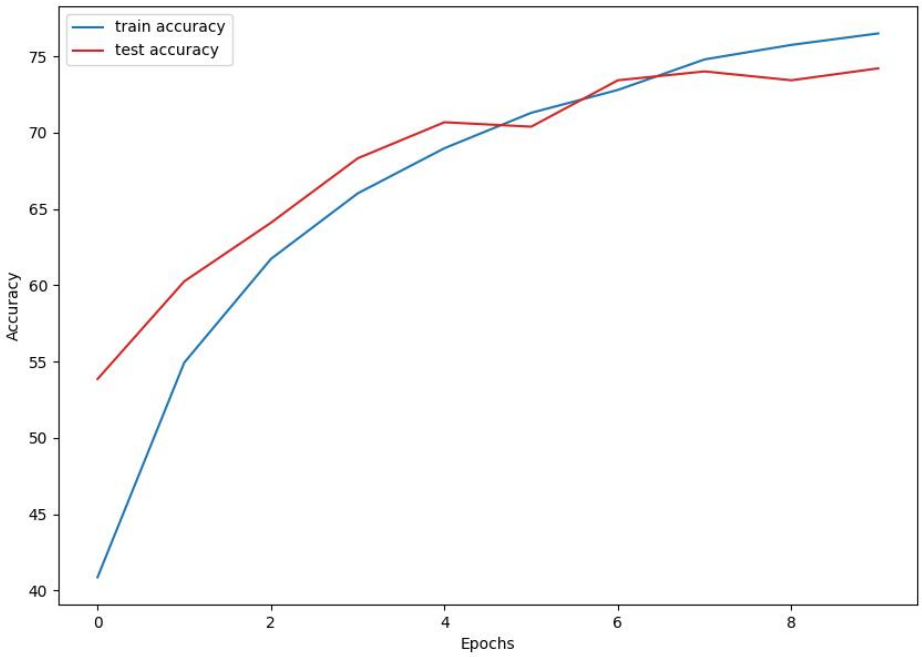
ResNet

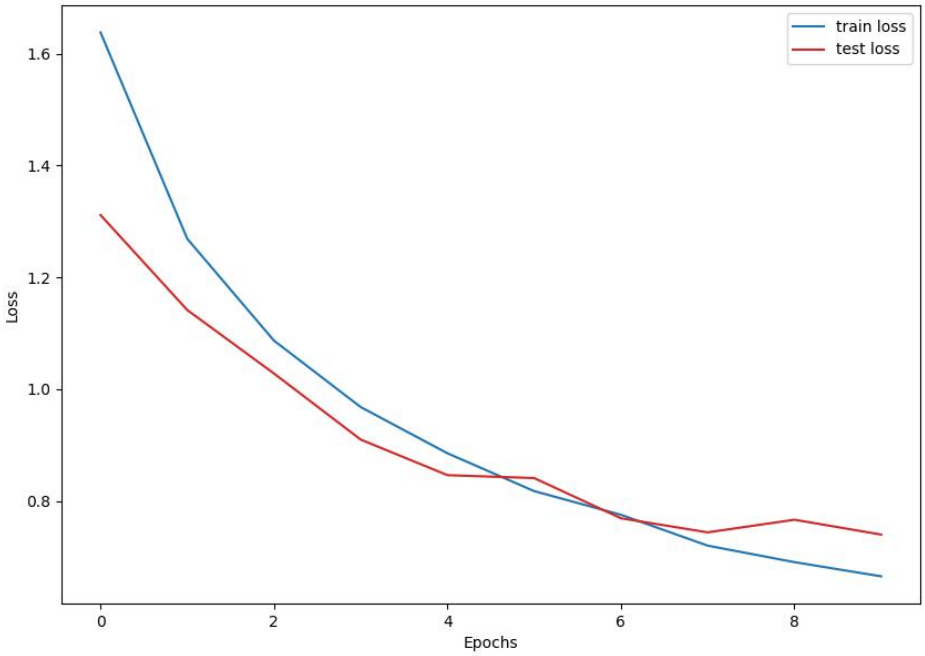
ResNet_normal(
(conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
(bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
(maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
(layer1): Sequential(
(0): BasicBlock(

```
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (layer2): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (layer3): Sequential(
```

```
      (0): BasicBlock(
        (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (layer4): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
```

```
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc): Linear(in_features=512, out_features=10, bias=True)
)
```





DenseNet

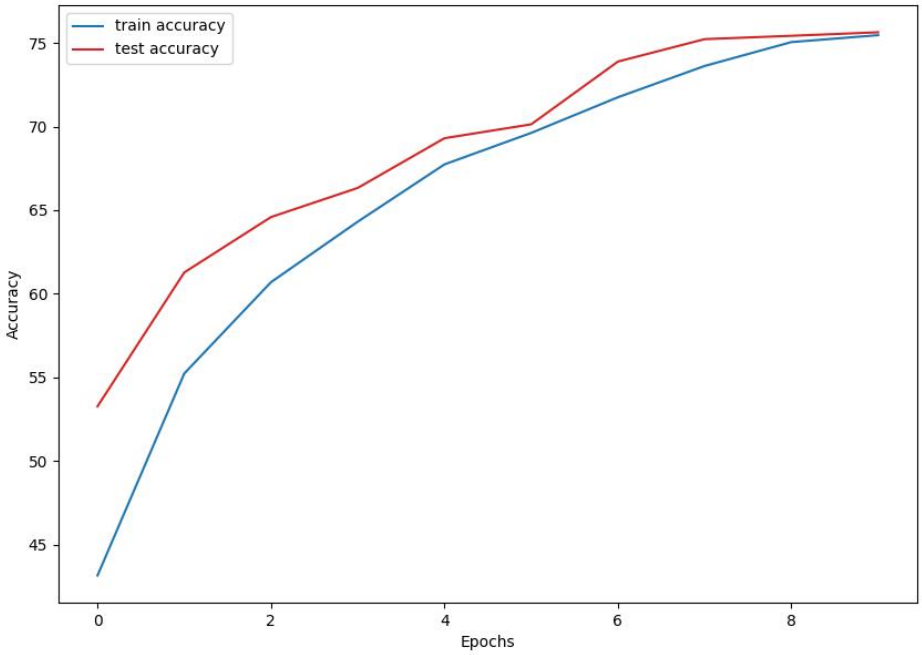
DenseNet(
  (conv): Conv2d(3, 32, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3))
  (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (relu): ReLU()
  (denseblock1): DenseBlock(
    (denseblock): Sequential(
      (0): DenseBasic(
        (layer): Sequential(
          (0): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): ReLU()
          (2): Conv2d(32, 128, kernel_size=(1, 1), stride=(1, 1))
          (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (4): ReLU()
          (5): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
)
)
      (1): DenseBasic(

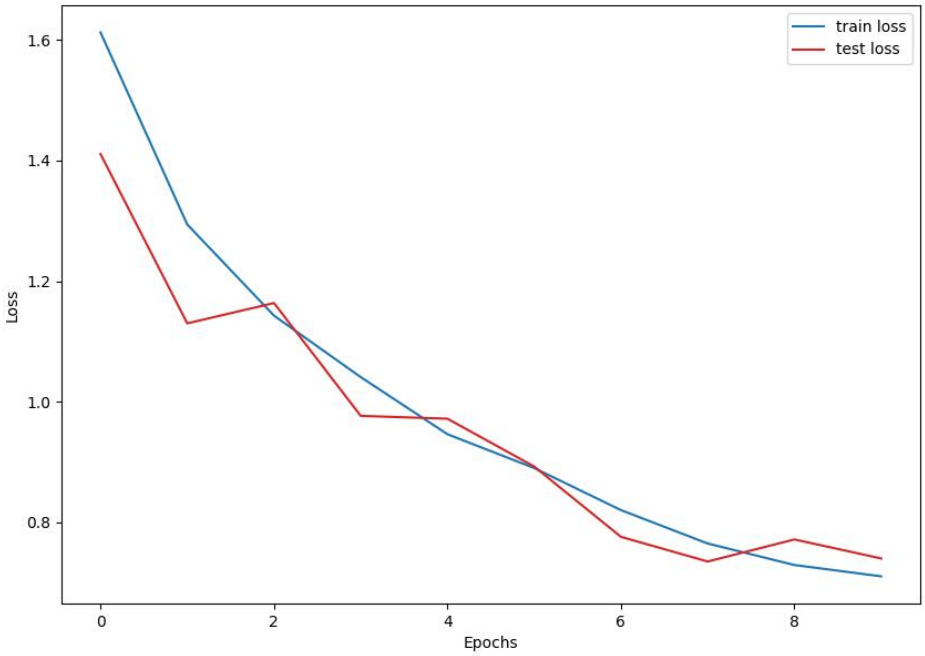
```
            (layer): Sequential(
                (0): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
                (1): ReLU()
                (2): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1))
                (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
                (4): ReLU()
                (5): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
            )
        )
        (2): DenseBasic(
            (layer): Sequential(
                (0): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
                (1): ReLU()
                (2): Conv2d(96, 128, kernel_size=(1, 1), stride=(1, 1))
                (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
                (4): ReLU()
                (5): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
            )
        )
        (3): DenseBasic(
            (layer): Sequential(
                (0): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
                (1): ReLU()
                (2): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
                (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
                (4): ReLU()
                (5): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
            )
        )
        (4): DenseBasic(
            (layer): Sequential(
                (0): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
                (1): ReLU()
                (2): Conv2d(160, 128, kernel_size=(1, 1), stride=(1, 1))
                (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
                (4): ReLU()
```

```
            (5): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          )
        )
        (5): DenseBasic(
          (layer): Sequential(
            (0): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (1): ReLU()
            (2): Conv2d(192, 128, kernel_size=(1, 1), stride=(1, 1))
            (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (4): ReLU()
            (5): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          )
        )
      )
    )
    (bn2): BatchNorm2d(224, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv1): Conv2d(224, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (avg1): AvgPool2d(kernel_size=2, stride=2, padding=0)
    (denseblock2): DenseBlock(
      (denseblock): Sequential(
        (0): DenseBasic(
          (layer): Sequential(
            (0): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (1): ReLU()
            (2): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1))
            (3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (4): ReLU()
            (5): Conv2d(256, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          )
        )
        (1): DenseBasic(
          (layer): Sequential(
            (0): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (1): ReLU()
            (2): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1))
            (3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (4): ReLU()
```

```
            (5): Conv2d(256, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          )
        )
        (2): DenseBasic(
          (layer): Sequential(
            (0): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (1): ReLU()
            (2): Conv2d(192, 256, kernel_size=(1, 1), stride=(1, 1))
            (3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (4): ReLU()
            (5): Conv2d(256, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          )
        )
        (3): DenseBasic(
          (layer): Sequential(
            (0): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (1): ReLU()
            (2): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
            (3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (4): ReLU()
            (5): Conv2d(256, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          )
        )
        (4): DenseBasic(
          (layer): Sequential(
            (0): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (1): ReLU()
            (2): Conv2d(320, 256, kernel_size=(1, 1), stride=(1, 1))
            (3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (4): ReLU()
            (5): Conv2d(256, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          )
        )
        (5): DenseBasic(
          (layer): Sequential(
            (0): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (1): ReLU()
```

```
            (2): Conv2d(384, 256, kernel_size=(1, 1), stride=(1, 1))
            (3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (4): ReLU()
            (5): Conv2d(256, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          )
        )
      )
    )
    (fc1): Linear(in_features=7168, out_features=10, bias=True)
)
```





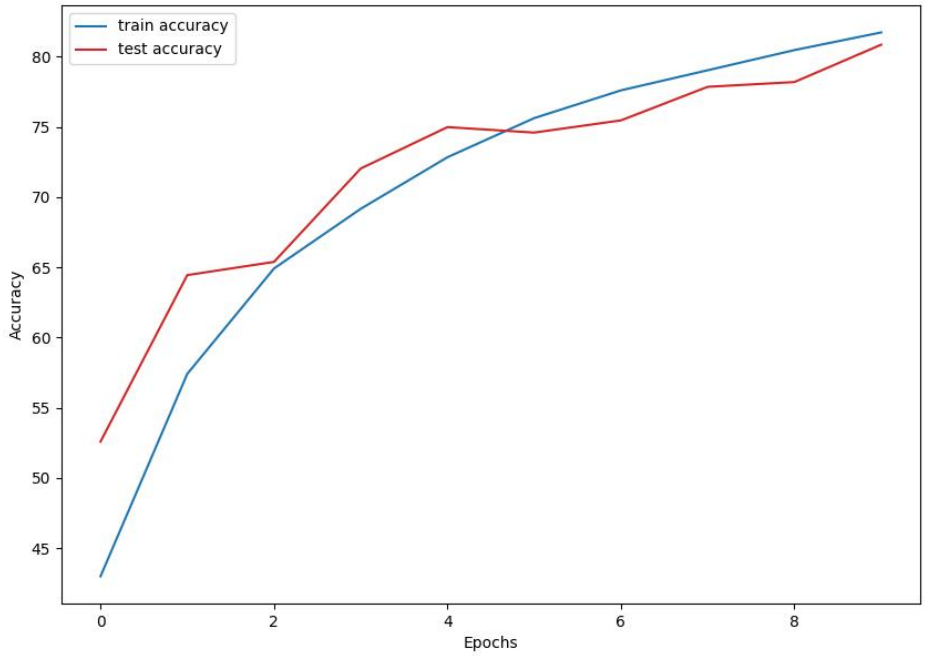
SE-ResNet

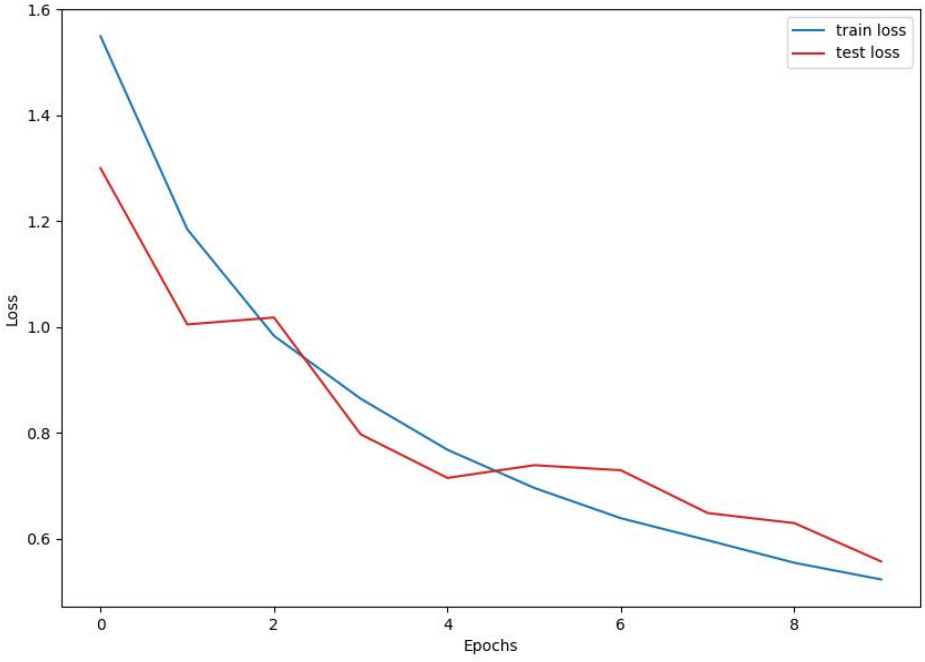
```
SEResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (se): SE(
        (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
        (fc1): Conv2d(64, 4, kernel_size=(1, 1), stride=(1, 1))
        (fc2): Conv2d(4, 64, kernel_size=(1, 1), stride=(1, 1))
      )
    )
    (1): BasicBlock(
```

```
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (se): SE(
          (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
          (fc1): Conv2d(64, 4, kernel_size=(1, 1), stride=(1, 1))
          (fc2): Conv2d(4, 64, kernel_size=(1, 1), stride=(1, 1))
        )
      )
    )
    (layer2): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (se): SE(
          (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
          (fc1): Conv2d(128, 8, kernel_size=(1, 1), stride=(1, 1))
          (fc2): Conv2d(8, 128, kernel_size=(1, 1), stride=(1, 1))
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (se): SE(
          (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
```

```
            (fc1): Conv2d(128, 8, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(8, 128, kernel_size=(1, 1), stride=(1, 1))
          )
        )
      )
      (layer3): Sequential(
        (0): BasicBlock(
          (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
          (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu): ReLU(inplace=True)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (downsample): Sequential(
            (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
            (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (se): SE(
            (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
            (fc1): Conv2d(256, 16, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(16, 256, kernel_size=(1, 1), stride=(1, 1))
          )
        )
        (1): BasicBlock(
          (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu): ReLU(inplace=True)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (se): SE(
            (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
            (fc1): Conv2d(256, 16, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(16, 256, kernel_size=(1, 1), stride=(1, 1))
          )
        )
      )
      (layer4): Sequential(
        (0): BasicBlock(
          (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
          (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (se): SE(
          (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
          (fc1): Conv2d(512, 32, kernel_size=(1, 1), stride=(1, 1))
          (fc2): Conv2d(32, 512, kernel_size=(1, 1), stride=(1, 1))
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (se): SE(
          (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
          (fc1): Conv2d(512, 32, kernel_size=(1, 1), stride=(1, 1))
          (fc2): Conv2d(32, 512, kernel_size=(1, 1), stride=(1, 1))
        )
      )
  )
  (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=10, bias=True)
)
```


train accuracy
test accuracy
Accuracy
Epochs
80
75
70
65
60
55
50
45
0
2
4
6
8


train loss
test loss
Loss
Epochs
1.6
1.4
1.2
1.0
0.8
0.6
0
2
4
6
8

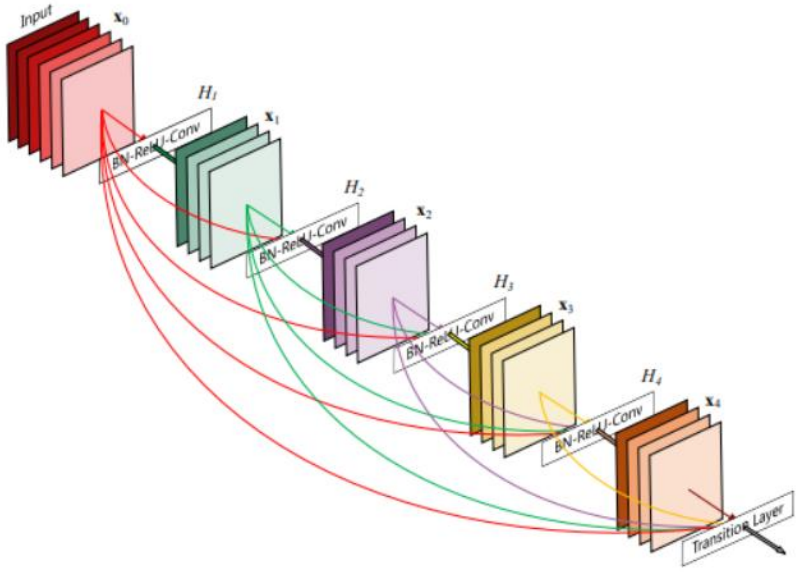
解释训练过程中的不同

resnet

残差网络添加了一个恒等映射的部分，他最终的输出是 f(x)+x 而非 f(x)，他的优势是首先它可以很好的解决梯度消失，因为再如何你还有个 x，不至于梯度消失，这样就可以让你的网络层数更深而仍具有学习能力，可以提高网络的效果。而且还可以比较好的解决网络退化的问题，因为新加入一层最差我们也可以让他是一个恒等映射（或极其接近恒等映射的）层，不会对后续效果有影响，那么自然不会比加入之前差，从而解决网络退化问题

densenet

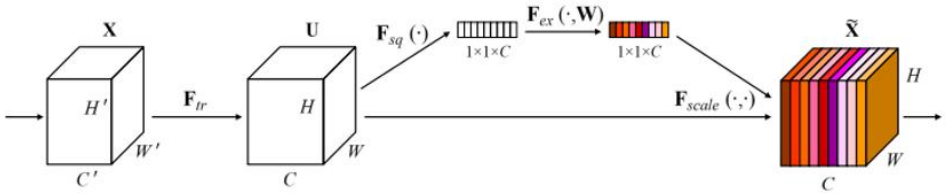
densenet 结构如下图



每个层从前面的所有层获得额外的输入，并将自己的特征映射传递到后续的所有层，使用级联方式。他的思想也来自于 resnet，同样的他也同 resnet 一样，能比较好的解决网络退化和梯度消失的问题，不过比起 resnet 不同，resnet 只接受上一层的输入，densenet 接受不止上一层的输入，此外，densenet 是将输出和残差进行拼接，而非相加。
而且虽然看起来更密集的连接会大大增加参数量，但实际上 DenseNet 比传统的卷积网络所需要的参数反而更少，因为密集的连接带来了特征重用，不需要重新学习冗余的特征图，而且维度拼接的操作，带来了丰富的特征信息，利用更少的卷积就能获得很多的特征图。

senet

se 模块如下图



其工作机理如下:
我们把得到的一系列特征图（通道），压缩成 1*1 大小（使用全局平均池化），然后再通过可学习的参数得到每个特征图（通道）的权重（使用全连接层去训练，也是有一个压缩，解压的过程先把通道数压缩到 in_channel/ratio，然后再给他回到 in_channel 通道数），最后和原本的特征图进行相乘，相当于加权运算。
这可以用较少的参数量，显式的构建对各个特征图（通道）的重要性的学习，以期得到对图像的更好的学习效果