暨南大学本科实验报告专用纸

课程名称 人工智能原理 成绩评定

实验项目名称 深度学习 指导教师 张佳

实验项目编号 实验项目类型 综合 实验地点 线上

学生姓名 陈宇 学号 2020101642

学院 信息科学技术学院 系 计算机系 专业 软件工程

实验时间 2022 年 11 月 25 日下午～11 月 25 日 下午 温度 ℃湿度

实验内容

本次实验学习和体会深度学习的神经网络算法，利用 python 的 tensorflow 实现对猫狗图片的识别。在实现对任务一的完成后，通过增加训练集，验证集的方法，观察模型训练的结果。完成任务 2 后，对神经网络模型的结构进行更改，观察在变化网络层数，卷积核数，卷积核大小后模型的变化。

核心代码

训练模型：train.py

```
import os

import tensorflow as tf

from keras.optimizers import RMSprop

from keras.preprocessing.image import

ImageDataGenerator

import matplotlib.pyplot as plt

# 数据集地址，如果出现地址错误，则考虑使用绝对地址

base_dir = 'dataset/cats_and_dogs'

# 指定每一种数据的位置

train_dir = os.path.join(base_dir, 'train')
```

```
validation_dir = os.path.join(base_dir, 'validation')
# Directory with our training cat/dog pictures
train_cats_dir = os.path.join(train_dir, 'cats')
train_dogs_dir = os.path.join(train_dir, 'dogs')
# Directory with our validation cat/dog pictures
validation_cats_dir = os.path.join(validation_dir,
'cats')
validation_dogs_dir = os.path.join(validation_dir,
'dogs')
# 模型处添加了 dropout 随机失效，也就是说有时候可能不用到某
些神经元，失效率为 0.5
# 下面是模型的整体结构，可以观察到每一层卷积之后，都会使用一
个最大池化层对提的数据进行降维，减少计算量，后续实验修改网络
结构主要修改下面部分
# 设计模型
model = tf.keras.models.Sequential([
 # 我们的数据是 150x150 而且是三通道的，所以我们的输入应该设
置为这样的格式。
 tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(150, 150, 3)),
 tf.keras.layers.MaxPooling2D(2, 2), # 最大池化层
 tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
```

```
 tf.keras.layers.MaxPooling2D(2, 2),
 tf.keras.layers.Conv2D(128, (3, 3),
activation='relu'),
 tf.keras.layers.MaxPooling2D(2, 2),
 tf.keras.layers.Dropout(0.5), # dropout 层通通过忽略一般
数量特征，可以减少过拟合现象
 tf.keras.layers.Flatten(), # 全链接层，将多维输入一维化
 tf.keras.layers.Dense(512, activation='relu'),
 # 二分类只需要一个输出
 tf.keras.layers.Dense(1, activation='sigmoid')
])
# 进行优化方法选择和一些超参数设置
# 因为只有两个分类。所以用 2 分类的交叉熵，使用 RMSprop，学
习率为 0.0001.优化指标为 accuracy
model.compile(loss='binary_crossentropy',  # 损失函数使用
交叉熵
  optimizer=RMSprop(lr=1e-4),  # 优化器，学习率设置为
0.0001
  metrics=['acc'])
# 数据处理
# 把每个数据都放缩到 0 到 1 范围内
```

```
# 这里的代码进行了更新，原来这里只进行归一化处理，现在要进行数据增强。
train_datagen = ImageDataGenerator(
 rescale=1. / 255,
 rotation_range=40,
 width_shift_range=0.2,
 height_shift_range=0.2,
 shear_range=0.2,
 zoom_range=0.2,
 horizontal_flip=True,
 fill_mode='nearest')
test_datagen = ImageDataGenerator(rescale=1. / 255)
# 生成训练集的带标签的数据
train_generator =
train_datagen.flow_from_directory(train_dir, # 训练图片的位置
 batch_size=20, # 每一个投入多少张图片训练
 class_mode='binary', # 设置我们需要的标签类型
 target_size=(150, 150)) # 将图片统一大小
# 生成验证集带标签的数据
```

```
validation_generator =
test_datagen.flow_from_directory(validation_dir, # 验证
图片的位置
 batch_size=20, # 每一个投入多少张图片训练
 class_mode='binary', # 设置我们需要的标签类型
 target_size=(150, 150)) # 将图片统一大小
# 进行训练
history = model.fit_generator(
 train_generator, # 训练集数据
 steps_per_epoch=20, # 每个 epoch 训练多少次
 epochs=10, # 训练轮数，建议在[10,50]如果电脑训练速度快，可
以大于 50
 validation_data=validation_generator, # 验证集数据
 validation_steps=10,
 verbose=1) # 训练进度显示方式，可取值 0，1（显示训练进度
条），2（一个 epoch 输出一条信息）
# 保存训练的模型到当前目录
model.save('model.h5')
# 得到精度和损失值
acc = history.history['acc'] # train_acc
val_acc = history.history['val_acc'] # val_acc
loss = history.history['loss'] # train_loss
```

```
val_loss = history.history['val_loss'] # val_loss

epochs = range(len(acc)) # 得到迭代次数

# 绘制精度曲线

plt.plot(epochs, acc)

plt.plot(epochs, val_acc)

plt.title('Training and validation accuracy')

plt.legend(('Training accuracy', 'validation

accuracy'))

plt.figure()

# 绘制损失曲线

plt.plot(epochs, loss)

plt.plot(epochs, val_loss)

plt.legend(('Training loss', 'validation loss'))

plt.title('Training and validation loss')

plt.show()
```

模型预测：predict.py

```
# 预测

from tensorflow.python.keras.models import load_model

import numpy as np

from keras.utils import image_utils

# 测试图片地址

path = 'dataset/predicted/dog1.jpeg'
```

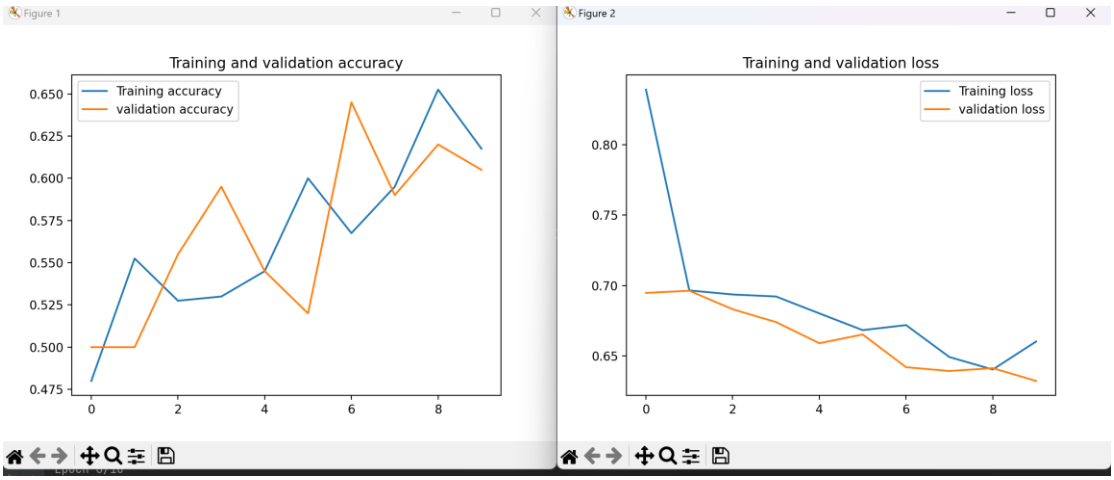
```
# 加载模型
model = load_model('model.h5')
# 将图片转换成 150*150 的格式，与模型训练的输入保持一致
img = image_utils.load_img(path, target_size=(150,
150))
x = image_utils.img_to_array(img) / 255.0
# 在第 0 维添加维度变为 1x150x150x3，和我们模型的输入数据一
样
x = np.expand_dims(x, axis=0)
# np.vstack:按垂直方向（行顺序）堆叠数组构成一个新的数组，我
们一次只有一个数据所以不这样也可以
images = np.vstack([x])
# batch_size 批量大小，程序会分批次地预测测试数据，这样比每
次预测一个样本会快
classes = model.predict(images, batch_size=1)
# classes[0]表示分类概率，大于 0.5 表示分类为 dog，小于 0.5
表示分类为 cat
print(classes[0])
if classes[0] > 0.5:
    print("It is a dog")
else:
    print("It is a cat")
```

实验结果

任务一：

模型数据：

```
20/20 [==============================] - 7s 337ms/step - loss: 0.6682 - acc: 0.6000 - val_loss: 0.6652 - val_acc: 0.5200
Epoch 7/10
20/20 [==============================] - 7s 339ms/step - loss: 0.6718 - acc: 0.5675 - val_loss: 0.6419 - val_acc: 0.6450
Epoch 8/10
20/20 [==============================] - 8s 374ms/step - loss: 0.6491 - acc: 0.5950 - val_loss: 0.6392 - val_acc: 0.5900
Epoch 9/10
20/20 [==============================] - 7s 328ms/step - loss: 0.6402 - acc: 0.6525 - val_loss: 0.6412 - val_acc: 0.6200
Epoch 10/10
20/20 [==============================] - 6s 306ms/step - loss: 0.6602 - acc: 0.6175 - val_loss: 0.6321 - val_acc: 0.6050
```

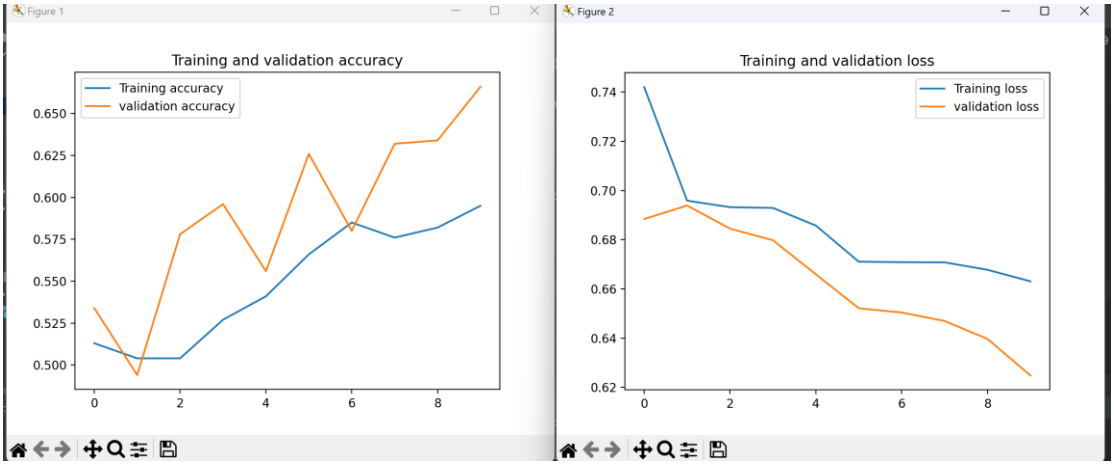


模型预测：

```
[0.8855228]
It is a dog
```

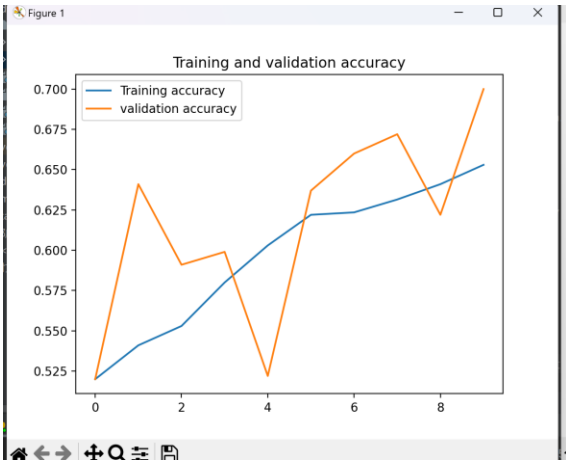
任务二：

训练集，验证集大小：50



```
D:\python3.8.0\pyt
2022-11-25 19:21:1
To enable them in
[0.9326456]
It is a dog
```

训练集，验证集大小：100



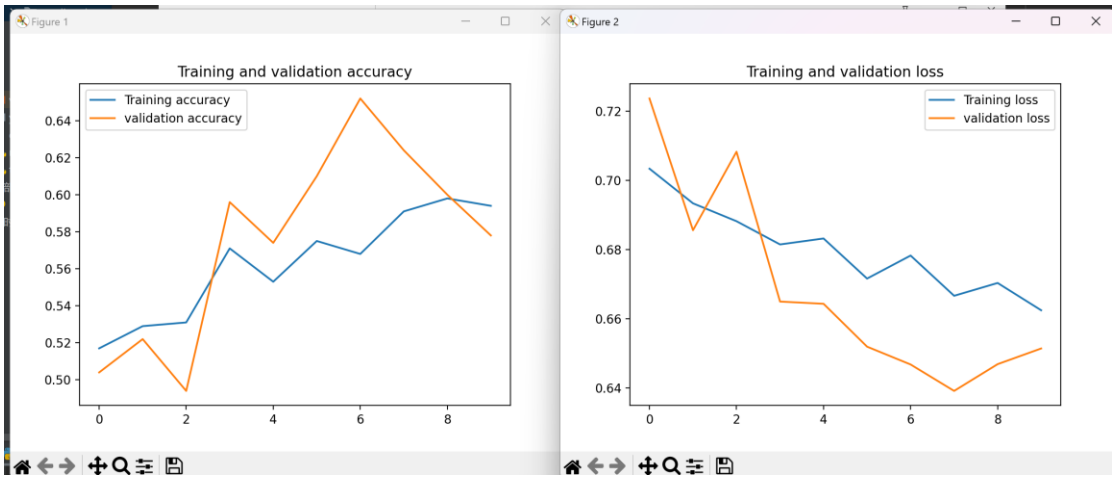

```
To enable them in other
[0.9801865]
It is a dog

进程已结束,退出代码0
```

由不同数量的训练集和验证集可以得知，随着训练轮次的增加，训练验证的准确率都上升，损失值都下降，随着训练集和验证集的数量增加，准确率进一步上升，损失值进一步下降：（注意到纵坐标的数值变化），对预测图片的分类概率也越来越大（偏向正确答案狗）

任务三：

增加网络层数：

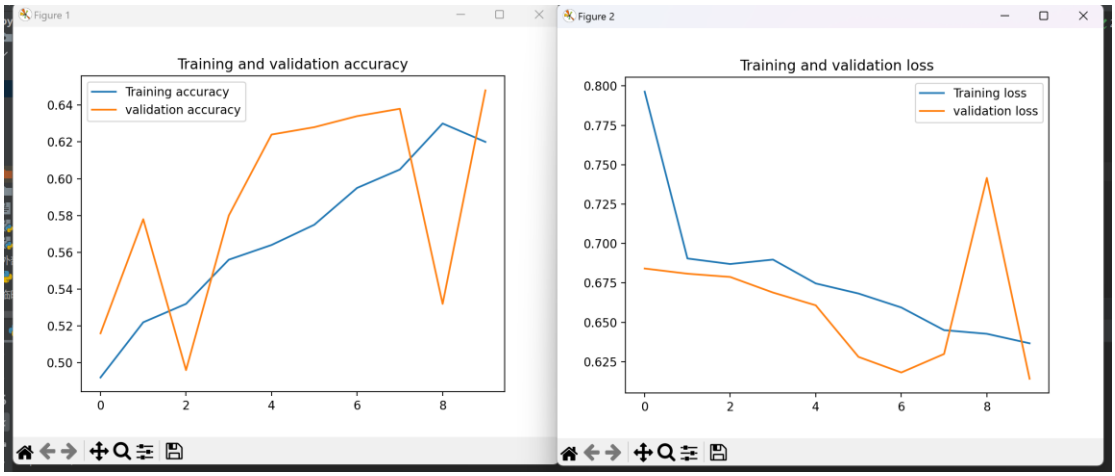
```python
model = tf.keras.models.Sequential([
    # 我们的数据是 150x150 而且是三通道的，所以我们的输入应该设置为这样的格
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input
    tf.keras.layers.MaxPooling2D(2, 2),# 最大池化层
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(256, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.5),# dropout 层通过忽略一般数量特征
    tf.keras.layers.Flatten(),# 全链接层，将多维输入一维化
    tf.keras.layers.Dense(512, activation='relu'),
    # 二分类只需要一个输出
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```




```
To enable them in oth
[0.82858557]
It is a dog

进程已结束,退出代码0
```

修改卷积核数量：

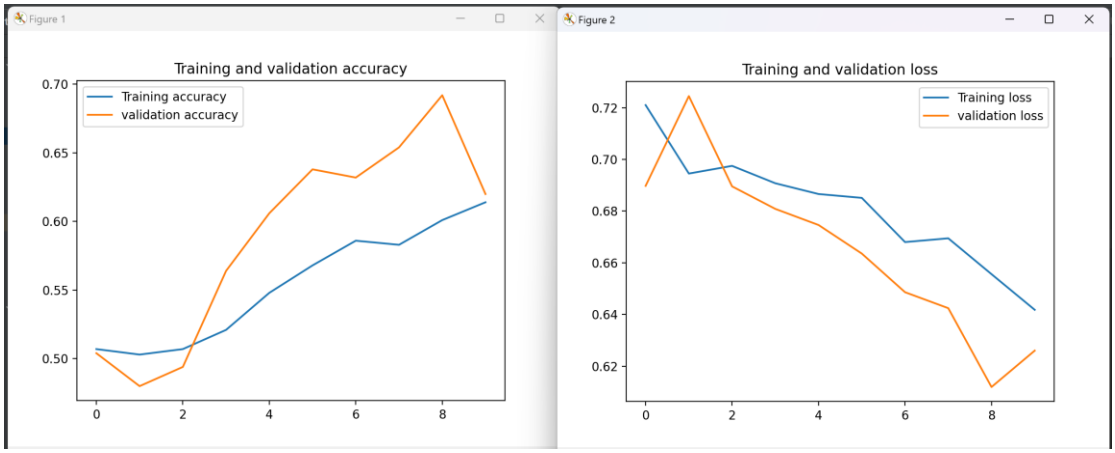
```
model = tf.keras.models.Sequential([
    # 我们的数据是 150x150 而且是三通道的，所以我们的输入应该设置为这样的格式。
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),# 最大池化层
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(256, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.5),# dropout 层通通过忽略一般数量特征，可以减少过拟合现象
    tf.keras.layers.Flatten(),# 全链接层，将多维输入一维化
    tf.keras.layers.Dense(512, activation='relu'),
    # 二分类只需要一个输出
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```



```
[0.9543191]
It is a dog

进程已结束,退出代码0
```

修改卷积核大小：

```
tf.keras.layers.Conv2D(32, (4, 4), activation='relu', input_shape=(
tf.keras.layers.MaxPooling2D(2, 2),# 最大池化层
tf.keras.layers.Conv2D(64, (4, 4), activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
tf.keras.layers.Conv2D(128, (4, 4), activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
tf.keras.layers.Dropout(0.5),# dropout 层通通过忽略一般数量特征，可以减
tf.keras.layers.Flatten(),# 全链接层，将多维输入一维化
tf.keras.layers.Dense(512, activation='relu'),
# 二分类只需要一个输出
tf.keras.layers.Dense(1, activation='sigmoid')
])
```




```
To enable them in
[0.85841256]
It is a dog
```

结论：

网络层数增加一层，对实验结果没有产生明显变化；增加两层，实验结果出现错误。

卷积核数量、修改卷积核大小的增加，模型的准确率少量上升，损失值少量下降。