研究生《深度学习》课程

实验报告

实验名称：	实验 3 卷积神经网络实验
姓　　名：	郑楚彬
学　　号：	21140129
上课类型：	专业课
日　　期：	2021.09.18

一、实验内容

二维卷积实验、空洞卷积实验、残差网络实验

二、实验设计

三、实验环境及实验数据集

1. 实验环境

Ubuntu 18.04、CPU 4 核、32GB、Pytorch 1.6.0、Jupyter Notebook

2. 数据集：**车辆分类数据**

- 输入图片，输出对应的类别
- 共 1358 张车辆图片
- 分别属于汽车、客车和货车三类
- 汽车:779 张、客车:218 张、货车:360 张
- 每个类别的后 20-30%当作测试集
- 图片的大小不一，需要将图片拉伸到相同大小
- 输入图片，输出对应的类别
- 预处理

3. 预处理：准备训练集、测试集

通过对图片变形、转化为 Numpy 格式、归一化，并按 25%的比例划分测试集，得到如下的数据集，其中，训练集 1017 张，测试集 340 张。

```
torch.Size([1017, 3, 100, 120]) torch.Size([1017])
torch.Size([340, 3, 100, 120]) torch.Size([340])
label2index: {'bus': 0, 'truck': 1, 'car': 2}
```

四、实验过程

1. 二维卷积实验

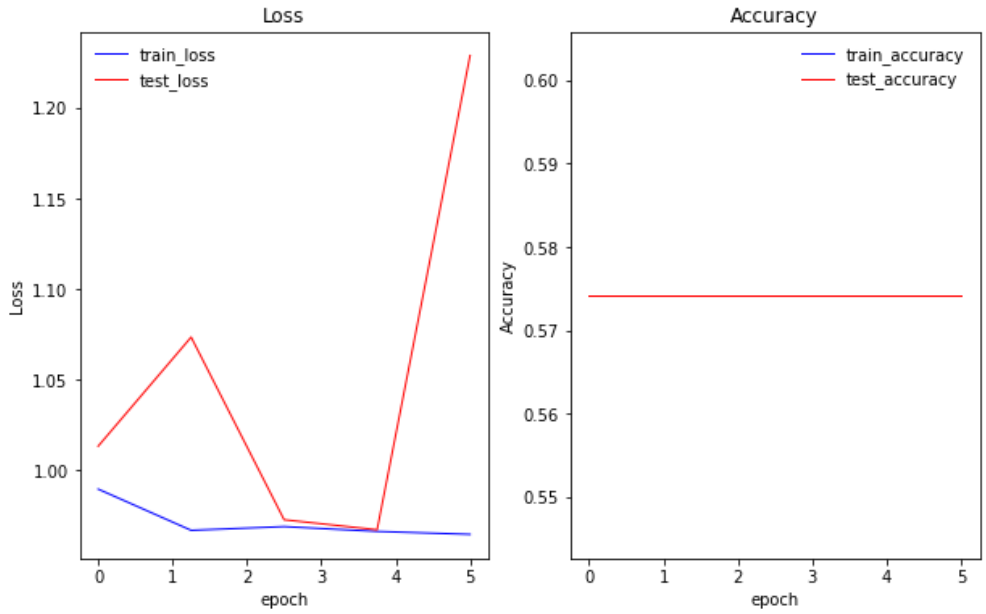
1.1 手写二维卷积

- 模型结构

```
MyConvModule(
  (conv): Sequential(
    (0): MyConv2D()
    (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (fc): Linear(in_features=8, out_features=3, bias=True)
)
```

□ 实验结果

训练速度异常的慢，1 轮耗时大于 1.5 小时，主要在于卷积操作使用循环，而不是矩阵操作。准确率基本维持在 60%上下，损失在 1.0 左右。



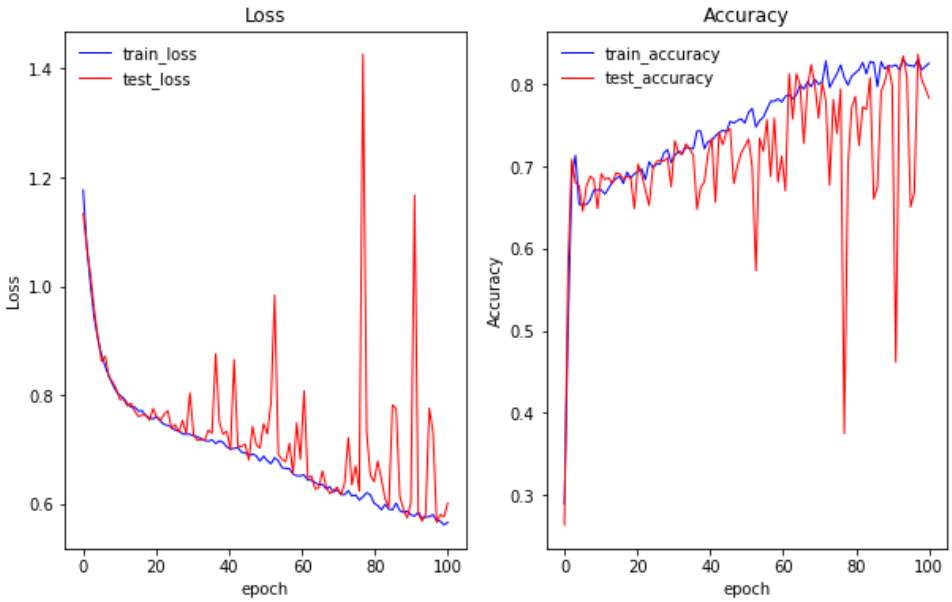
1.2 PyTorch 实现二维卷积

□ 模型结构

```
ConvModule(
  (conv): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (fc): Linear(in_features=16, out_features=3, bias=True)
)
```

□ 实验结果

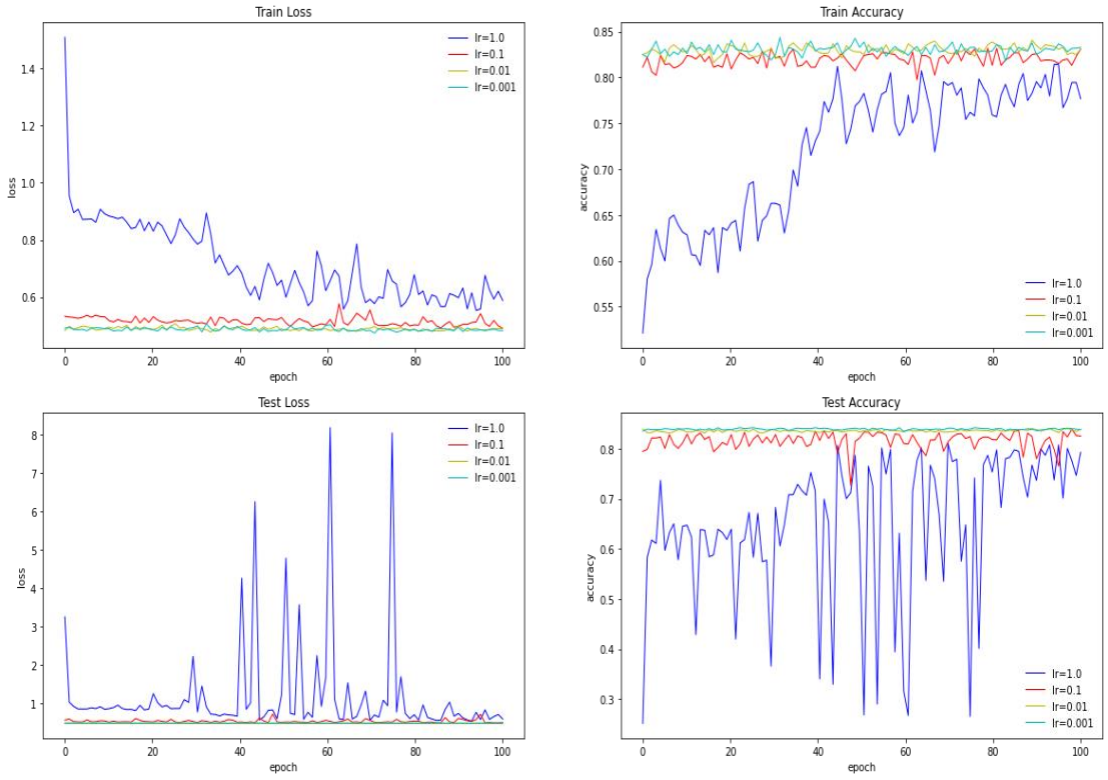
训练时间上 PyTorh 实现的二维卷积速度非常快，100 个迭代 20 分钟内就跑完了；训练集上的 loss 持续下降、accuracy 持续上升，而在测试集上，loss 和 accuracy 均会出现震荡的情况。



1.3 不同超参数的对比分析

在这里，主要对学习率(LR)该参数进行了对照实验，在相同条件下，分别设置了 LR=1.0、0.1、0.01、0.001。其中，**蓝色**线条代表 lr=1.0、**红色**线条代表 lr=0.1、**黄色**线条代表 lr=0.01、**青色**线条代表 lr=0.001。

显而易见，无论是测试集还是训练集、无论是 loss 还是 accuracy，学习率越高，曲线越不稳定；而当 LR=0.01 或 0.001 时，两者的曲线都较为平稳。



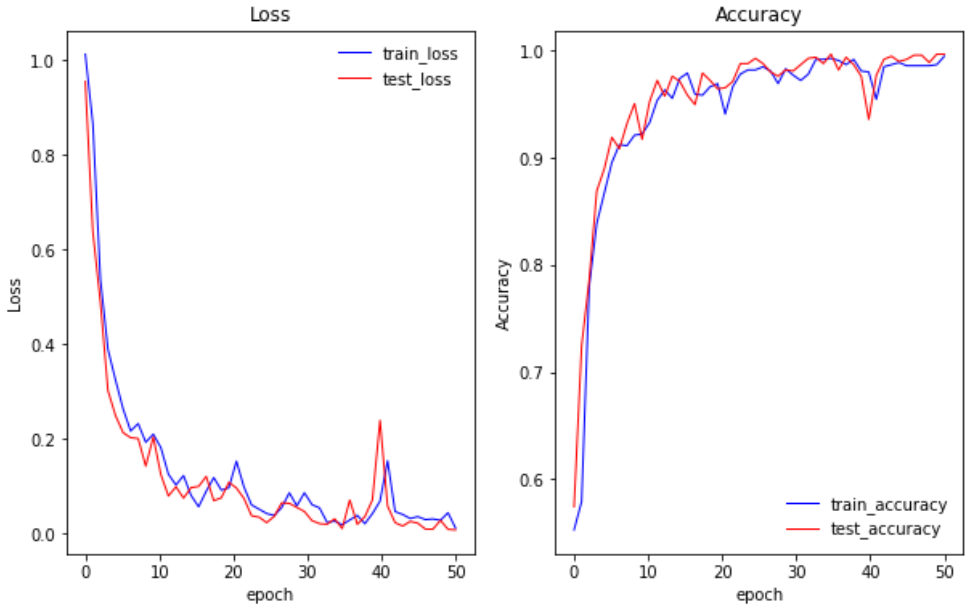
1.4 PyTorch 实现经典模型 AlexNet

□ 模型结构

```
AlexNetModule(
  (conv): Sequential(
    (0): Conv2d(3, 48, kernel_size=(11, 11), stride=(4, 4))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(48, 128, kernel_size=(5, 5), stride=(1, 1), padding=(1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(128, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
  )
  (fc): Sequential(
    (0): Linear(in_features=3840, out_features=2048, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=2048, out_features=2048, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=2048, out_features=3, bias=True)
  )
)
```

□ 实验结果

AlexNet 训练的速度很快，50 个迭代 20 分钟内就跑完了；并且在训练集和测试集上的表现相当不俗，测试集上的预测准确率高达 99%。



3. 空洞卷积实验

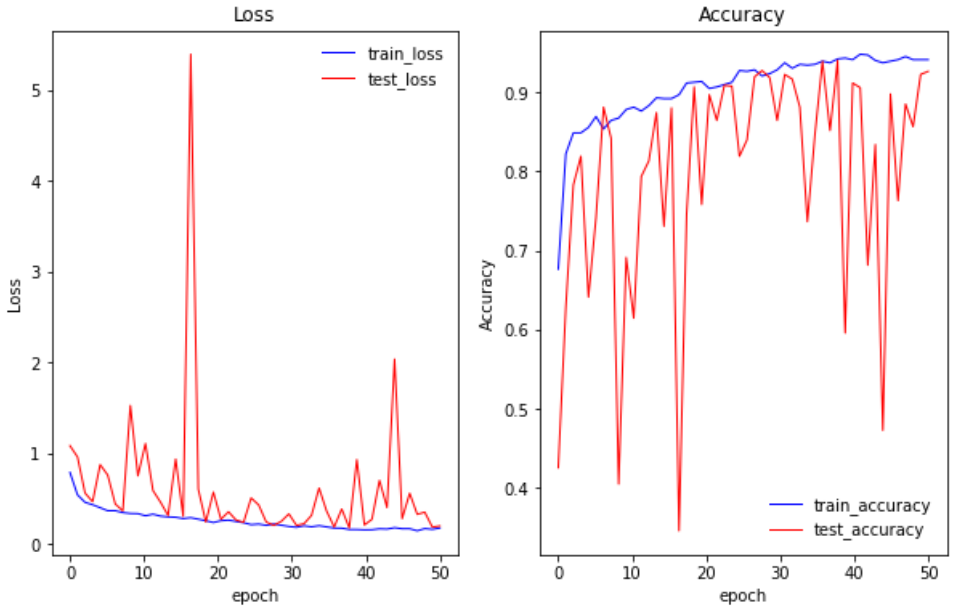
3.1 PyTorch 实现空洞卷积

□ 模型结构

```
DilatedConvModule(
  (conv): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), dilation=(2, 2))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), dilation=(5, 5))
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
  )
  (fc): Linear(in_features=128, out_features=3, bias=True)
)
```

□ 实验结果

空洞卷积训练的速度慢，50 个迭代需要将近 250 分钟，整体表现一般，存在 loss、accuracy 均不稳定，时高时低。



□ 与卷积模型(AlexNet)的结果比对

（1）训练时间上远超 AlexNet

（2）准确率远低于 AlexNet

（3）整体而言，在该任务中，空洞卷积的性能远不如卷积模型。

4. 残差网络实验

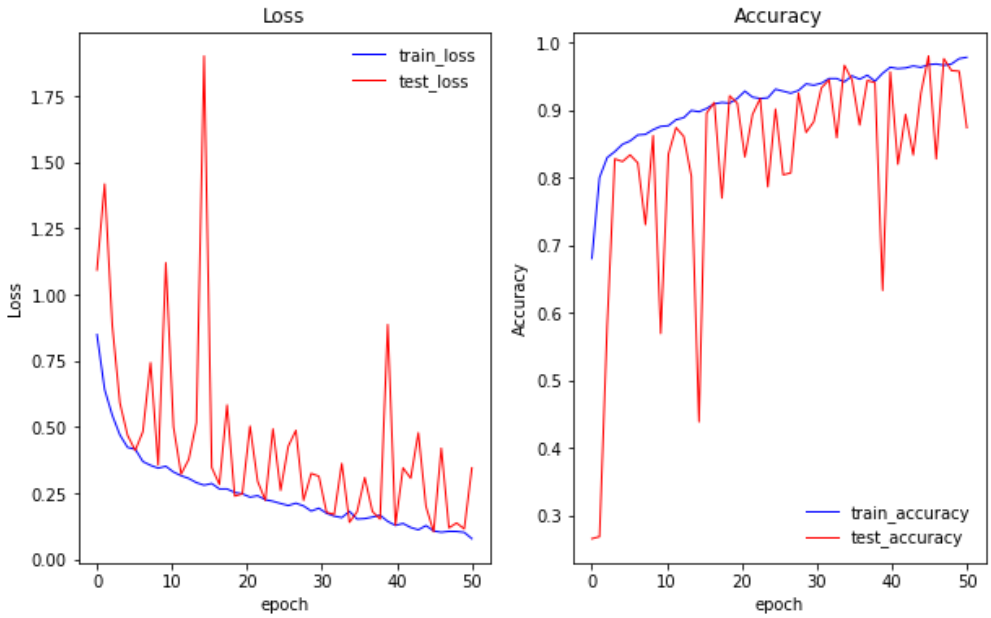
4.1 PyTorch 实现残差网络

□ 模型结构

```
ResNetsModule(
  (conv1): ResidualBlock(
    (left): Sequential(
      (0): Conv2d(3, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(4, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (4): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (short_cut): Sequential(
      (0): Conv2d(3, 4, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (conv2): ResidualBlock(
    (left): Sequential(
      (0): Conv2d(4, 8, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (4): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (short_cut): Sequential(
      (0): Conv2d(4, 8, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (conv3): ResidualBlock(
    (left): Sequential(
      (0): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (4): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (short_cut): Sequential(
      (0): Conv2d(8, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (conv4): ResidualBlock(
    (left): Sequential(
      (0): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (short_cut): Sequential(
      (0): Conv2d(16, 32, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (fc): Linear(in_features=32, out_features=3, bias=True)
)
```

□ 实验结果

残差网络训练的时间不长，50 个迭代需要将近 1 小时，整体表现不错，准确率达到 95%左右。



五、实验结果

六、实验心得体会

本次实验加深了对 CNN 相关模型的理解，从零实现了 CV 领域的分类任务，不仅学习到了如何从图片转化为张量、并被模型所捕捉特征及学习，还进一步探究了几个举足轻重的模型，如 AlexNet、空洞卷积、残差网络。整体而言，获益匪浅。

七、参考文献

八、附录

需要补充说明的内容，如无可略。

实验报告编写要求

1. 正文要求小四号宋体，行间距 1.5 倍；

2. 英文要求小四号 Times New Roman；

3. 在实验内容、实验过程、实验结果三部分需要针对当次实验不同的实验内容分别填写（模版以实验一为例），实验设计中如有必要也可以分开填写；

4. 实验报告配图的每幅图应有编号和标题，编号和标题应位于图下方处，居中，中文用五号宋体；

5. 表格应为三线表，每个表格应有编号和标题，编号和标题应写在表格上方正中，距正文段前 0.5 倍行距。表格中量与单位之间用“/”分隔，编号与标题中的中文用五号宋体；

6. 图、表、公式、算式等，一律用阿拉伯数字分别依序连续编排序号。其标注形式应便于互相区别，可分别为：图 1、表 2、公式(5)等。