# 《机器学习》大作业

题目： 基于Tensorflow的物体识别

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **学号** | **班级** | **总成绩(100）** |
| 吴思玮 | 201992326 | 软数1901 |  |
| 宋俊炜 | 201992309 | 软数1902 |  |
| 杨杰 | 201992200 | 软数1901 |  |
| 魏苏州 | 201992142 | 软数1901 |  |

大连理工大学软件学院

2021年12月

## 一．选题背景意义

随着人们对于计算机的深入研究。如今，人工智能已经成为一个非常火热的领域，并且具有众多活跃的研究课题以及惠及生活方方面面的实际应用。在人工智能发展初期，计算机很容易处理一些人类很难甚至是无法解决的问题，这些问题可以用一种形式化的数学规律来描述。当今人们更是希望可以借助人工智能自动地处理一些主观的，非规范性的事物，就比如识别物体。然而人工智能真正面临的任务是那些很难用形式化符号描述的任务，当然对于人类来说很容易执行。举个例子，人们能够轻松识别对方说的话，也能够轻易识别图像中的物体。对于这类问题，计算机却无法给出自己的判断。计算机很擅长助理抽象和形式化的任务，但是人类却觉得这是很困难的脑力劳动。早在上个世纪，计算机就在国际象棋方面战胜了人类选手。但一直到最近几年，计算机才在语音和图像识别任务中达到人类一般水平。通常一个人的思维发育需要海量有关外界的知识。同人类一样，计算机也要获得同样数量级的知识才能表现得智能化。因此，对于我们来说，关键的挑战就是如何将这些主观的、非形式化的知识教会计算机来学习。所以，我们接下来将动手实现日常物体的检测。

## 二．解决思路及方案

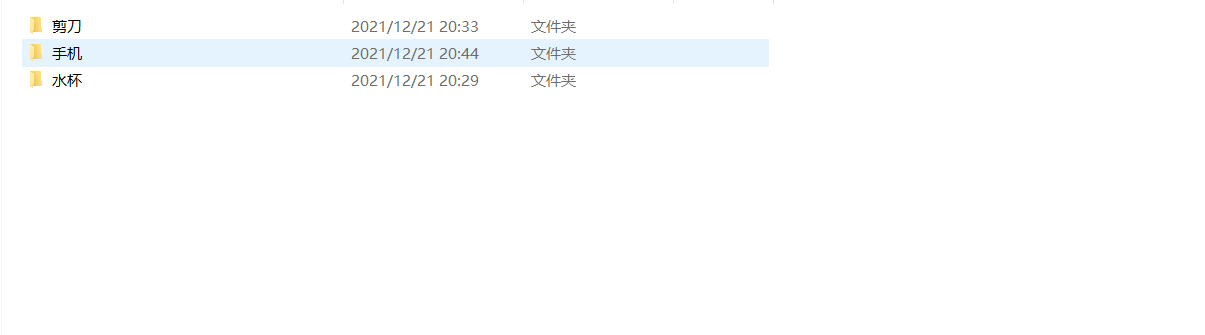
我们通过上网查阅相关资料，了解到目前在图像识别分类领域中表现最好的模型全部都是卷积神经网络。经过比较各个模型，权衡模型大小，运行效率和识别精度等几个方面，我们决定使用VGG16模型。

由于大型神经网络的训练需要巨大的数据和算力，而我们只有小规模的数据集，并且没有足够的算力。因此我们决定使用在ImageNet数据集上预训练好的VGG16网络。由于该网络是在ImageNet上训练的，而ImageNet数据集有1000个类别。我们删除网络中最后的全连接层，并重新添加自己设定的全连接层。然后冻结VGG16网络剩下层的参数。在我们自己的数据集上训练。

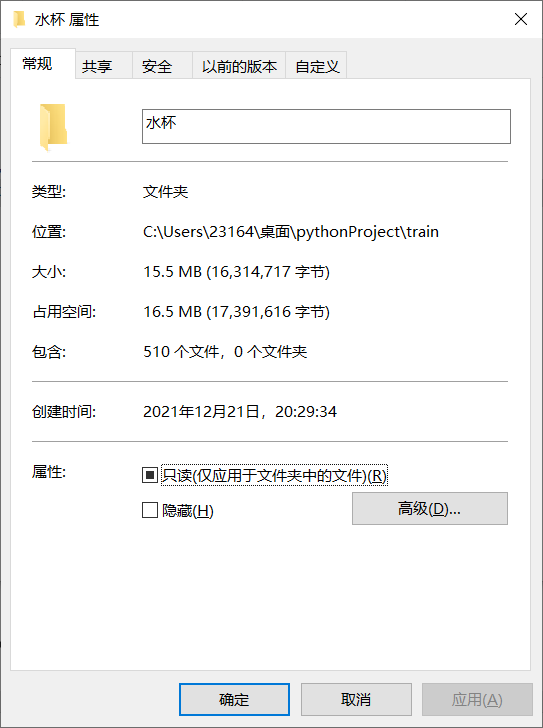
## 三．具体实现

### 1.数据集的准备

我们通过网络爬虫爬取了百度图片，收集了剪刀，水杯，手机三个类别的图像各510张作为训练集，210张作为训练集。





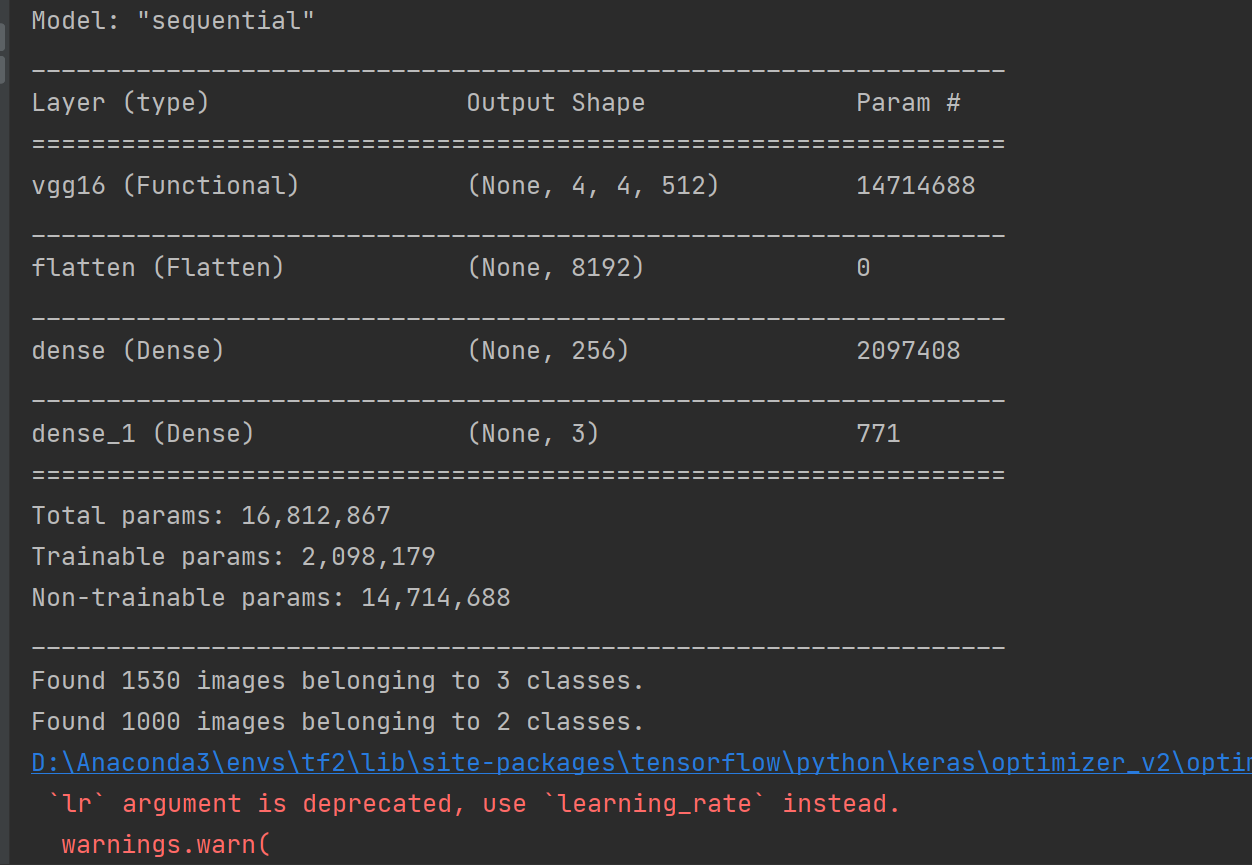


然后我们对数据集进行了清洗，对人眼难以分辨的不符合要求的图片删除，然后重新补充符合要求的图片。利用Tensorflow中的ImageDateGenerator处理图片。ImageDateGenerator可对图片进行缩放和旋转。VGG16原始输入图像形状是244\*244\*3，在构建网络时，我们将输入处理成了150\*150\*3，以此减少参数。因此我们需要利用ImageDateGenerator将图片都处理成150\*150\*3。同时，因为数据量太小，我们需要对数据进行增强。通过旋转，镜像等操作，我们可以得到更多的数据。由于Tensoflow处理的都是张量，因此还需要将图像数据都转化为float32形式的浮点数。同时为了减小训练的误差，对图像像素点数据进行归一化处理。处理完成后图像将被ImageDateGenerator生成器一批一批输入网络中进行训练。

### 2.网络搭建

我们利用Google开发的Tensorflow框架进行网络的搭建，通过Tensorflow中集成的Keras模块加载了预训练的VGG16网络，并删除了网络顶端的全连接层。将处理后的VGG16网络加入到我们自己的网络结构中，然后在最后加上一层Flatten层和两层Dense层，最后一层Dense层输出一个三维的向量，该向量经过softmax操作，向量中的每个数可看作是对应类别的概率。

网络结构如下：



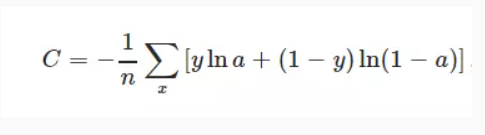
可以看到VGG16层中有大量的参数，这对于我们所拥有的算力来说是难以训练的。因此我们冻结了VGG16层中的所有参数，这样就大大提升了训练的速度。而且由于加载的VGG16是在大数据集上经过预训练的，因此卷积层中学习到的参数对于我们任务的识别也是有帮助的。

### 3.模型训练

**损失函数**：由于该问题是一个多分类的问题，因此我们使用categorical\_crossentropy（交叉熵损失函数)作为损失函数。

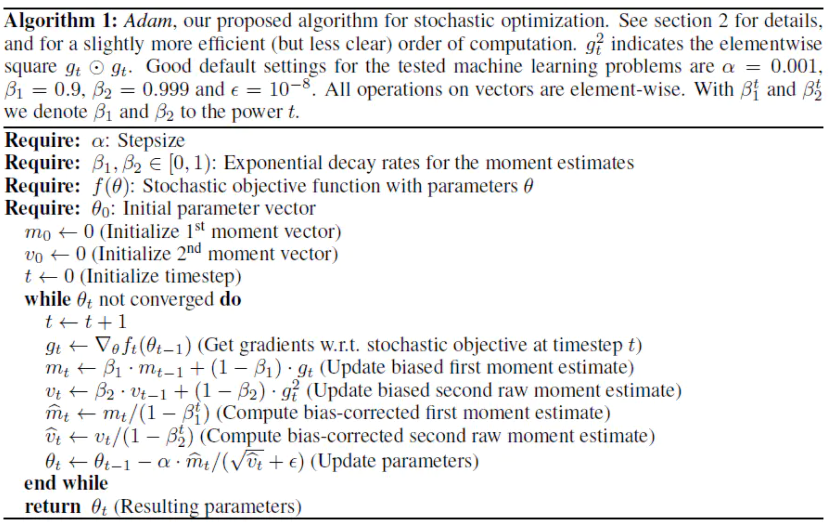
交叉熵是用来评估当前训练得到的概率分布与真实分布的差异情况。它刻画的是实际输出（概率）与期望输出（概率）的距离，也就是交叉熵的值越小，两个概率分布就越接近。

其公式如下：



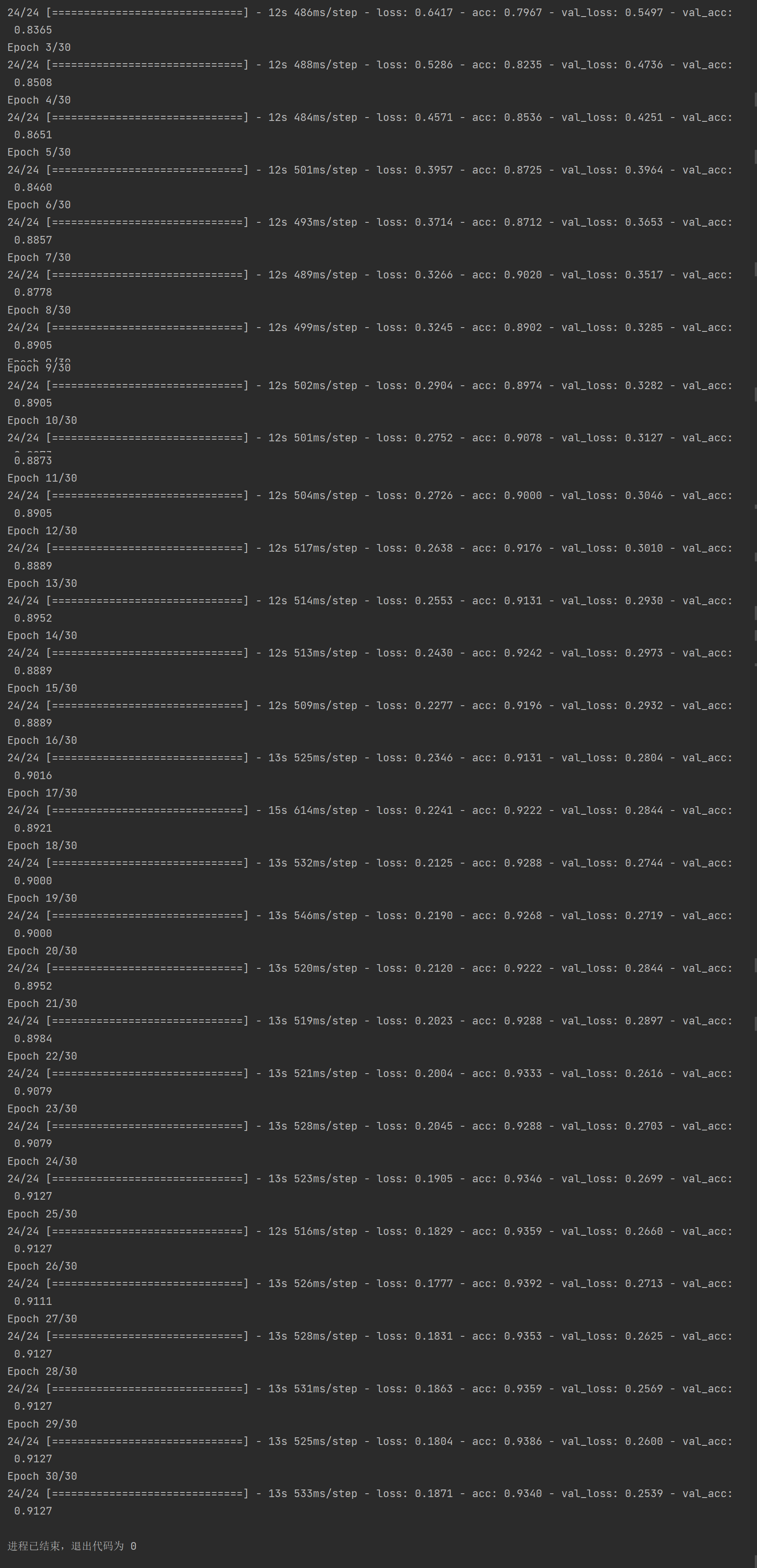
**优化器**：我们使用Adam作为优化器，Adam优化器结合了AdaGrad和RMSProp两种优化算法的优点。对梯度的一阶矩估计（First Moment Estimation，即梯度的均值）和二阶矩估计（Second Moment Estimation，即梯度的未中心化的方差）进行综合考虑，计算出更新步长。因此Adam在很多情况下算作默认工作性能比较优秀的优化器。

其计算过程如下：

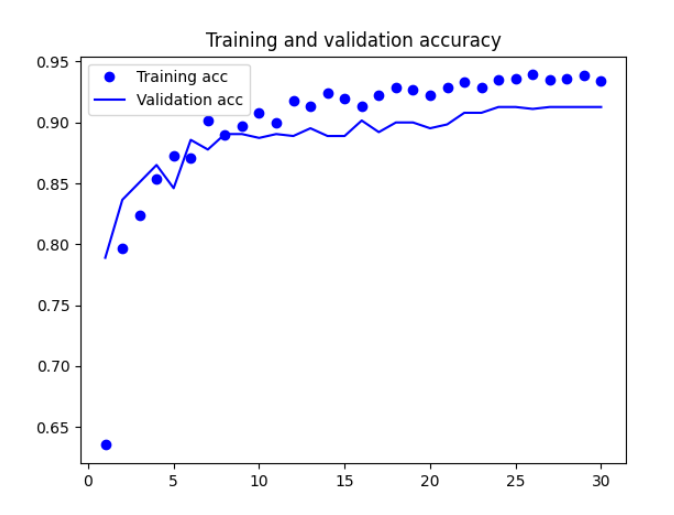
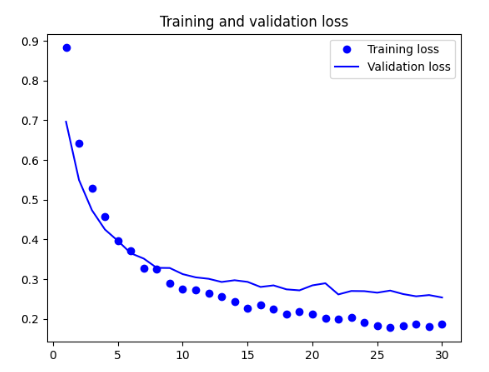


最后我们选取accuracy(准确率)作为评判标准，以batch\_size等于64训练了30个epoch。学习率我们取较小值1e-5。

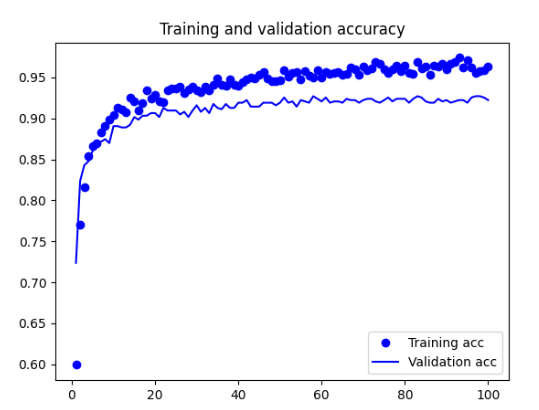
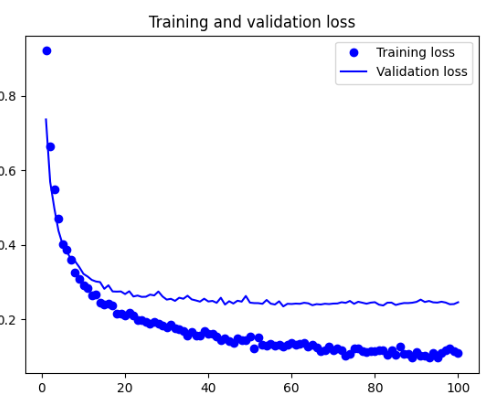
训练过程如下：



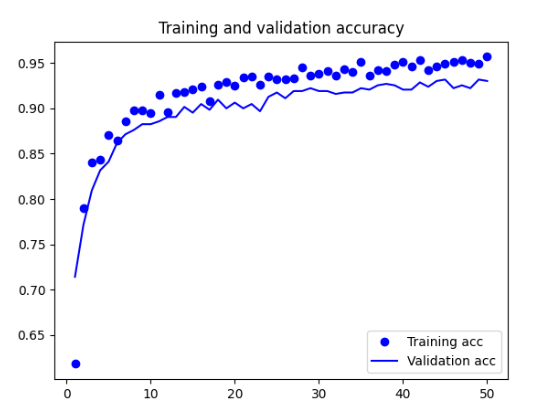
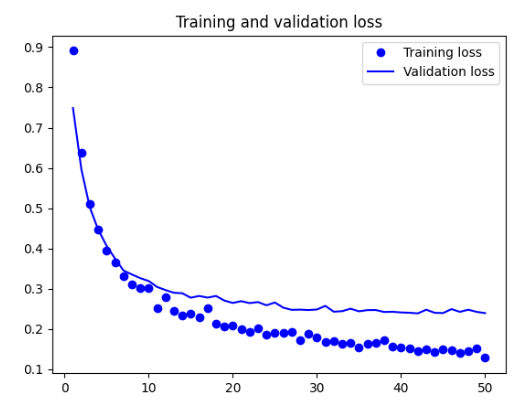
可以看到每一轮耗费的时间大致时12s，随着训练轮数的增加，模型的损失在下降，精度在提升。作出图像：



分析可知，模型存在欠拟合，因此我们将训练轮数提升至100轮。



可以看到验证集精度从第50轮开始就稳定在百分之九十一左右，因此我们选择训练50轮作为最终的模型。

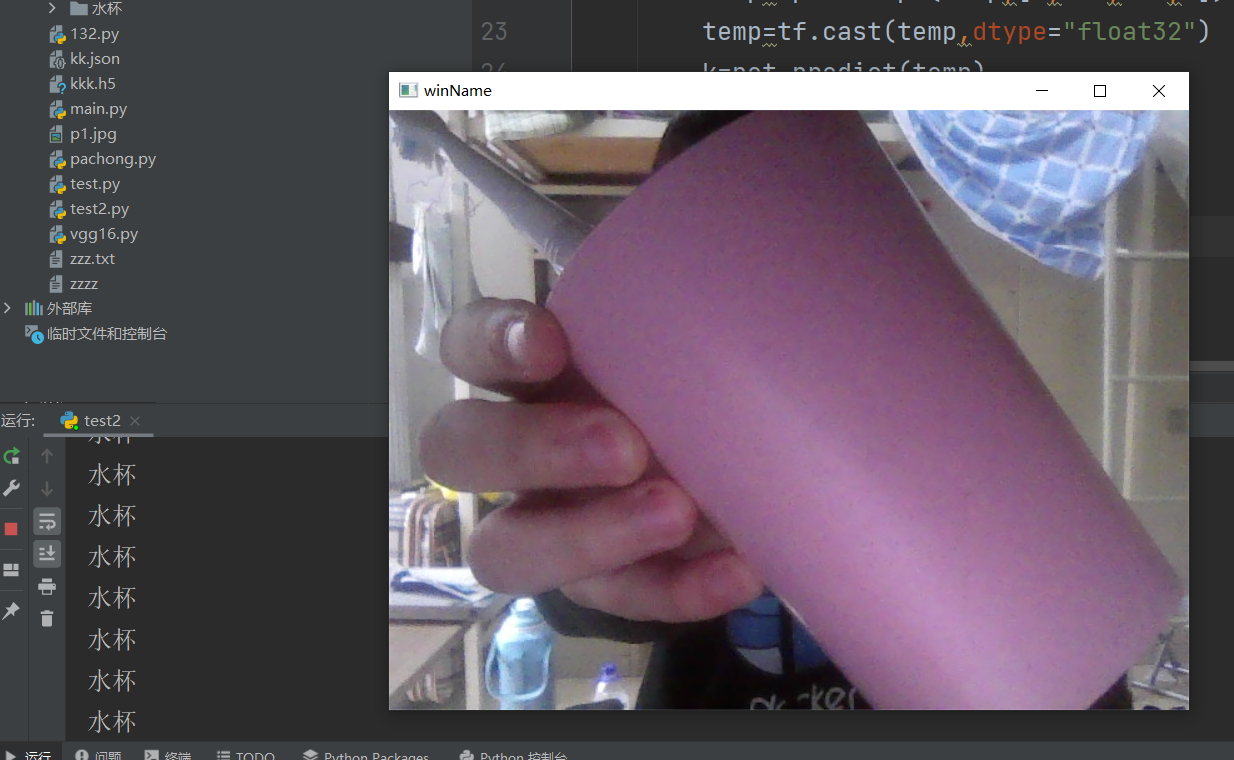


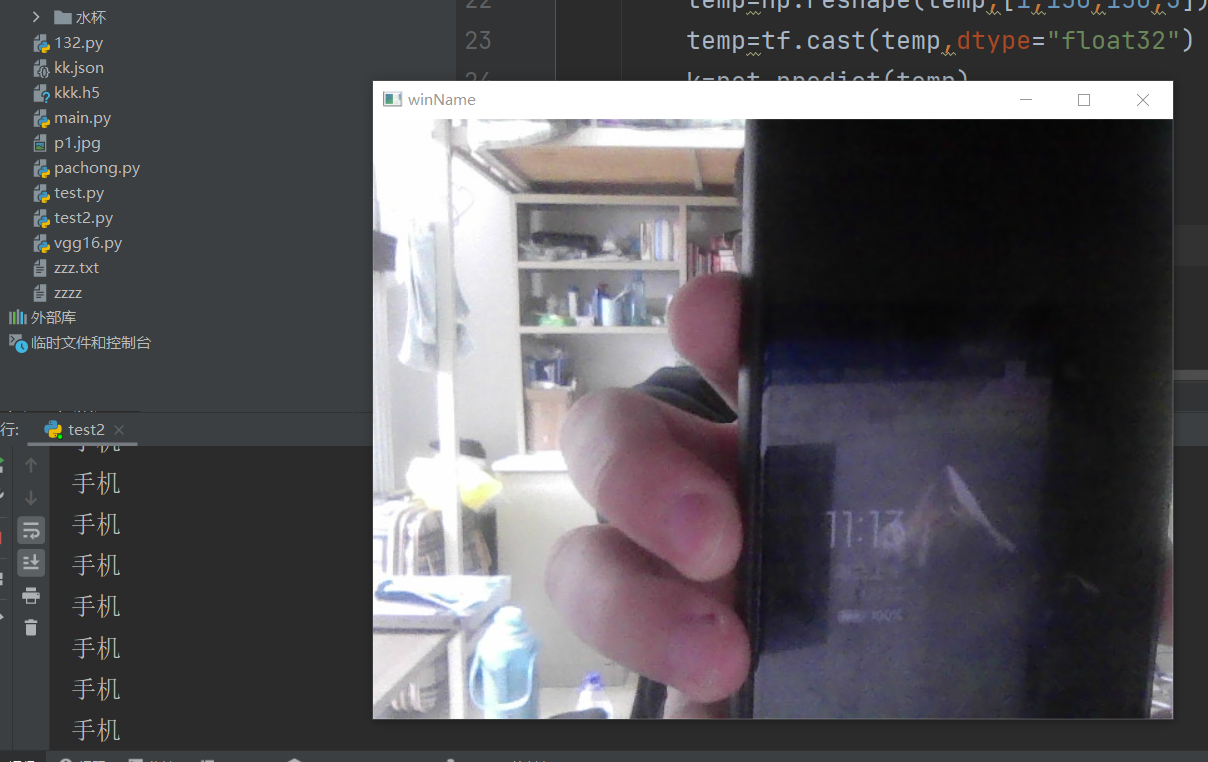
最后将模型保存为.h5文件。

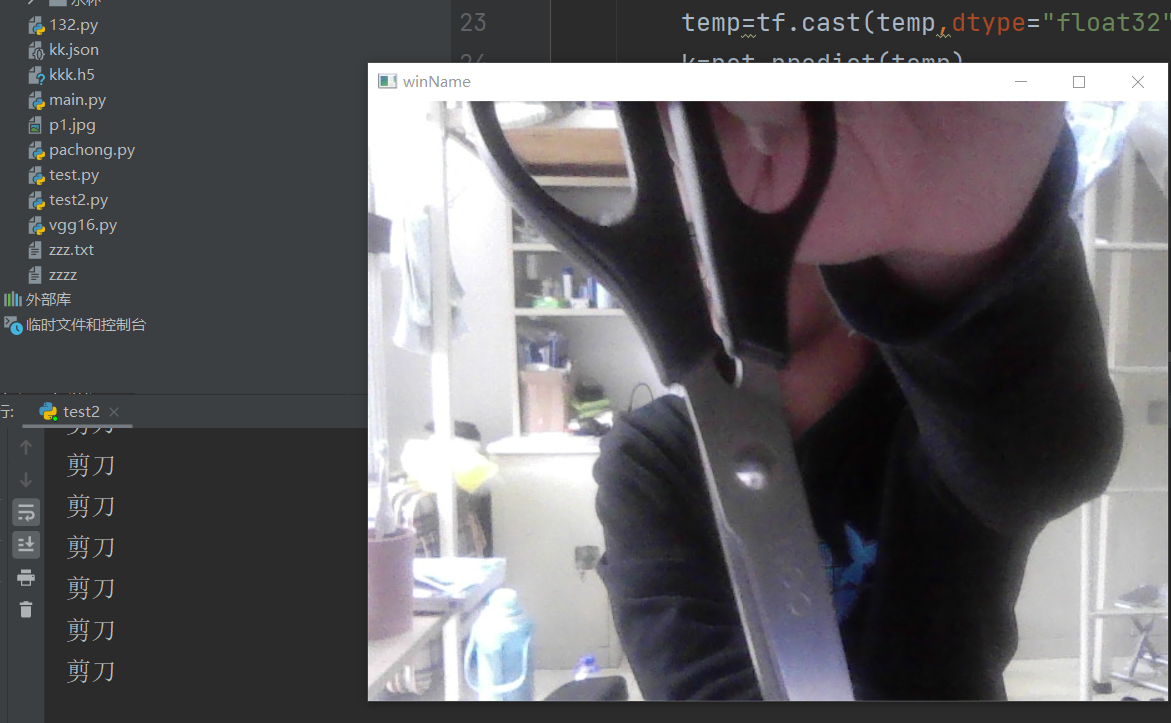
### 4.使用模型

加载保存的.h5文件，将训练好的模型载入内存。然后使用opencv图像处理库，调用电脑摄像头。摄像头拍摄的视频其实是一帧一帧的图片。因此我们可以读取每一张图片，将其经过opencv和numpy处理成网络的输入格式。经过网络处理后，输出其对应的类别，然后打印在屏幕上。这样就实现了通过摄像头对物体的实时分类。

实现效果如图：







可以看到，检测结果比较准确。

## 四．分析总结

通过本次实验，我们可以看到，即使没有庞大的数据集和强大的算力，通过预训练模型，我们同样可以获得效果不错的可用模型。

当然如果想进一步提高模型的准确率和精度，仅仅使用网络爬虫爬取的图片进行训练是不够的，我们需要更大的数据集。同时为了获得更好的效果我们需要尝试不同的超参数，而这需要更多的算力。接下来，我们将尝试不同的网络结构，构建更多的类别，并尝试将模型移植到手机端。

总之，通过本次学习实践，我们深刻体会到了机器学习在我们生活中的应用。在动手时间中我们加深了理解。本次实验不仅检验了我们所学习的知识，也培养了我们如何去把握一件事情，如何去做一件事情，又如何完成一件事情。在实践过程中，与同学们分工设计，和同学们相互探讨，相互学习，相互监督。而这可能是日后最实用的，可谓受益匪浅。在面对社会的挑战，一定要不断的学习、实践。这对于我们的将来工作也会有很大的帮助。

## 附页

### 分工

吴思玮查阅资料和选择模型。

宋俊炜构建网络，并训练模型。

杨杰爬取图片，并对图片进行了筛选。

魏苏州撰写实验报告。

### 爬虫代码：

import requests  
#杯子  
# url1="https://image.baidu.com/search/acjson?tn=resultjson\_com&logid=11600832514977741391&ipn=rj&ct=201326592&is=&fp=result&fr=&word=%E6%9D%AF%E5%AD%90&queryWord=%E6%9D%AF%E5%AD%90&cl=2&lm=-1&ie=utf-8&oe=utf-8&adpicid=&st=&z=&ic=&hd=&latest=&copyright=&s=&se=&tab=&width=&height=&face=&istype=&qc=&nc=1&expermode=&nojc=&isAsync=&pn="  
# url2="&rn=30&gsm=3c&1640072795289="  
#剪刀  
# url1="https://image.baidu.com/search/acjson?tn=resultjson\_com&logid=11224354450780956549&ipn=rj&ct=201326592&is=&fp=result&fr=&word=%E5%89%AA%E5%88%80&queryWord=%E5%89%AA%E5%88%80&cl=2&lm=-1&ie=utf-8&oe=utf-8&adpicid=&st=-1&z=&ic=&hd=&latest=&copyright=&s=&se=&tab=&width=&height=&face=0&istype=2&qc=&nc=1&expermode=&nojc=&isAsync=&pn="  
# url2="&rn=30&gsm=5a&1640087805403="  
#手机  
url1="https://image.baidu.com/search/acjson?tn=resultjson\_com&logid=10844385999817288927&ipn=rj&ct=201326592&is=&fp=result&fr=&word=%E6%89%8B%E6%9C%BA&queryWord=%E6%89%8B%E6%9C%BA&cl=2&lm=-1&ie=utf-8&oe=utf-8&adpicid=&st=-1&z=&ic=&hd=&latest=&copyright=&s=&se=&tab=&width=&height=&face=0&istype=2&qc=&nc=1&expermode=&nojc=&isAsync=&pn="  
url2="&rn=30&gsm=3c&1640088238902="  
header={  
 "user-agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36 Edg/96.0.1054.62"  
}  
cnt=1  
for i in range(7):  
 z=i+20  
 print(z)  
 # if i==14 or i==15 or i==16:  
 # continue  
 url=url1+str(z\*30)+url2  
 r=requests.get(url,headers=header)  
 js=r.json()  
 for j in range(30):  
 url3=js["data"][j]["middleURL"]  
 r2=requests.get(url3,headers=header)  
 f=open("validation\\手机\\"+str(cnt)+".jpg","wb")  
 cnt+=1  
 f.write(r2.content)  
 f.close()

### 模型训练代码：

import tensorflow as tf  
import tensorflow.keras as keras  
import numpy as np  
import os, shutil  
  
from keras\_preprocessing.image import ImageDataGenerator  
  
base\_d = "D:/cat\_vs\_dog"  
# os.mkdiniddr(base\_d)  
train\_d = os.path.join(base\_d, "train")  
val\_d = os.path.join(base\_d, "val")  
test\_d = os.path.join(base\_d, "test")  
# os.mkdir(train\_d)  
# os.mkdir(test\_d)  
# os.mkdir(val\_d)  
  
train\_cat\_d = os.path.join(train\_d, "cat")  
val\_cat\_d = os.path.join(val\_d, "cat")  
test\_cat\_d = os.path.join(test\_d, "cat")  
  
train\_dog\_d = os.path.join(train\_d, "dog")  
val\_dog\_d = os.path.join(val\_d, "dog")  
test\_dog\_d = os.path.join(test\_d, "dog")  
  
# os.mkdir(train\_dog\_d)  
# os.mkdir(test\_dog\_d)  
# os.mkdir(val\_dog\_d)  
#  
# os.mkdir(train\_cat\_d)  
# os.mkdir(test\_cat\_d)  
# os.mkdir(val\_cat\_d)  
  
# for i in range(2000):  
# f=r"D:\浏览器下载\dogs-vs-cats\train"  
# s=os.path.join(f,"cat."+str(i)+".jpg")  
# if (i<1000):  
# dst=os.path.join(train\_cat\_d,"cat."+str(i)+".jpg")  
# shutil.copyfile(s,dst)  
# elif (i<1500):  
# dst=os.path.join(val\_cat\_d,"cat."+str(i)+".jpg")  
# shutil.copyfile(s,dst)  
# else:  
# dst = os.path.join(test\_cat\_d, "cat." + str(i) + ".jpg")  
# shutil.copyfile(s, dst)  
#  
# for i in range(2000):  
# f=r"D:\浏览器下载\dogs-vs-cats\train"  
# s=os.path.join(f,"dog."+str(i)+".jpg")  
# if (i<1000):  
# dst=os.path.join(train\_dog\_d,"dog."+str(i)+".jpg")  
# shutil.copyfile(s,dst)  
# elif (i<1500):  
# dst=os.path.join(val\_dog\_d,"dog."+str(i)+".jpg")  
# shutil.copyfile(s,dst)  
# else:  
# dst = os.path.join(test\_dog\_d, "dog." + str(i) + ".jpg")  
# shutil.copyfile(s, dst)  
  
net = keras.applications.VGG16(include\_top=False, weights="imagenet", input\_shape=(150, 150, 3))  
model = keras.models.Sequential()  
model.add(net)  
model.add(keras.layers.Flatten())  
model.add(keras.layers.Dense(256, activation="relu"))  
model.add(keras.layers.Dense(3, activation="softmax"))  
net.trainable = False  
model.summary()  
  
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')  
train\_generator = train\_datagen.flow\_from\_directory(  
 "train",  
 target\_size=(150, 150),  
 batch\_size=64,  
 class\_mode="categorical"  
)  
test\_datagen = ImageDataGenerator(rescale=1. / 255)  
validation\_generator = test\_datagen.flow\_from\_directory(  
 "validation",  
 target\_size=(150, 150),  
 batch\_size=64,  
 class\_mode='categorical')  
model.compile(loss='categorical\_crossentropy',  
 optimizer=keras.optimizers.RMSprop(lr=2e-5),  
 metrics=['acc'])  
history = model.fit(  
 train\_generator,  
 epochs=100,  
 validation\_data=validation\_generator)  
  
model.save("kkk.h5")  
  
import matplotlib.pyplot as plt  
acc = history.history['acc']  
val\_acc = history.history['val\_acc']  
loss = history.history['loss']  
val\_loss = history.history['val\_loss']  
epochs = range(1, len(acc) + 1)  
plt.plot(epochs, acc, 'bo', label='Training acc')  
plt.plot(epochs, val\_acc, 'b', label='Validation acc')  
plt.title('Training and validation accuracy')  
plt.legend()  
plt.figure()  
plt.plot(epochs, loss, 'bo', label='Training loss')  
plt.plot(epochs, val\_loss, 'b', label='Validation loss')  
plt.title('Training and validation loss')  
plt.legend()  
plt.show()

### 模型使用代码：

import tensorflow as tf  
import tensorflow.keras as keras  
import time  
  
#conv\_base = keras.applications.VGG16(weights="imagenet", include\_top="True", input\_shape=(224, 224, 3))  
# net=keras.Sequential()  
# net.add(conv\_base)  
# net.add(keras.layers.Softmax)  
net=keras.models.load\_model("kkk.h5")  
import numpy as np  
import cv2  
  
cap = cv2.VideoCapture(0) # 我的摄像头位置是700  
cv2.waitKey(5) # 等待5ms  
  
while cap.isOpened(): # 也可以是 while True  
 ok, img = cap.read()  
 if ok: # 若读取到图像再进行显示  
 cv2.imshow('winName', img)  
 temp=cv2.resize(img,dsize=(150,150))  
 temp = np.array(temp)  
 temp=np.reshape(temp,[1,150,150,3])  
 temp=tf.cast(temp,dtype="float32")  
 k=net.predict(temp)  
 k=tf.argmax(k,axis=1)  
 k=tf.squeeze(k)  
 if (int(k)==0):  
 print("剪刀")  
 elif(int(k)==1):  
 print("手机")  
 else:  
 print("水杯")  
  
 if cv2.waitKey(5) == ord('q'): # 等到5ms检测是否退出，同时防止读取太快  
 cv2.destroyWindow('winName')  
 break  
  
cap.release()