机器学习课程设计报告

指导教师：冯杰、张国萍

班级：18 智科（2）班

学号：2018329621262

姓名：刘恒玮

机器学习课程设计任务书

一、课程设计目的：

1、通过本课程的学习，提高学生综合运用软件开发工具对课堂所授理论进行软件编程的能力；

2、较强的理论联系实际能力和自主学习能力，以提高学生学习机器学习理论和技术的积极性和学习兴趣；

3、主动探索和独立思考的能力，以激发学生的应用创新意识。

二、设计内容：

进入 ilab 竞赛平台（ilab.cmcm.com/index），对猫狗图片进行训练和识别，要求：综合使用各种机器学习算法，总结调试经验。将结果提交至竞赛平台进行自动评判。

三、要求

3.1 总体要求

1、要充分认识课程设计对培养自己的重要性，认真做好课程设计前的各项准备工作。尤其是对竞赛平台和 python 工具的使用有基本的认识。

2、既要虚心接受老师的指导，又要充分发挥主观能动性。结合课题，独立思考，努力钻研，勤于实践，勇于创新。

3、独立按时完成规定的工作任务，不得弄虚作假，不准抄袭他人内容，否则成绩以不及格计。

4、在设计过程中，要严格要求自己，树立严肃、严密、严谨的科学态度，必须按时、按质、按量完成课程设计。

3.2 实施要求

1、理解各种机器学习方法确切意义。

2、独立进行方案的制定，系统结构设计要合理。

3、在程序开发时，则必须清楚主要实现函数的目的和作用，需要在程序书写时说明做适当的注释。要理解每个函数的具体意义和适用范围，在写课设报告时，必须要将主要函数的功能和参数做详细的说明。

4、通过多幅不同形式的图像数据来检测该系统的稳定性和正确性。

3.3 课程设计报告的内容及要求

在完成课程设计验收后，学生应在规定的时间内完成课程设计报告一份（不少于 2000 字）。

四、工作内容及工作计划：（一周，分散进行）

序号	内　　容	所用时间
1	课题介绍、人员安排、熟悉编程环境	1 天
2	分析题目，编写程序	3 天
3	上机调试程序并写出设计报告	1 天
合计		1 周

五、成绩评定标准与考核：

本课程设计的评价由自动评判成绩、课程设计报告两部分组成。其中自动评判成绩（50%），课程设计报告（50%）。

1、 自动评判成绩：0～100，教师在查看学生代码后确认成绩并转换为五级制。

2、 课程设计报告：

（1） 优 包括设计内容，设计思想，已经完成的任务及达到的目标，设计思路清晰、书写条理清楚，源程序结构合理、清晰，注释说明完整，有对本次课程设计的心得体会。

（2） 良 包括设计内容，设计思想，已经完成的任务及达到的目标，设计思路基本清晰、书写条理基本清楚，源程序结构合理、清晰，注释说明基本完整，有对本次课程设计的心得体会。

（3） 中 课程设计报告内容基本完整，思路较清晰，书写基本清楚，源程序结构尚可，有注释说明但不完整

（4） 及格 课程设计报告内容基本完整，思路较差，书写尚清楚。

（5） 不及格 课程设计报告内容不完整，书写没有条理。

一、课程设计目的

1、通过本课程的学习，提高学生综合运用软件开发工具对课堂所授理论进行软件编程的能力；

2、较强的理论联系实际能力和自主学习能力，以提高学生学习机器学习理论和技术的积极性和学习兴趣；

3、主动探索和独立思考的能力，以激发学生的应用创新意识。

二、总体设计

利用 Keras 框架下的 resnet50、resnet152 等现有训练好的模型基础上，针对猫狗分类这个具体的二分类问题进行学习训练，进行特征提取，可以大大加快训练过程，提高模型精度，同时简化学习过程。

三、详细设计（对问题的分析，解决方案设计）

（1） 首先将训练集分为训练集和验证集，分别保存在 train 和 val 文件夹中，并对训练集中的猫狗图片进行数据增强，用 Keras 里的 ImageDataGenerator 从训练集和测试集中批量地引入图像流，其次创建两个生成器 train_generator 和 validation_generator，这样每个批次引入随机修改后的图片，可以生成更多的图片使得我们的模型不会看见两张完全相同的图片，这种方法可以防止过拟合，也有助于模型保持更好的泛化性。

（2） 导入 Keras 框架下的 resnet50 预处理模型，并在最后添加一个 Flatten 层用来将输入“压平”，即把多维的输入一维化，用在从 resnet50 的卷积层到全连接层的过渡，最后的全连接层由于是二分类问题，激发函数选择为 sigmoid。

（3） 定义学习率之后，经过一定 epoch 迭代之后，模型效果不再提升，该学习率可能已经不再适应该模型。若初始的学习率过小，会需要非常多次的迭代才能使模型达到最优状态，训练缓慢。如果可以在训练过程中不断缩小学习率，可以快速又精确的获得最优模型。利用 Keras 中的回调函数 ReduceLROnPlateau 在训练过程中缩小学习率，进而提升模型，同时用回调函数 EarlyStopping 判断模型是否再继续优化从而决定是否要继续迭代训练模型。

四、结果与分析（模型选择，调参）

四层 CNN：

利用 Keras 框架搭建了四层的 CNN 神经网络模型， 在经过 3 个 全连接层（Dense 层）之前，将第 3 个 Dropout 层的（4，4，128）输出形状为（2048，）的向量，在训练和验证数据上训练了 batch size = 256 ， epoch=10 的模型，验证集的准确率达到了 88%左右，具体的网络结构如下图所示：

```
cnn4 = Sequential()
cnn4.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_sh
cnn4.add(BatchNormalization())

cnn4.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
cnn4.add(BatchNormalization())
cnn4.add(MaxPooling2D(pool_size=(2, 2)))
cnn4.add(Dropout(0.25))

cnn4.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
cnn4.add(BatchNormalization())
cnn4.add(Dropout(0.25))

cnn4.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
cnn4.add(BatchNormalization())
cnn4.add(MaxPooling2D(pool_size=(2, 2)))
cnn4.add(Dropout(0.25))

cnn4.add(Flatten())

cnn4.add(Dense(512, activation='relu'))
cnn4.add(BatchNormalization())
cnn4.add(Dropout(0.5))

cnn4.add(Dense(128, activation='relu'))
cnn4.add(BatchNormalization())
cnn4.add(Dropout(0.5))

cnn4.add(Dense(1, activation='sigmoid'))
```

Resnet50、Resnet152：

（1）进行数据增强时，对图片进行归一化处理，并加入动态调整学习率，在 ilab 平台上的分数从 93.1 提升到 95.3。具体的图片增强的参数如下：

```
rotation_range=45,#随机旋转
width_shift_range=0.2,#是图像在水平上平移的范围
height_shift_range=0.2,#垂直方向上平移的范围
shear_range=0.2,#随机错切变换的角度
```

zoom_range=0.2,#随机缩放的范围

horizontal_flip=True,#随机将图像水平翻转

preprocessing_function=preprocess_input#归一化

将 batch_size 的大小设置为 64，训练集图片的大小全部统一到 128*128，并将学习率 lr 设置为 0.00001。

动态调整学习率的具体参数如下图所示：

```
[3]: from keras.callbacks import ReduceLROnPlateau,EarlyStopping
Reduce=ReduceLROnPlateau(
    monitor ='acc',#监测的值，可以是accuracy，val_loss,val_accuracy
    factor=0.1,#缩放学习率的值，学习率将以lr = lr*factor的形式被减少
    patience=2,#当patience个epoch过去而模型性能不提升时，学习率减少的动作会被触发
    verbose=1,
    mode='auto',#‘auto’，‘min’，‘max’之一 默认‘auto’就行
    cooldown=0,#学习率减少后，会经过cooldown个epoch才重新进行正常操作
    min_lr=0 #学习率最小值，能缩小到的下限
)
early_stop = EarlyStopping(
    monitor='val_loss',
    patience=10
)
```

（2）将图片大小从原先的 128*128 改为 216*216，在平台上训练了两轮，最后在 ilab 平台上的准确率从 95.3 提高到了 97.9，具体的网络结构如下图所示：

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 7, 7, 2048)        23587712
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
dense (Dense)                (None, 1)                 100353
=================================================================
```

（3）最后将 resnet50 预处理模型改为 resnet152，同时在最后的全连接层前面加上了 Dropout 层防止过拟合，并将图片大小改为 300*300，训练了 20 轮迭代之后，准确率从 97.9 提高到了 99.0。

（4）用 GlobalAveragePooling2D 替换 Flatten、增加了两个密集连接层，同时添加 BN、Activation、Dropout 层，具体的网络结构如下图所示，ilab 平台上的分数从 99.0 提高到了 99.15。

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet152 (Functional)       (None, 10, 10, 2048)      58370944
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
batch_normalization (BatchNo (None, 1024)              4096
_________________________________________________________________
activation (Activation)      (None, 1024)              0
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               262400
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
activation_1 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
_________________________________________________________________
```

五、课程设计心得

本次课程设计的猫狗分类问题属于计算机视觉领域中的图像分类问题。图像分类的过程非常明确：给定已经标记的数据集，提取特征，训练得到分类器。对给定的猫和狗的图片进行分类，因此是二分类问题，其中具体的任务就是使用4000 张猫和 4005 张狗的图片作为训练集，训练一个合适的模型，能够在给定的2023 张未见过的图像中辨别猫和狗。

深度学习中有一种重要的学习方法是迁移学习，可以在现有训练好的模型基础上针对具体的问题进行学习训练，使用 resnet50、resnet152 模型进行迁移学习训练猫狗分类模型，使得相比于简单的多层神经网络训练得到的模型效果会好很多，但是在训练模型之前，一定要“冻结”预处理模型 resnet50 或者 resnet152 的卷积基。冻结一个或多个层是指在训练过程中保持其权重不变。如果不这么做，那么卷积基之前学到的表示将会在训练过程中被修改。因为其上添加的 Dense 层是随机初始化的，所以非常大的权重更新将会在网络中传播，对之前学到的内容造成很大破坏。

总的来说，本次课程设计在一定程度上巩固了对机器学习理论课上所学的各种神经网络的结构的理解，同时对设计过程中所使用的 resnet50 和 resnet152 有了更深入的了解。比如说 resnet50 有四组大 block，每组分别是 3，4，6，3 个小 block，每个小 block 里面有三个卷积层，另外这个网络的最开始有一个单

独的卷积层，因此是：（3+4+6+3）*3+1=49，最后又一个全连接层，因而一共50层，所以命名为resnet50。Resnet152则是在resnet50的基础上加厚了第三个和第四个block，从原来的4个小block和6个小block变成了8个小block和36个小block。本次课程设计中所学到的各种方法和各种神经网络模型，相对来说简单易用，对于我们所要识别和分类的数据集，不管网络内部的参数情况如何，最终能够得到比较理想的识别和分类效果，甚至可以说是非常理想的分类效果，准确率可以达到99%以上，但是各种方法最大的缺点就是都需要消耗较多的时间去得到一个理想的效果（一个较高的分数），但对于内部的结构和各种参数，还需要更多时间去继续深入学习和更好的理解。不过，通过本次的课程设计提高了自己对机器学习这方面相关知识的学习兴趣，并培养了自己主动探索和独立思考的能力。

六、主要代码

```
import os
from keras import layers, optimizers, models
from keras.applications.resnet import ResNet152, preprocess_input
from keras.layers import *
from keras.models import Model
train_dir = os.path.join(r'D:\jupyterlab\training')
validation_dir = os.path.join(r'D:\jupyterlab\val')
from keras.callbacks import ReduceLROnPlateau,EarlyStopping
Reduce=ReduceLROnPlateau(
    monitor ='val_loss',#监测的值，可以是accuracy，
val_loss,val_accuracy
    factor=0.1,#缩放学习率的值，学习率将以lr = lr*factor的形式被减少
    patience=2,#当patience个epoch过去而模型性能不提升时，学习率减少的动
作会被触发
    verbose=1,
    mode='auto',#‘auto’，‘min’，‘max’之一 默认‘auto’就行
    cooldown=0,#学习率减少后，会经过cooldown个epoch才重新进行正常操作
    min_lr=0 #学习率最小值，能缩小到的下限
)
early_stop = EarlyStopping(
    monitor='val_loss',
    patience=10
)
```

```
resnet152 = ResNet152(weights='imagenet', include_top=False, input_s
hape=(300,300, 3))

model = models.Sequential()
model.add(resnet152)
model.add(GlobalAveragePooling2D())
model.add(Dense(1024))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.2))
# model.add(layers.Flatten())
# model.add(Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))

resnet152.trainable = False
#冻结一个层意味着将其排除在训练之外，即其权重将永远不会更新
optimizer = optimizers.RMSprop(lr=1e-4)

def get_lr_metric(optimizer):
    def lr(y_true, y_pred):
        return optimizer.lr
    return lr
lr_metric = get_lr_metric(optimizer)
model.compile(loss='binary_crossentropy', optimizer = optimizers.RMS
prop(lr=1e-4), metrics=['acc',lr_metric])

from keras.preprocessing.image import ImageDataGenerator

batch_size = 64

train_datagen = ImageDataGenerator(
    rotation_range=45,#随机旋转
    width_shift_range=0.2,#是图像在水平上平移的范围
    height_shift_range=0.2,#垂直方向上平移的范围
    shear_range=0.2,#随机错切变换的角度
    zoom_range=0.2,#随机缩放的范围
    horizontal_flip=True,#随机将图像水平翻转
    preprocessing_function=preprocess_input#归一化
)
```

```
val_datagen = ImageDataGenerator(preprocessing_function=preprocess_i
nput)


train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(300,300),
        batch_size=batch_size,
        class_mode='binary')
validation_generator = val_datagen.flow_from_directory(
        validation_dir,
        target_size=(300,300),
        batch_size=batch_size,
        class_mode='binary')

model.save('cat-dog-resnet152-epoch30.h5')
```

成绩评定表

自动评判成绩	
报告成绩	
最终成绩	