模式识别大作业

Caltech256 图像分类

姓名：罗雁天

院系：清华大学电子系

学号：2018310742

日期：June 3, 2019

目录

第 1 章 引言

Caltech 256 数据集是加利福尼亚理工学院收集整理的数据集，该数据集选自 Google Image 数据集，并手工去除了不符合其类别的图片。在该数据集中，图片被分为 256 类，每个类别的图片超过 80 张。图1.1展示了 Caltech 256 数据集中其中 20 类图片的示例。

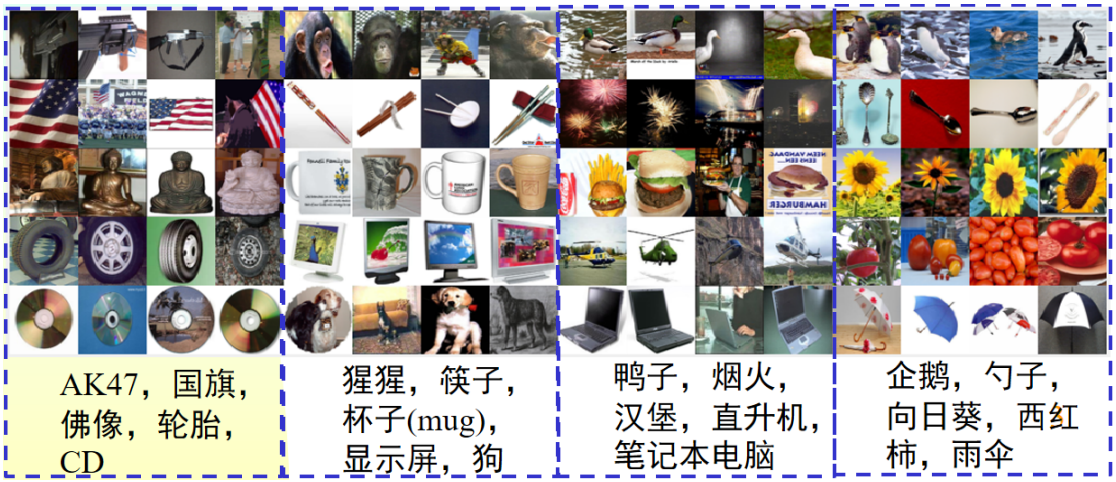

图 1.1: Caltech 256 数据集中其中 20 类图片示例

本次大作业在 Caltech 256 数据集的基础上复习和熟悉模式识别课上学习到的相关算法，完成如下几部分内容：

- PCA/Fisher 图像分类。采用 PCA 方法或 Fisher 线性判别准则进行特征变换，然后使用 K 近邻分类算法对选出的 20 类图片进行分类，分析选取不同主分量个数对识别率和虚警率的影响；并且进行开集测试；
- 使用其他特征和方法进行图像分类，比如 HOG 特征 +SVM 分类，同样进行开集测试，并比较以上两种方法的优缺点；
- (选做 1) 在全体 257 类图像上使用以上两种方法进行分类；
- (选做 2) 用深度学习的方法在原始数据集上进行图像分类；

第 2 章 PCA 降维 +KNN 分类

2.1 算法描述

在本部分我们使用图像的像素值作为特征，使用 PCA 进行降维，然后再使用 KNN 进行分类。由于数据集中的图片大小不全相同，图像类型也不尽相同（有灰度图和彩色图），因此，在我们使用 PCA 降维之前需要首先对图像进行预处理，预处理的算法如 Algorithm 1所示。

Algorithm 1 图像预处理算法

输入： 原始图片 X
输出： 预处理之后的图片向量 X'
1: 首先将图片 resize 到统一大小 224×224；
2: 归一化处理 $X = X/255$，将像素值归一化到 $[0, 1]$ 区间内；
3: 将单通道的灰度图通过复制通道变为 3 通道的图，彩色图不变；
4: 最后将像素值拉平成一个向量，方便之后的 PCA，$X' = X.flatten()$.

在图像预处理之后，由于我们处理之后的图像为 $224 \times 224 \times 3$，特征数量很多，直接使用 KNN 进行分类非常复杂并且耗时较多，并且在这么多的像素中有些像素肯定是对分类无关紧要的，因此，我们采用 PCA 算法首先对预处理之后的数据集进行降维，如 Algorithm 2所示。

Algorithm 2 图像 PCA 降维

输入： 预处理之后的训练集图片向量 $X \in \mathbb{R}^{N \times 50176}$，主成分数量 K
输出： PCA 降维重建之后的 $X' \in \mathbb{R}^{N \times k}$
1: 将 X 进行中心化 $X = X - mean(X, 1)$
2: 计算 X 的协方差矩阵 $\Sigma = X^T X$；
3: 对 Σ 进行特征值分解；
4: 将前 K 个最大特征值对应的特征向量组成投影矩阵 W；
5: 计算投影 $y = W^T(X - \mu)$;
6: 计算重建向量 $x' = Wy + \mu$;

在 PCA 降维之后，我们使用 KNN 算法进行分类，如 Algorithm 3所示。

2.2 实验结果

由于在此问题中，我们使用像素值作为特征，这样的话特征数量大于样本数量，所以我们设置主成分分量的时候要小于样本数量。在此，我们设置主成分分

Algorithm 3 KNN 分类

输入： 训练集 (X,Y)，测试数据 (z)
输出： 测试数据的类别

初始化 k 近邻的集合 *knnlist* 为前 k 个点，与 k 近邻距离的集合 *distlist* 为到前 k 个点的距离
for $x \in X$ **do**
    计算距离最大的近邻为 *maxk*，最大距离为 *maxdist*
    计算 x,z 之间的距离 *dist*
    if $dist < maxdist$ **then**
        将 x 加入 *knnlist* 并将 *maxk* 在 *knnlist* 里删掉；
        将 *dist* 加入 *distlist* 并将 *maxdist* 在 *knnlist* 里删掉；
    end if
end for
统计 k 个最近邻样本中每个类别出现的次数
选择出现频率最大的类别作为未知样本的类别

量为 1 : 5 : 1000，观察对于不同的主成分分量的个数，识别率和虚警率的情况。在此，识别率和虚警率均指整体识别率和整体虚警率，计算方式如下：

$$\begin{aligned} P_{TP} &= \frac{\sum_{i=1}^{c} N_{i,j}}{\sum_{i=1}^{c} N_i} \\ P_{FP} &= \frac{\sum_{i=1,\ldots,c,j=1,\ldots,c,i\neq j} N_{i,j}}{(c-1) * \sum_{j=1,\ldots,c} N_j} \end{aligned} \tag{2.1}$$

2.2.1 在 KNN 算法中取 $k = 1$ 时

绘制出识别率和虚警率随主成分个数的变化曲线如图2.1所示，从图中可以看出，随着主成分数量的增加，最开始识别率呈现递增趋势，在达到最高点之后，识别率有所下降并趋于稳定，这可能就是因为我们的特征数量高于样本数量的原因吧，并且直接使用像素作为特征可能分类效果也不好。从中我们可以看出，当主成分数量为 51 左右时，识别率较高，此时的识别率为 0.2928709055876686，虚警率为 0.03721732075854376。我们绘制出此时的混淆矩阵如图2.2所示。

2.2.2 在 KNN 算法中取 $k = 3$ 时

绘制出识别率和虚警率随主成分个数的变化曲线如图2.3所示，从中我们可以看出，当主成分数量为 31 左右时，识别率较高，此时的识别率为 0.28901734104046245，虚警率为 0.03742013994523882。我们绘制出此时的混淆矩阵如图2.4所示。

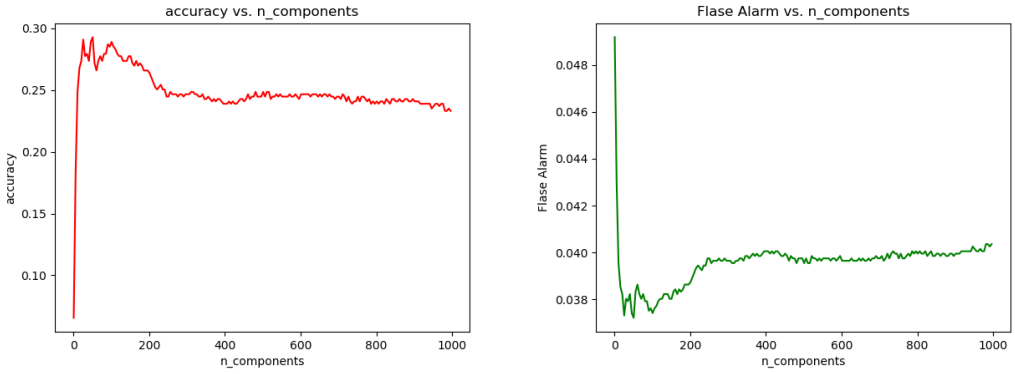

图 2.1: $k = 1$ 时结果图。左图：识别率变化曲线；右图：虚警率变化曲线

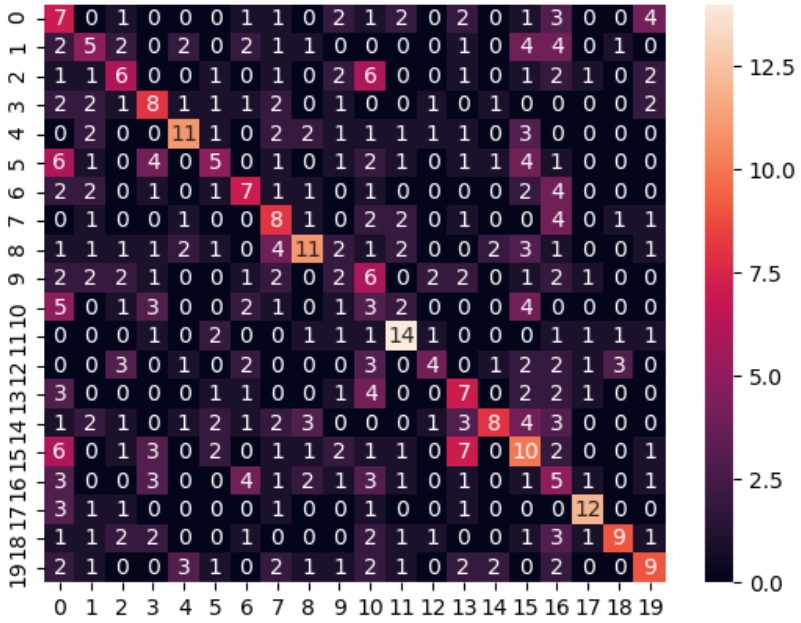

图 2.2: $k = 1, n\_components = 51$ 时混淆矩阵图

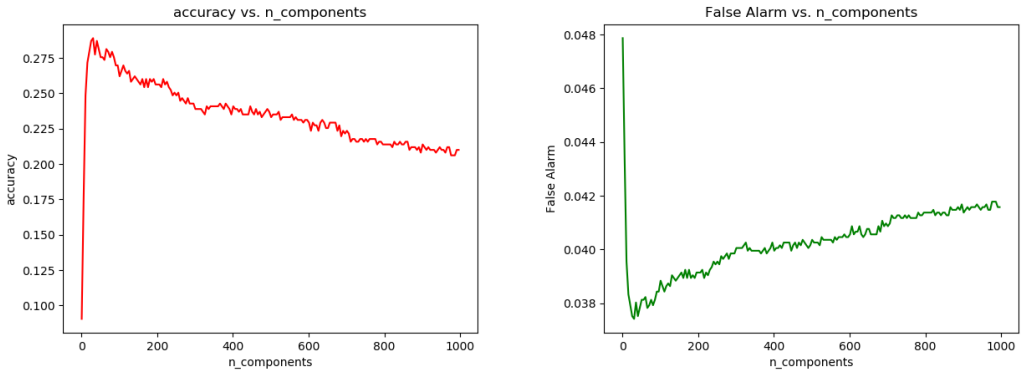

图 2.3: $k = 3$ 时结果图。左图：识别率变化曲线；右图：虚警率变化曲线

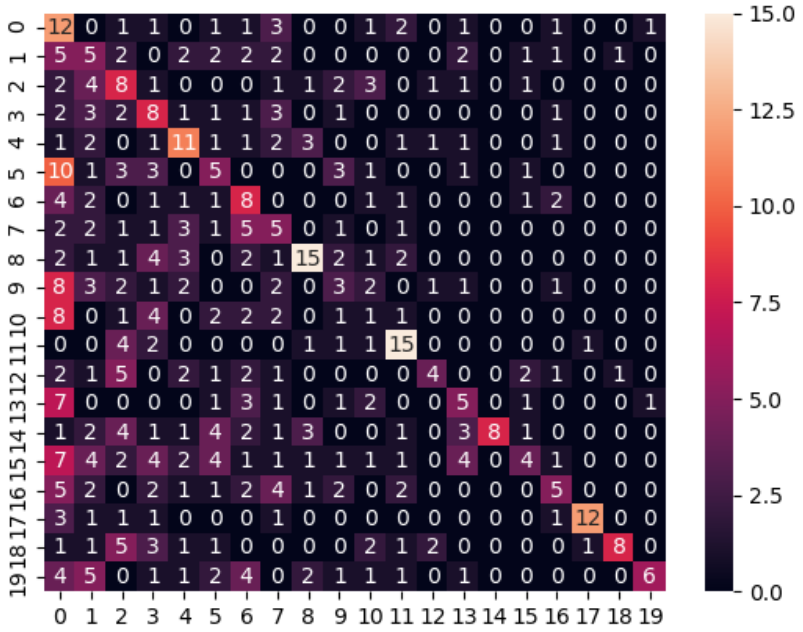

图 2.4: $k = 3, n\_components = 31$ 时混淆矩阵图

2.2.3 在 KNN 算法中取 $k = 5$ 时

绘制出识别率和虚警率随主成分个数的变化曲线如图2.5所示，从中我们可以看出，当主成分数量为 21 左右时，识别率较高，此时的识别率为 0.31021194605009633，虚警率为 0.036304634418415985。我们绘制出此时的混淆矩阵如图2.6所示。

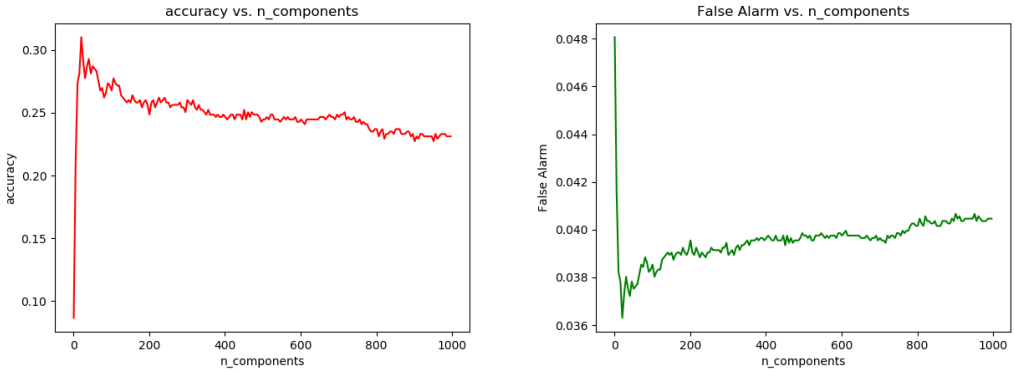

图 2.5: $k = 5$ 时结果图。左图：识别率变化曲线；右图：虚警率变化曲线

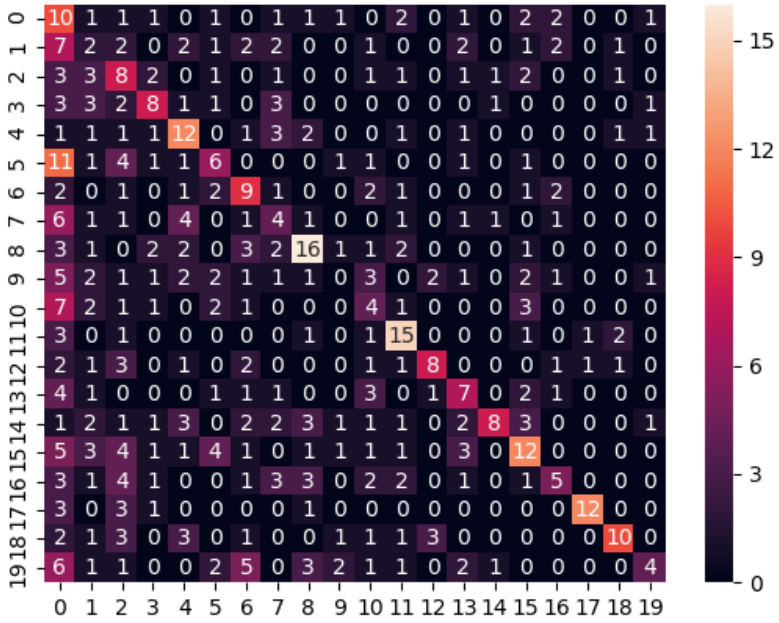

图 2.6: $k = 5, n\_components = 21$ 时混淆矩阵图

2.3 开集测试

在此部分，我们不仅需要分类 20 类物体，还要对背景进行拒识，为此，我们选择 Caltech 256 数据集中第 257 类背景类中不在提供的 test_neg.txt 中的部分图片进行训练，使用的图片保存在 train_neg.txt 中。将其转换为一个 21 分类的问题。由于在之前的部分已经讨论过了实验结果与 KNN 中的 k、PCA 中主成分数量之间的关系，因此在本部分开集测试不再对此进行对比实验，我们选择 $k = 1, n\_components = [1 : 5 : 500]$ 进行实验，绘制出识别率和虚警率随主成分个数的变化曲线如图2.7所示，从中我们可以看出，当主成分数量为 126 左右时，识别率较高，此时的识别率为 0.30737134909596664，虚警率为 0.03674694385476905，我们绘制出此时的混淆矩阵如图2.8所示。

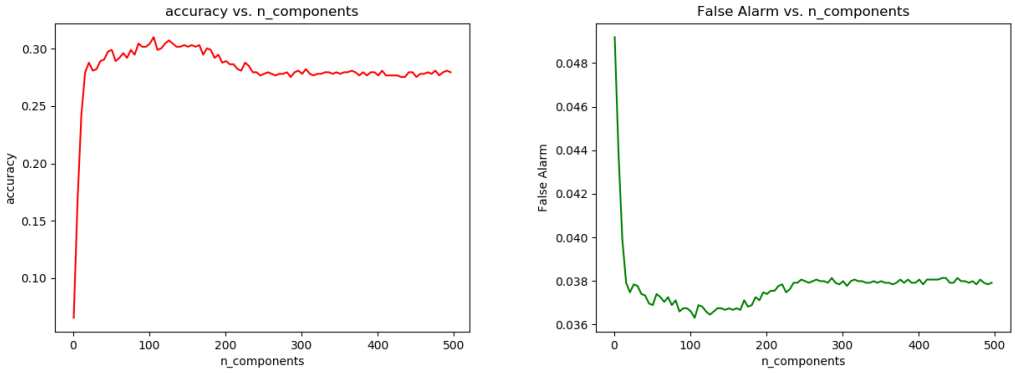

图 2.7: $k = 1$ 时结果图。左图：识别率变化曲线；右图：虚警率变化曲线

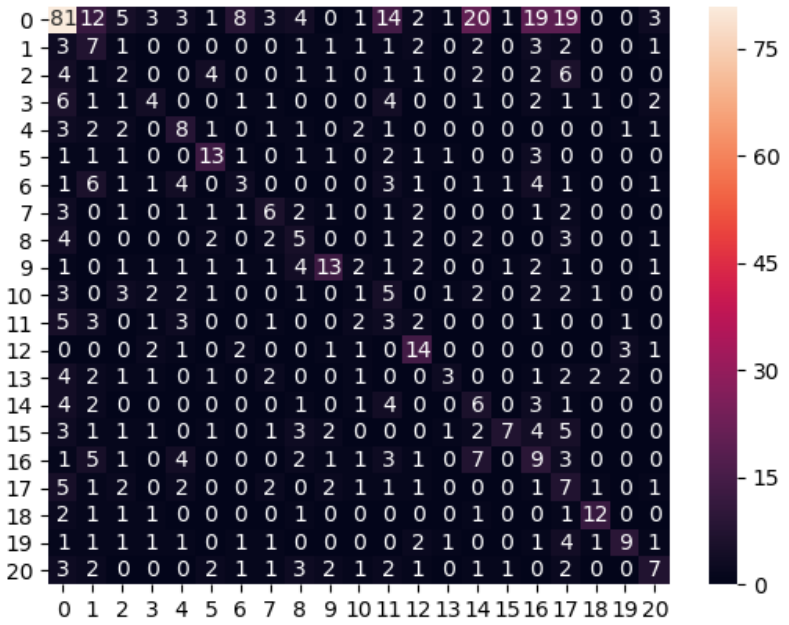

图 2.8: $k = 1, n\_components = 126$ 时混淆矩阵图

第 3 章 HoG 特征 +SVM 分类

3.1 算法描述

在上一章，我们使用像素值作为特征直接进行分类，可以看出，分类的正确率略低，在本章，我们使用 HoG 算法提取图片的 HOG 特征，然后使用 SVM 算法进行多分类。

方向梯度直方图 (Histogram of Oriented Gradient, HOG) 特征是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。它通过计算和统计图像局部区域的梯度方向直方图来构成特征。Hog 特征结合 SVM 分类器已经被广泛应用于图像识别中，尤其在行人检测中获得了极大的成功。因此在此问题上，我们也才用此种分类方法来尝试进行分类。

使用 HoG 算法提取特征之后我们使用 SVM 算法进行分类，SVM 算法主要是针对二分类问题，对于我们的问题主要有两种思路：训练每一类和其他所有类的分类界面 (ovr)，训练每两类之间的分类界面 (ovo)，在本实验中，我们选择了 ovr 的方式进行多分类。

3.2 实验结果

通过多次尝试，在使用线性核 SVM 时 HoG 特征提取算法的最优参数及识别率、虚警率如表3.1所示，对应的混淆矩阵如图3.1所示。从实验结果来看，使用 HoG 特征提取算法能够提取一些较高级的特征，相比于直接用像素值作为特征来说要好很多，并且使用 SVM 分类能够针对线性可分、线性不可分问题有较好的分类界面，因此使用 HoG 特征 +SVM 分类的方式解决此问题要比 PCA+KNN 的方式得到的识别率要高，虚警率更低。但是使用 HoG 特征之后使用 SVM 不能根据自己的需求选择特征，并且 HoG 的特征数量较多，不适合于直接分类，所以可以将这两种算法进行结合，使用 HoG+PCA+SVM 的方式来解决此问题。如图3.2是使用 HoG+PCA+SVM 的方式分类的混淆矩阵示意图，在选择主成分数量为 700 的条件下，最优的识别率为 45.87%，虚警率为 0.02849，可以看出，在使用 PCA 之后识别率有略微提升，并且特征数量相比于直接使用 HoG 有明显减少。

表 3.1: HoG+ 线性 SVM 最优参数及其对应的识别率、虚警率

orientations	block_norm	pixels_per_cell	cells_per_block	识别率	虚警率
13	L1	[8,8]	[4,4]	45.66%	0.028597

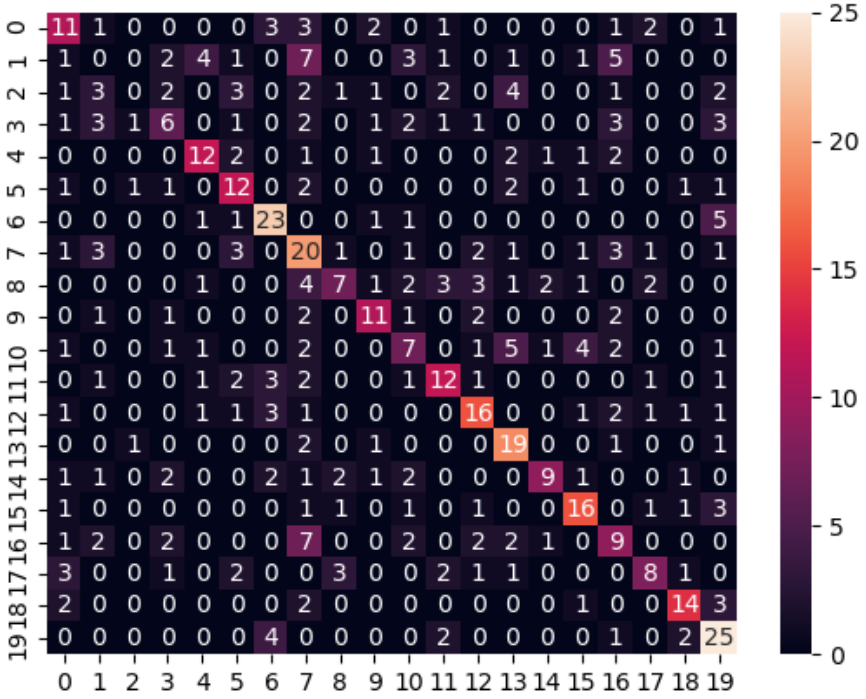

图 3.1: HoG+ 线性 SVM 分类时混淆矩阵图

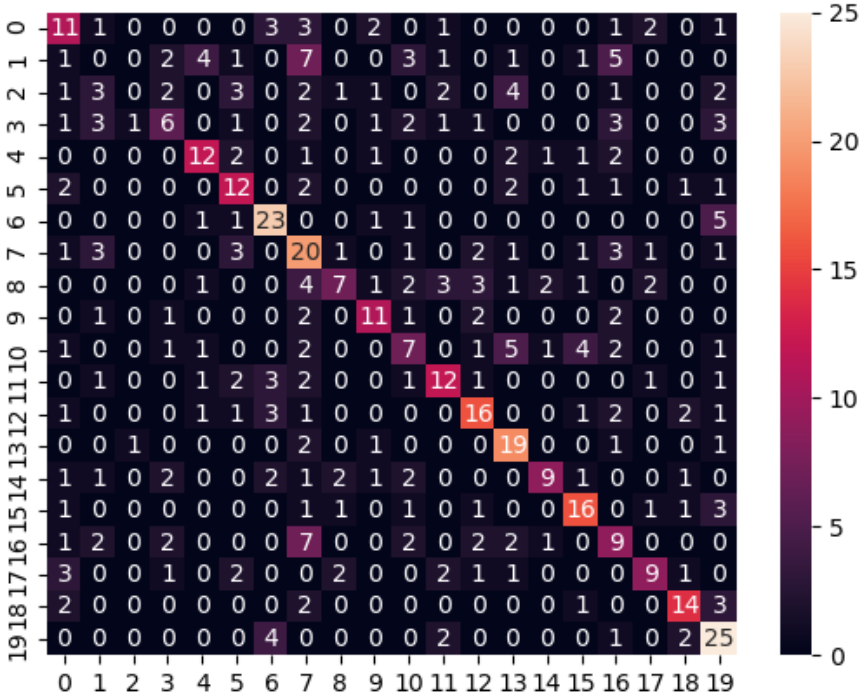

图 3.2: HoG+PCA+ 线性 SVM 分类时混淆矩阵图

3.3 开集测试

与上一章节开集测试类似，我们依然选择 Caltech 256 数据集中第 257 类背景类中不在提供的 test_neg.txt 中的部分图片进行训练，使用的图片保存在 train_neg.txt 中。同样的，我们使用 HoG+SVM 和 HoG+PCA+SVM 两种方式对此进行分类，实验结果如表3.2所示，相对应的混淆矩阵如图3.3和3.4所示，可以看出使用 PCA 降维之后的效果还是相比于直接进行 SVM 的效果好。

表 3.2: 开集测试时的识别率、虚警率

分类方法	识别率	虚警率
HoG+SVM	47.42698%	0.02767
HoG+PCA(n_components=700)+SVM	47.70515%	0.02752

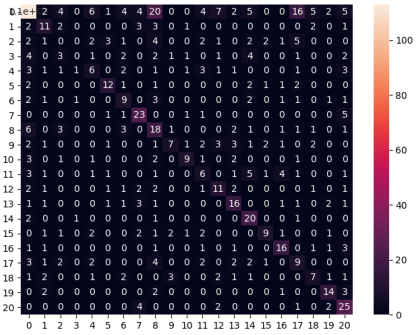

图 3.3: HoG+ 线性 SVM 分类开集测试时混淆矩阵图

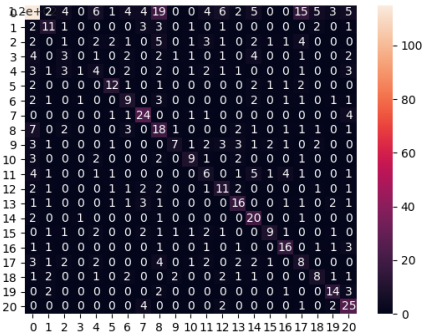

图 3.4: HoG+PCA+ 线性 SVM 分类开集测试时混淆矩阵图

第 4 章 (选做 1) 在全体 257 类图像上使用以上两种方法进行分类

4.1 训练集和测试集创建

首先我们创建训练集和测试集，在此，我们在每一类样本中选择 25 张图片加入测试集，其余图片加入训练集中进行训练，由于使用的算法与之前两章提到的算法类似，在此不再赘述，并且，由于在全体数据集中已经加入了负样本，因此，在此部分也无需进行开集测试。

4.2 PCA+KNN 实验结果

由于在之前章节已经对比了不同 k 和 n_components 对实验结果的影响，因此 (其实还有一方面的原因是电脑太破跑的太慢了 emmm)，本部分只使用 k=5，n_components=500 来进行分类，最终的识别率为 accuracy=8.7345%，虚警率为 0.048034。相比于随机分类的正确率 1/257 要高，但是我们也可以看出，使用此种方法在全体数据集上的分类效果还是很差的。分类后的混淆矩阵图如图4.1所示。

4.3 HOG+SVM 实验结果

使用 HoG+SVM 算法得到的识别率为 16.1045%，虚警率为 0.04415。相比于 PCA+KNN 效果稍微好点，但是识别率依然很低，与我们下一章节即将提到的神经网络分类的方法效果差很多。分类后的混淆矩阵图如图4.2所示。

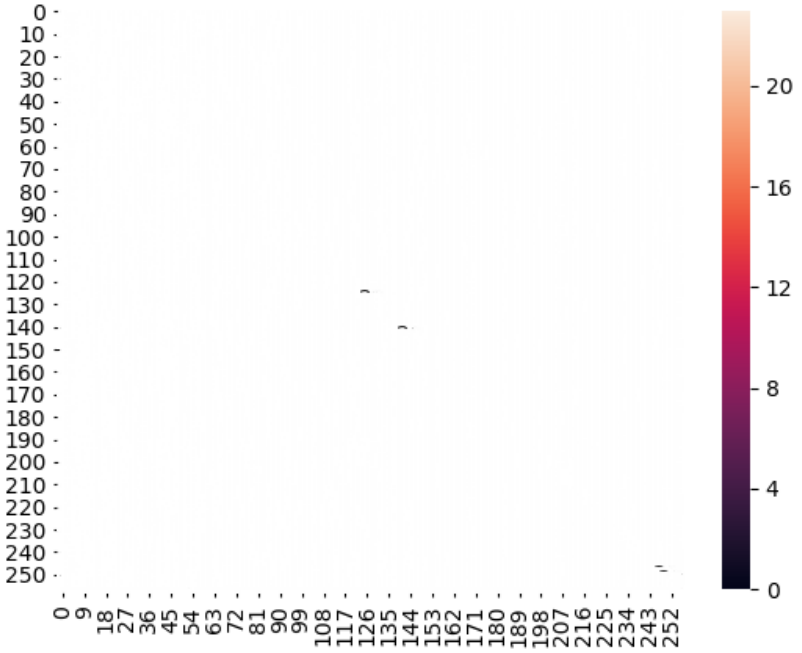

图 **4.1:** $k = 5, n\_components = 500$ 时全体数据集混淆矩阵图

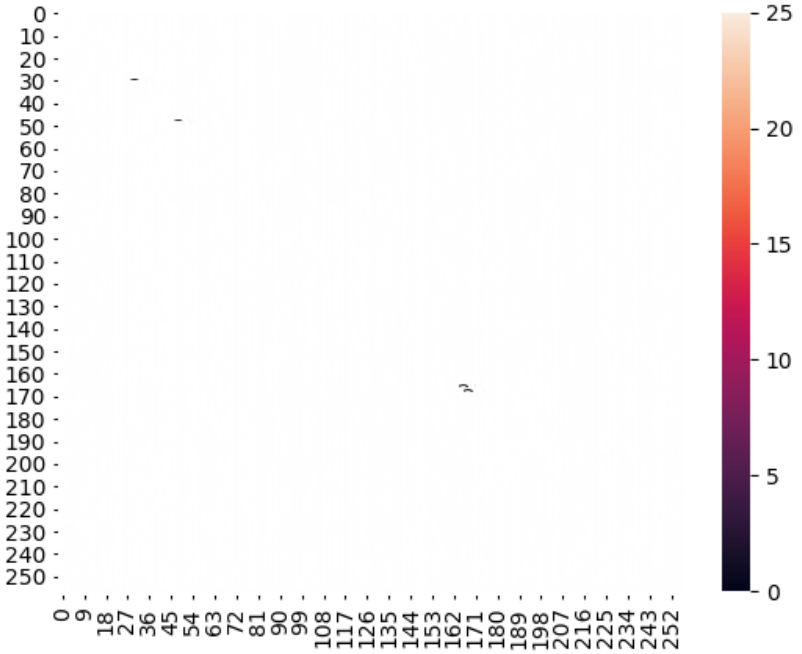

图 **4.2:** HoG+SVM 时全体数据集混淆矩阵图

第 5 章 (选做 2) 在全体 257 类图像上使用 CNN 进行分类

在本章，我们使用深度学习的方法对全体 257 类图像进行分类，首先我们创建训练集和测试集，在此，我们在每一类样本中选择 25 张图片加入测试集，其余图片加入训练集中进行训练。

5.1 算法描述

在本部分，我们使用 ResNet 和 DenseNet 网络结构进行实验，并且将实验结果进行对比。ResNet 是由微软研究院的 Kaiming He 等四名华人提出，并且获得了 ILSVRC2015 的冠军，其核心结构为 ResNet block(图5.1)，传统的卷积网络或者全连接网络在信息传递的时候或多或少会存在信息丢失，损耗等问题，同时还有导致梯度消失或者梯度爆炸，导致很深的网络无法训练。通过引入 ResNet Block，避免了在反向传播中的梯度消失现象，能够设计很深的网络结构。

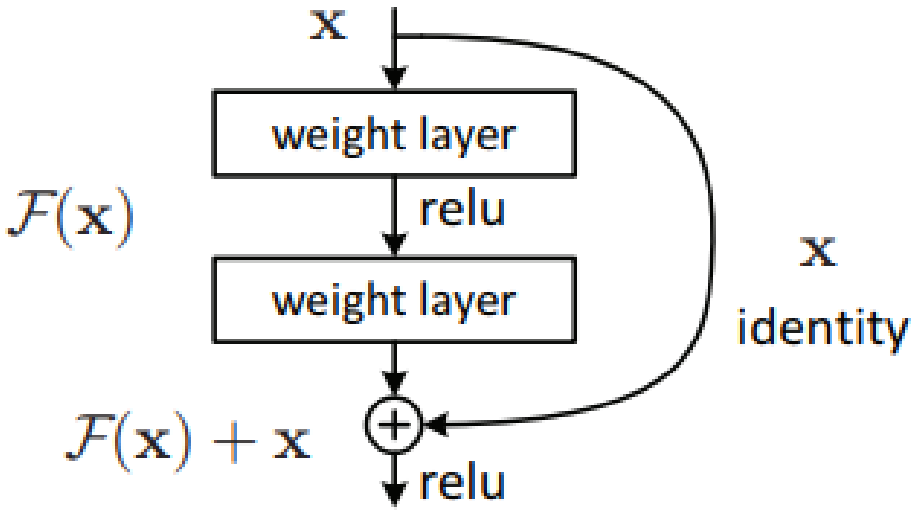

图 5.1: ResNet Block 示意图

DenseNet 是 2017 年 CVPR 的 Best Paper 中提出来的网络结构，其脱离了加深网络层数 (ResNet) 和加宽网络结构 (Inception) 来提升网络性能的定式思维，从特征的角度考虑，通过特征重用和旁路 (Bypass) 设置，既大幅度减少了网络的参数量，又在一定程度上缓解了梯度消失问题的产生，其主要结构为密集连接模块

(Dense Block)，如图5.2所示。

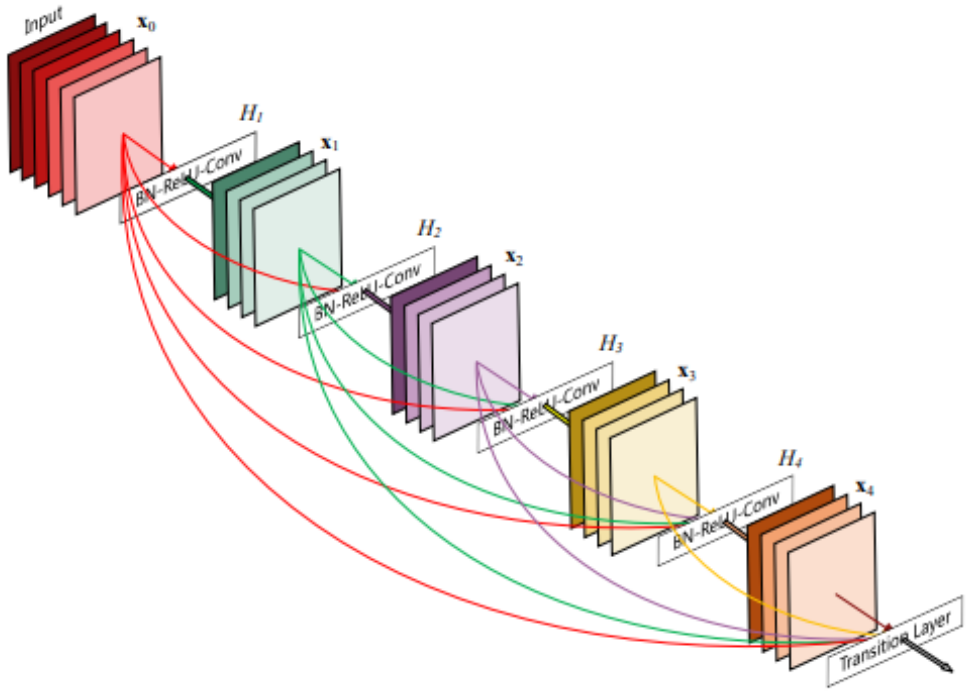

图 5.2: Dense Block 示意图

5.2 实验结果

在本次实验中，我们尝试了多种网络结构与优化方法，使用 ImageNet 上预训练的参数进行训练，学习率采用步长衰减的方式设置，初始学习率为 0.001. 总体实验结果如表5.1所示：

表 5.1: 使用 CNN 进行分类的实验结果

网络结构	优化方法	损失函数	test loss	test accuracy
DenseNet121	SGD(finetune)	cross_entropy	83.48%	0.0427
DenseNet121	Adam(no finetune)	cross_entropy	78.26%	0.0140
DenseNet121	Adam(finetune)	cross_entropy	85.56%	0.0087
ResNet18	Adam(finetune)	cross_entropy	80.36%	0.0119
ResNet18	SGD(finetune)	cross_entropy	71.45%	0.0237

并且针对不同情况，我们都绘制出了 train loss, train accuracy, test loss 和 test accuracy 曲线进行对比。DenseNet121+SGD 的曲线如图5.3和5.4所示，在此次实验中，我们固定了除最后一层外的所有参数，均使用预训练的参数，而只通过训练微调最后一层的参数，因此可以看出收敛较快；

相比较而言，我们的第二次尝试，DenseNet121+Adam，我们虽然依然使用了预训练的参数，但是并没有固定除最后一层之外的其它层的参数，因此所有层

的参数都是可以微调的，可以看出，此时的收敛速度并没有上一种情况的快，如图5.5和5.6所示，从实验结果中可以看出，训练集的正确率明显高于测试集的正确率，这有可能是因为我们微调所有的参数使得可变化的参数量很多，有过拟合现象，因此在此问题中，我们固定其中一些参数可能能取得较好的效果。固定参数使用 DenseNet121+Adam 的实验结果如图5.7和5.8所示，可以看出使用 finetune 方式的正确率更优，且收敛速度更快。

我们还使用了 ResNet18 作为网络结构对此问题进行分类，根据我们上面的讨论，在此我们还是使用 finetune 的方式进行训练，即固定除最后一层之外的其它层的参数，使用 Adam 优化算法的实验结果如图5.9和5.10所示，使用 SGD 优化算法的实验结果如图5.11和5.12所示，从实验结果中可以看出使用 Adam 优化算法进行实验比使用 SGD 优化算法的正确率和 loss 都要更优，并且我们还可以看出，使用 Adam 优化算法收敛速度也要比 SGD 快，这与我们之前在 DenseNet121 模型上对比的结果是一致的。

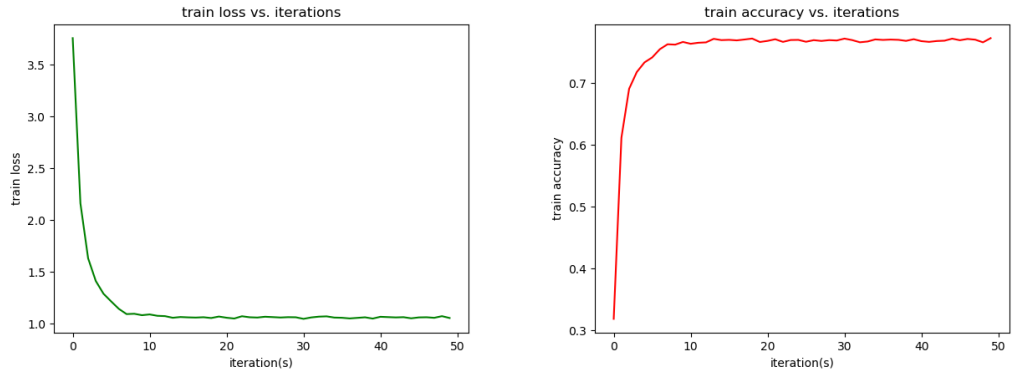

图 5.3: train loss curve and train accuracy curve using DenseNet121+SGD

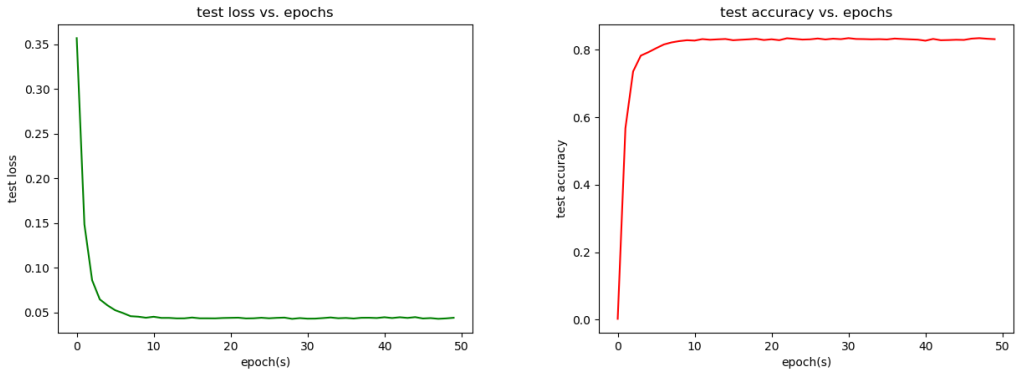

图 5.4: test loss curve and test accuracy curve using DenseNet121+SGD

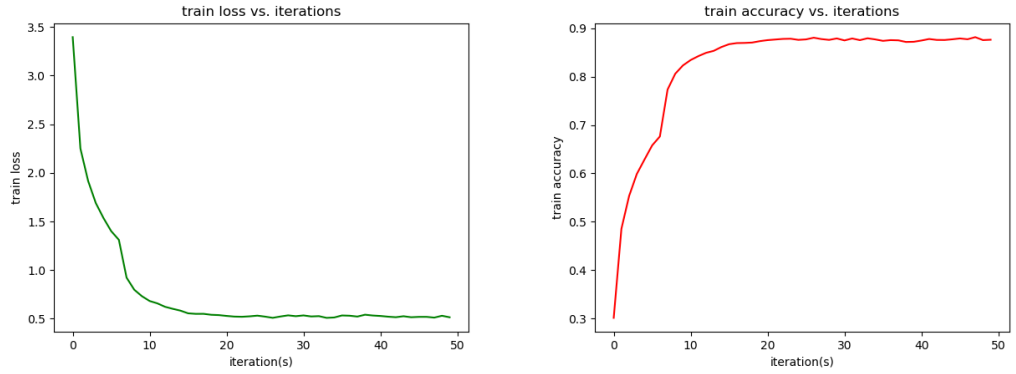

图 5.5: train loss curve and train accuracy curve using DenseNet121+Adam (without finetuning)

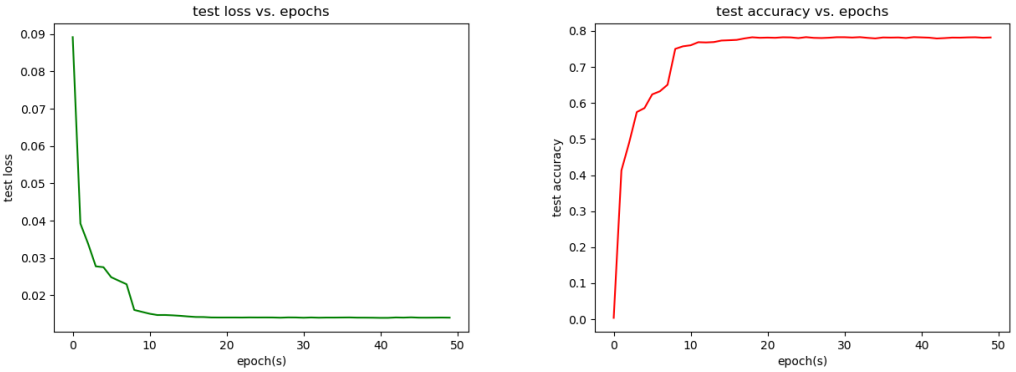

图 5.6: test loss curve and test accuracy curve using DenseNet121+Adam (without finetuning)

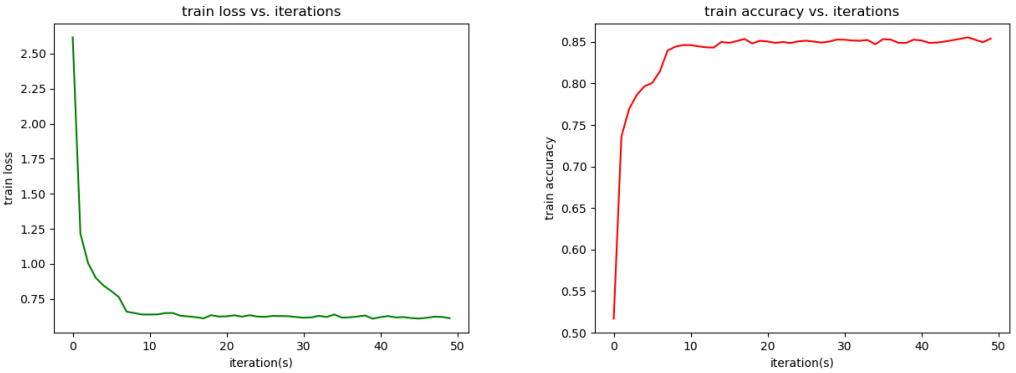

图 5.7: train loss curve and train accuracy curve using DenseNet121+Adam (with finetuning)

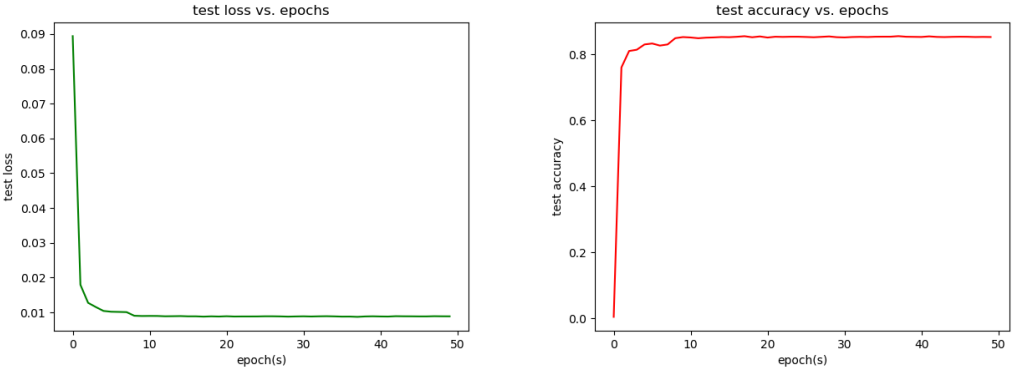

图 5.8: test loss curve and test accuracy curve using DenseNet121+Adam (with finetuning)

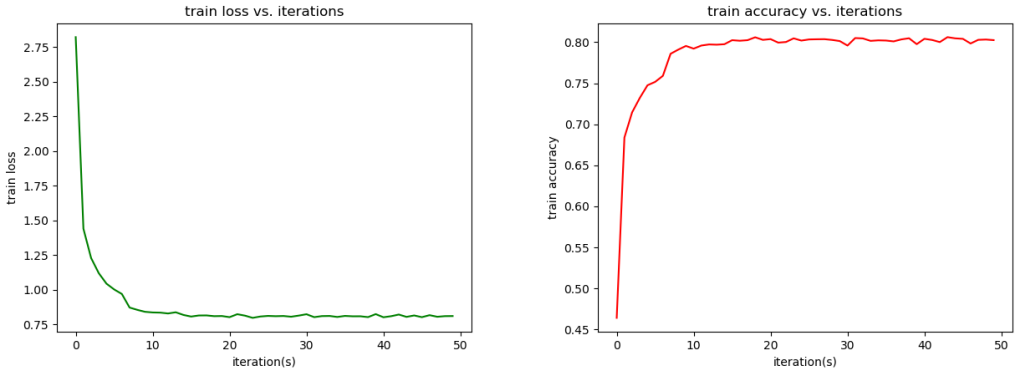

图 5.9: train loss curve and train accuracy curve using ResNet18+Adam

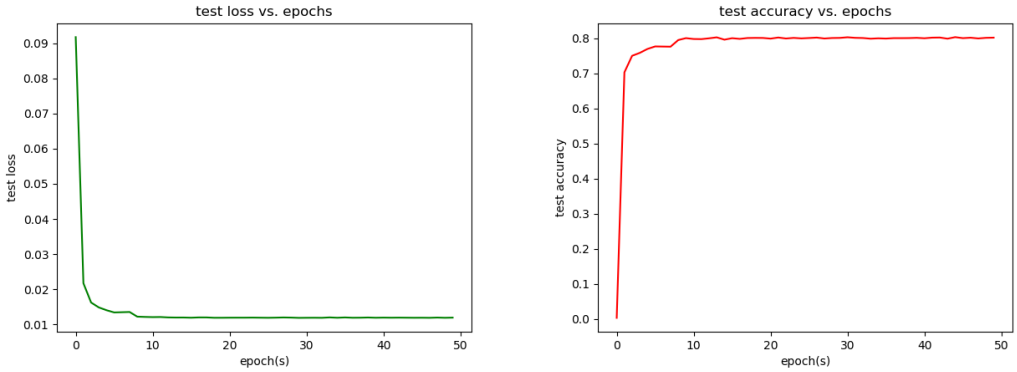

图 5.10: test loss curve and test accuracy curve using ResNet18+Adam

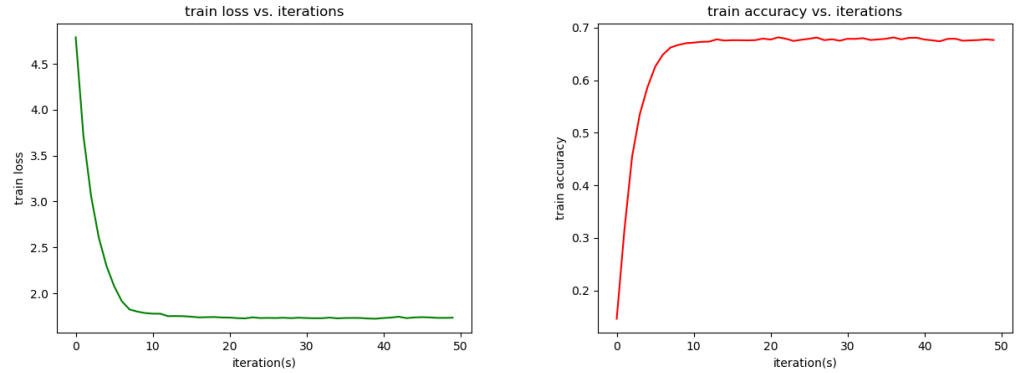

图 5.11: train loss curve and train accuracy curve using ResNet18+SGD

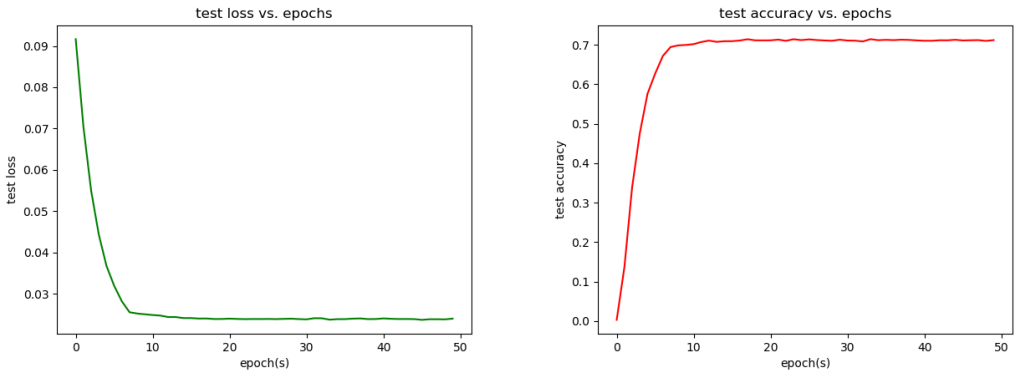

图 5.12: test loss curve and test accuracy curve using ResNet18+SGD

第 6 章 总结与代码说明

6.1 总结

在本次实验中，我们采用了 PCA 算法降维、HOG 算法提取图像特征、KNN 分类、SVM 分类、神经网络 CNN 分类等方式对 Caltech 256 数据集上的数据进行分类，从分类结果来看，使用神经网络进行分类比使用传统机器学习的方法进行分类的正确率高出很多，这也是为什么最近深度学习很火的原因之一吧，通过神经网络可以提取到一些传统方法提取不到的特征，这样更适合分类，对于神经网络来说，网络结构的定义、损失函数的定义以及参数更新方法都会对最后结果带来一定的影响，本次大作业中使用了 ResNet18 和 DenseNet121 等网络结构、交叉熵损失函数、随机梯度下降法和 Adam 等参数更新方法对此数据集进行分类，从结果对比之中发现 DenseNet121 网络结构以及 SGD 参数更新方法在此问题的表现上较优，但是针对不同的问题不同的网络结构会有不同的优势，这需要我们在针对不同问题时多进行尝试找到最优结构，并且我们还可以对数据集进行分析，提出一些创新型的网络结构，或许会有更好的效果。相比较而言，传统方法虽然正确率较低，但是有完整的理论支持，不像神经网络一样是黑盒模型，并且传统模型训练时间也比神经网络方法少很多。

总体来说，本次大作业不仅复习了模式识别课上讲过的一些内容，而且亲自动手实现了相应的算法，并且使用 PyTorch 对深度学习的算法也进行了一些尝试，学到了很多东西。

6.2 代码说明

本次大作业代码全部使用 python3.7 完成，文件说明如下：

- code_base 文件夹：此文件夹下放置的为基本问题的代码：
  - pca_classify.py：使用 PCA+KNN 分类的代码；
  - hog_svm.py：使用 HoG 提取特征，SVM 分类的代码，其中包含使用 PCA 降维特征的代码；
  - create_train_neg.py：创建背景类的训练集，此训练集中的图片与所给的 test_neg.txt 中均不同；
  - pca_classify_open.py：使用 PCA+KNN 开集测试分类的代码；
  - hog_svm.py：使用 HoG 提取特征，SVM 分类的开集测试代码，其中

包含使用 PCA 降维特征的代码；

- code_additional 文件夹：此文件夹下放置的为选做问题的代码：
  - create_train_test.py：创建选做题部分的训练集和测试集 txt 文件，并且将文件保存在 data_additional 文件夹下
  - my_dataset.py：创建适用于 pytorch 读取图片的 Dataset 类；
  - run_cnn.py：使用 CNN 进行训练和测试的代码；
  - pca_knn.py：使用 PCA+KNN 在全体数据集上分类的代码；
  - hog_svm.py：使用 HoG+SVM 在全体数据集上分类的代码；
- data 文件夹：此文件夹下放置的为基础问题所用到的训练集和测试集的 txt 文件
- data_additional 文件夹：此文件夹下放置的为选做问题训练集和测试集的 txt 文件

第 7 章 致谢

- 感谢老师一学期以来的辛勤讲述，在课堂上对模式识别的相关算法有了初步的了解，并且从老师分享的自己实验组课题项目中可以学到一些模式识别实战性的经验；
- 感谢助教一学期的认真批改作业，辛苦助教了；