中国科学院大学
University of Chinese Academy of Sciences

课程作业报告

基于卷积神经网络的猫狗分类

作者姓名：段宏键

学科专业：计算机系统结构

所在单位：中国科学院大学计算机科学与技术学院

2020 年 6 月

1. 摘要

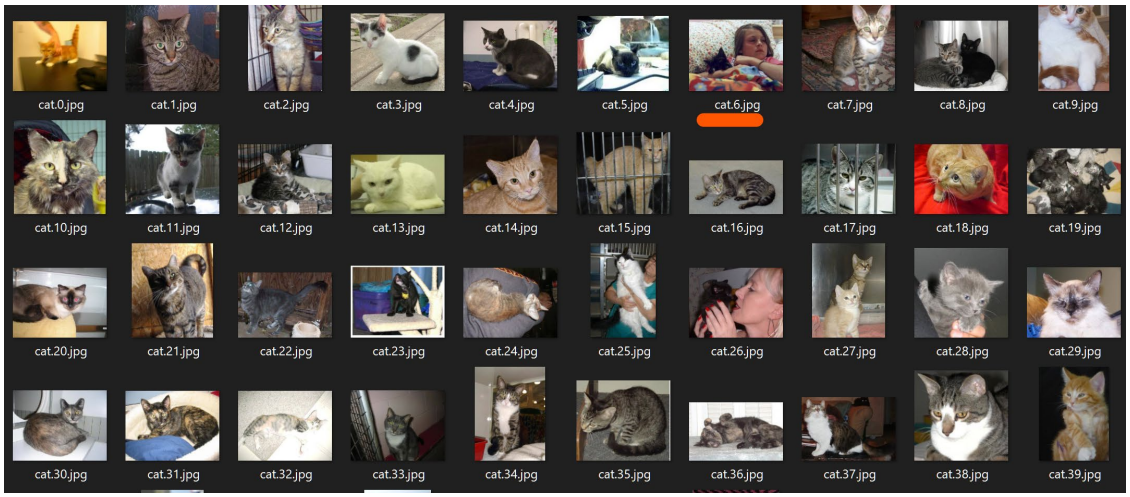
本项目是为了实现对猫狗图像的分类。本项目设计并实现了一个轻量级的 CNN 网络架构（三层的卷积，ReLU，最大池化后跟三层的 ReLU 激活的全连接层），同时在训练时采用了训练数据数值归一化以及 dropout 的策略，使用 kaggle 提供的猫狗照片数据集，经过 10 轮迭代在训练数据上达到了 77.62%的识别准确率。同时本项目还对比了两个卷积层网络与三个卷积层网络的准确率差别。

代码设计采用训练（train.py）与测试（test.py）分离的模式，训练代码会保存模型的参数，测试代码加载参数能在本地快速完成测试工作。

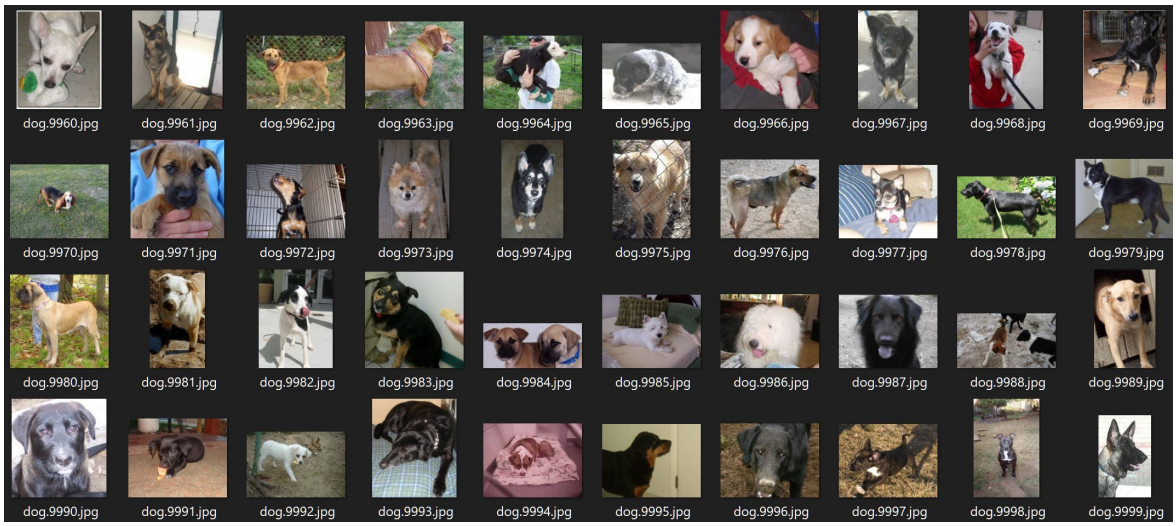
1. 介绍

(1)猫狗识别是一个传统的二分类问题，其标准的训练集包含 25000 张图片，均放置在同一文件夹下，命名格式为<category>.<num>.jpg，如 cat.10000.jpg、dog.100.jpg，测试集包含 12500 张图片，命名为<num>.jpg，如 1000.jpg。对于 kaggle 比赛的参与者来说需要根据训练集的图片训练模型，并在测试集上进行预测，输出它是狗的概率，最后提交 csv 文件。但对于我们这个课程项目来说，由于测试集没有 label，所以需要重新划分标准训练集，把其中一部分图片用作测试，这样才能完整的完成这个项目。

小猫图片数据：



小狗图片数据：



简单的观察数据集就很容易发现，小猫小狗姿态不一。同时，图片尺寸也不一致，有的是竖放的长方形，有的是横放的长方形。我们最终需要固定尺寸的输入数据，所以需要进行图片处理。

(2) 首先，对于我们这个项目来讲，为了使测试集中的数据也有标签，我把标准训练集（25000 张，猫狗各半）重新划分成为了新的训练集（20000 张，猫狗各半）和测试集（5000 张，猫狗各半）。

其次对于输入数据尺寸不规则以及图片的标签（猫、狗）的问题，我没有使用课程提供的指导文档《实验二 猫狗分类实验指导书》中用 numpy 来加载数据，而是重写了 pytorch 的 torch.utils.data.Dataset，根据图片的名字来解析其 label，这样可以乱序的载入图片进行训练，同时在加载图片的时候将图像按比例缩放至合适尺寸，同时做了一些裁切。

和手写数字识别的项目一样，采用卷积神经网络来完成图片特征提取与分类。卷积神经网络是典型的多层神经网络，擅长处理图像特别是大图像的相关机器学习问题。卷积神经网络通过一系列的方法，成功地将大数据量的图像识别问题不断降维，最终使其能够被训练。CNN 最早由 Yann LeCun 提出并应用在手写体识别上。

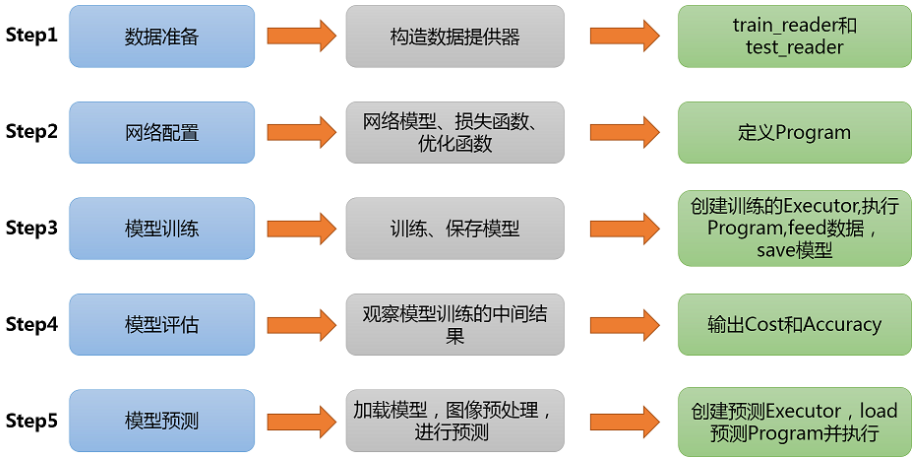
(3) 和手写数字识别的项目类似，我仍然采用类似 LeNet-5 的结构来构造我网络，使用第一个卷积层来提取图片中细粒度的特征，使用第二个卷积层来把细粒度特征汇总会比较大的特征，使用第三层卷积来提取更高层次的特征。同时每次卷积之后均使用 ReLU 和最大池化操作来减少特征维度，突出重要特征。训练

过程中开启 dropout，防止网络过拟合。最后使用三个全连接层来根据这些特征进行分类。其中训练程序（train.py）会保存训练模型，测试程序(test.py)可以直接加载模型参数来进行准确率验证，最后的准确率能达到 77.62%。 我也对两层卷积的神经网络进行了测试，发现其最后测试识别准确率在 76.8%，其实与三层网络差不太多。

2. 解决方案

实验代码采用了模块化设计的方法，并且把模型训练与测试分开，这样能解耦代码结构，同时也能在把训练模型得到的模型数据保存，以便测试代码能在弱计算环境下顺利测试，而训练代码则可以使用云平台或其他平台。同时模型的训练与测试均兼容了纯 CPU 平台与 CPU+GPU（cuda）平台。

整个实验过程遵循以下几个步骤：



数据准备阶段是把 20000 张数据（0-9999cat、0-9999dog）作为了训练数据（“data/train”文件夹），把 5000 张（10000-12499cat、10000-12499dog）数据作为了训练集（“data/test”文件夹）。并实现了对载入数据的尺寸统一以及归一化，以及裁切（图像中心裁剪合适大小的图像）。数据的 batch size 是 16

网络设计采用了 卷积+ ReLU + MaxPooling + 卷积 + ReLU + MaxPooling +卷积+ ReLU + MaxPooling+三层 ReLU 激活的线性全连接层，详细的网络结构设计如下：

```
Net(
```

```
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1,
 1))
  (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1
, 1))
  (conv3): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1
, 1))
  (fc1): Linear(in_features=10000, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=2, bias=True)
)
```

与前述手写数字识别网络不同，此处没有使用 pytorch 的序列工具来快速构建。

训练优化方法与损失函数如下：

```
    optimizer = torch.optim.Adam(model.parameters(), lr=lr)

    criterion = torch.nn.CrossEntropyLoss()
```

损失函数的设计采用交叉熵损失函数，优化方法是 Adam。

设置了学习率为 0.0001，学习率采用固定模式。一共训练迭代了 10 轮。

具体 train 的设计可以参见代码。

测试部分的设计与训练部分很相似，只是加载的数据集变为了测试数据集，同时去掉了误差反向传播的过程。

3. 实验结果和分析

实验配置如下：

深度学习平台：anaconda + pytorch（具体环境包依赖可以参见实验目录下的“环境依赖.txt”）

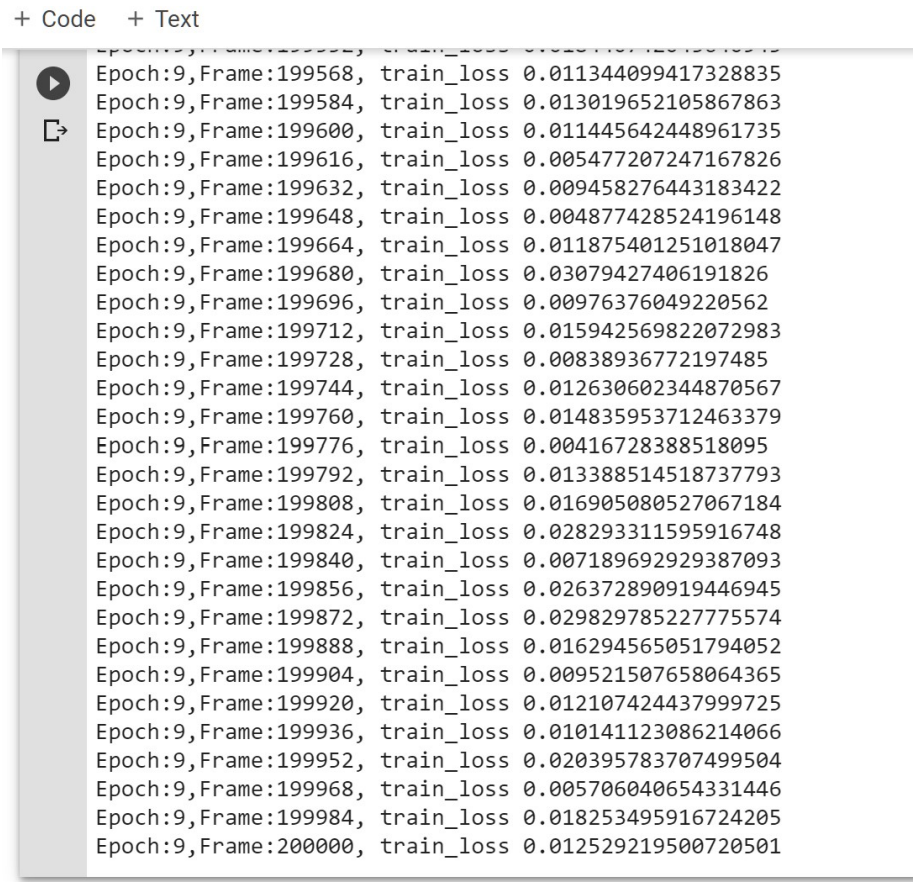
数据集：Kaggle 提供的猫狗分类的训练集按照 4：1 重新划分为训练集和测试集。

训练环境：google colaboratory(训练使用的 runtime 是 GPU 类型)

测试环境：自己的笔记本电脑，无 GPU

训练过程中会定时输出 loss 值，并在最后保存训练模型的参数以便用于测试。

两层卷积网络模型训练过程中的 loss：



最后在训练 10 轮使用两层卷积网络的测试结果如下：

```
processed [3520/5000] images, now the correctness is 0.7659
processed [3680/5000] images, now the correctness is 0.7666
processed [3840/5000] images, now the correctness is 0.7667
processed [4000/5000] images, now the correctness is 0.7662
processed [4160/5000] images, now the correctness is 0.7675
processed [4320/5000] images, now the correctness is 0.7664
processed [4480/5000] images, now the correctness is 0.7676
processed [4640/5000] images, now the correctness is 0.7690
processed [4800/5000] images, now the correctness is 0.7690
processed [4960/5000] images, now the correctness is 0.7677
end...
the final correctness in test dataset is  0.768
```

三层卷积网络模型训练过程中的 loss 与最后在训练 10 轮使用三层卷积网络的测试结果如下：

+ Code + Text

```
Epoch:9,Frame:199632, train_loss 0.01425907202064991
Epoch:9,Frame:199648, train_loss 0.025241835042834282
Epoch:9,Frame:199664, train_loss 0.017954949289560318
Epoch:9,Frame:199680, train_loss 0.05597168952226639
Epoch:9,Frame:199696, train_loss 0.02052813209593296
Epoch:9,Frame:199712, train_loss 0.03060639649629593
Epoch:9,Frame:199728, train_loss 0.012496637180447578
Epoch:9,Frame:199744, train_loss 0.023524943739175797
Epoch:9,Frame:199760, train_loss 0.030736997723579407
Epoch:9,Frame:199776, train_loss 0.0337182879447937
Epoch:9,Frame:199792, train_loss 0.021347830072045326
Epoch:9,Frame:199808, train_loss 0.016564082354307175
Epoch:9,Frame:199824, train_loss 0.026270892471075058
Epoch:9,Frame:199840, train_loss 0.021446743980050087
Epoch:9,Frame:199856, train_loss 0.0552099384367466
Epoch:9,Frame:199872, train_loss 0.02592713199555874
Epoch:9,Frame:199888, train_loss 0.02895507402718067
Epoch:9,Frame:199904, train_loss 0.011836763471364975
Epoch:9,Frame:199920, train_loss 0.02815663069486618
Epoch:9,Frame:199936, train_loss 0.012736891396343708
Epoch:9,Frame:199952, train_loss 0.025967177003622055
Epoch:9,Frame:199968, train_loss 0.028110457584261894
Epoch:9,Frame:199984, train_loss 0.021473992615938187
Epoch:9,Frame:200000, train_loss 0.017641404643654823
```

```
Dataset loaded! length of train set is 5000
processed [160/5000] images, now the correctness is 0.7937
processed [320/5000] images, now the correctness is 0.7906
processed [480/5000] images, now the correctness is 0.8063
processed [640/5000] images, now the correctness is 0.7922
processed [800/5000] images, now the correctness is 0.7788
processed [960/5000] images, now the correctness is 0.7750
processed [1120/5000] images, now the correctness is 0.7741
processed [1280/5000] images, now the correctness is 0.7789
processed [1440/5000] images, now the correctness is 0.7778
processed [1600/5000] images, now the correctness is 0.7806
processed [1760/5000] images, now the correctness is 0.7773
processed [1920/5000] images, now the correctness is 0.7719
processed [2080/5000] images, now the correctness is 0.7688
processed [2240/5000] images, now the correctness is 0.7719
processed [2400/5000] images, now the correctness is 0.7738
processed [2560/5000] images, now the correctness is 0.7727
processed [2720/5000] images, now the correctness is 0.7724
processed [2880/5000] images, now the correctness is 0.7750
processed [3040/5000] images, now the correctness is 0.7753
processed [3200/5000] images, now the correctness is 0.7756
processed [3360/5000] images, now the correctness is 0.7759
processed [3520/5000] images, now the correctness is 0.7756
processed [3680/5000] images, now the correctness is 0.7755
processed [3840/5000] images, now the correctness is 0.7768
processed [4000/5000] images, now the correctness is 0.7795
processed [4160/5000] images, now the correctness is 0.7776
processed [4320/5000] images, now the correctness is 0.7769
processed [4480/5000] images, now the correctness is 0.7766
processed [4640/5000] images, now the correctness is 0.7759
processed [4800/5000] images, now the correctness is 0.7762
processed [4960/5000] images, now the correctness is 0.7758
end...
the final correctness in test dataset is  0.7762
```

除此之外，我观察到在两层和三层网络上训练出来的模型最后准确率相差不多。这个原因还有待进一步探究。

4. 结论

这个实验我分别使用两层卷积与三层卷积的卷积神经网络对在猫狗各 1 万张的图片上进行了训练，在身下的 5000 张猫狗图片上做了测试。依次按照数据准备、网络构建、模型训练、模型评估以及模型测试这一系列步骤来完成这个任务。这是本课程中我完成的第二个课程项目，代码按照训练与测试分离解耦的方式组织，既方便了线上的训练，也方便了线下的测试。同时模块性相比手写数字识别要变好了一点，上述各个过程均封装到函数中了。同时通过本实验我进一步理解了卷积神经网络的作用，同时发现在我的实验设置的情况下，两层与三层卷积网络对最终的结果影响不大。

5. 参考文献

[1] 课程资源《实验三猫狗分类实验指导书》
[2] Belongie, Serge, Jitendra Malik, and Jan Puzicha. "Shape matching and object recognition using shape contexts." IEEE transactions on pattern analysis and machine intelligence 24.4 (2002): 509-522.