图像压缩对图像识别结果影响探究——以“猫狗”识别算法为例

1.摘要

随着机器学习技术的发展，深度学习算法渐渐成为了“模式识别”的主流。其中图像识别的服务最为热门。本实验针对的问题是：**特定图像经过压缩后由识别算法处理，是否可以维持识别结果准确？** 为此，本实验分为三个部分且均已实现：1、使用Matlab开发基于DCT变换的图像压缩算法，和基于K-L变换的图像压缩算法。经压缩后的图片体积大幅减小。2、使用Python中的Keras深度学习框架开发并训练一准确度较高的“猫狗”识别算法。实现输入一张图片，输出分类结果。 3、进行试验，分别将随机选择的10张猫的图片和10张狗的图片经压缩处理前后分别投入算法，查看准确率。实验结果为：图像经压缩后算法依旧可以识别出图像，但压缩后整体识别准确率略小于压缩前。 后期可以针对不同的识别算法和压缩算法进行实验。并尝试更好的在原理上解释实验结果。

2. 引言

本门课程学习了信号处理和模式识别的基本知识。模式识别的根本目的就是通过算法实现或超过人类分辨事物的水平。在此任务中无论是否压缩，正常人类均可轻易区分猫与狗。我们希望知道算法是否具备这样的能力。本实验可以帮助更好的理解图像分类算法的本质。同时，本实验也有可应用的实际意义。若压缩后的识别表现依旧较好。在为用户提供图像识别服务时，用户端可将压缩后的图像发送至服务器进行识别。既节约了用户的流量，又减轻了服务器存储的压力。且不必训练新的模型 现有相关工作主要集中在开发兼顾图片质量与压缩效率的压缩算法。

3. 实验设计

3.1 使用离散余弦变换（DCT）的图像压缩算法

DCT算法是一种正交变换，它可以将图像由空间域变换到频域上，由此可以对频域进行处理。而图像的大部分信息均集中在低频部分。所以去除掉高频部分的系数，再进行逆DCT变换就得到了压缩后的图像。以较少信息损失量的代价获得减小存储图片所需的数据量，达到压缩图片的目的。对于一张(m,n)的二维图片f（x,y)，其离散余弦变换为

$$F(u,v)=\alpha(u)\beta(v)\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}f(u,v)\cos\frac{(2x+1)u\pi}{2m}\cos\frac{(2y+1)v\pi}{2n}$$

其中

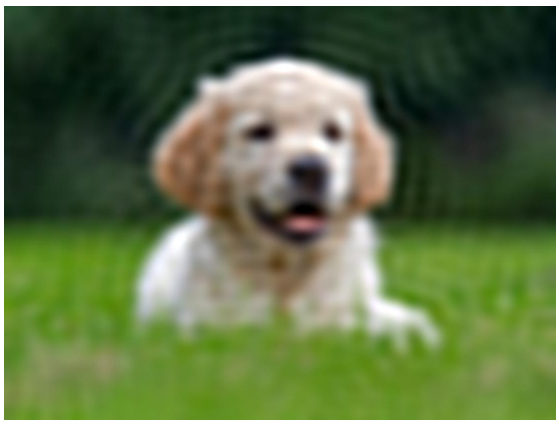
$$\alpha(u)=\left\{\frac{1}{\sqrt{m}},u=0|\sqrt{\frac{2}{m}},u=1\dots m-1\right\},\ \beta(v)=\left\{\frac{1}{\sqrt{n}},v=0|\sqrt{\frac{2}{n}},v=1\dots n-1\right\}$$

本质上来说 DCT算法是图像空间域的低通滤波器。而其具体实现方法分为通过FFT计算和通过变换矩阵计算。在代码实现时，选择通过FFT计算的方式。将彩色图像RGB三通道分别处理后合并得到压缩图像。

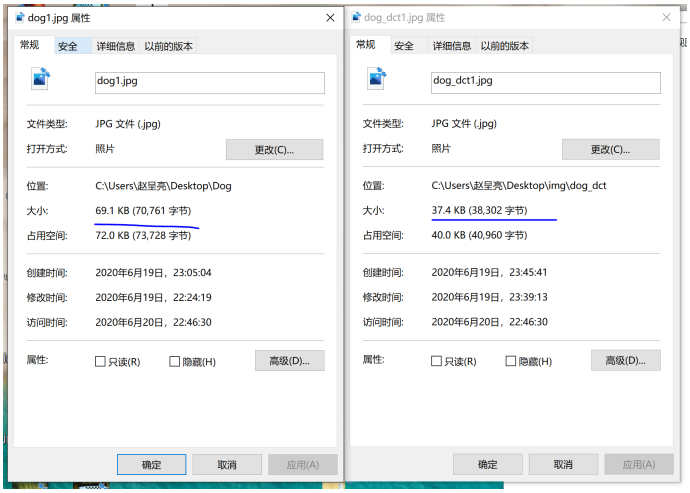
压缩结果

原图

压缩后（压缩比例0.03）



压缩前后体积对比



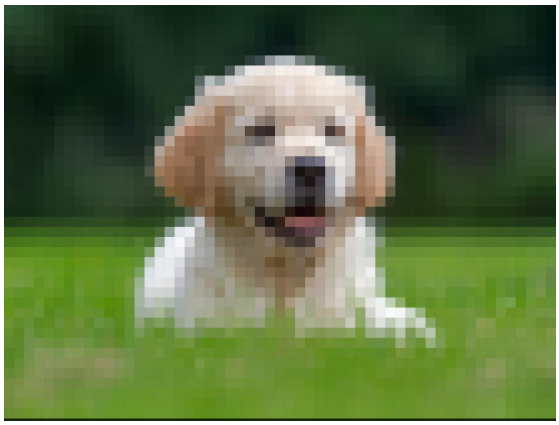
可以看出，图像清晰度虽然降低，但依旧可辨认出这是一张狗的图片。而且图像所占体积大幅缩小。在测试中使用的20张图片中，体积平均缩小40%。

3.2 使用 K-L变换的图像压缩算法

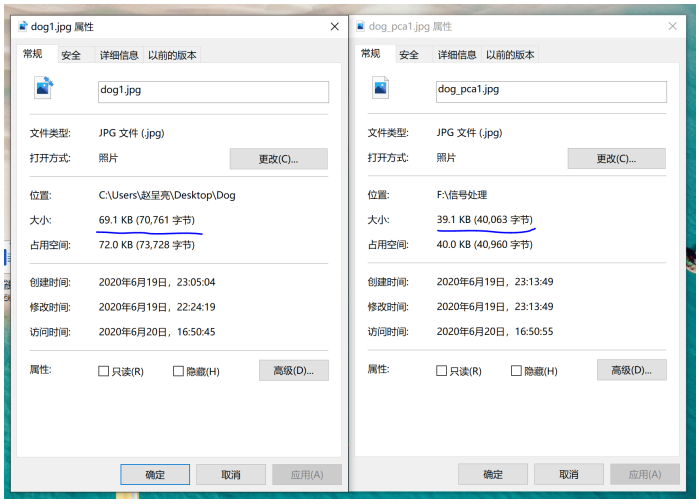
K-L变换，又称主成分分析（PCA），其目的是对图像的的协方差矩阵进行特征值分解，保留较大的特征值及其特征向量。计算“主成分”后对图像进行变换。在压缩图像的过程中，K-L变换的目标是去除原有图像中的相关性，从而将图像信息集中在少数分量上（选取的主成分）。从而达到压缩图像的效果。具体实现步骤为：1、获取图像，并分割为多个小块。将每个小块中的点依次排列化为列向量。2、将第一步得到的矩阵中心化并计算协方差矩阵。3、计算协方差矩阵的特征值和特征向量。4，保留较大的特征值和其对应的特征向量 并变换。 在实验中，保留的主成分个数越少，图片压缩后体积越小，图像也会出现失真。

原图

压缩后（保留1个主成分）



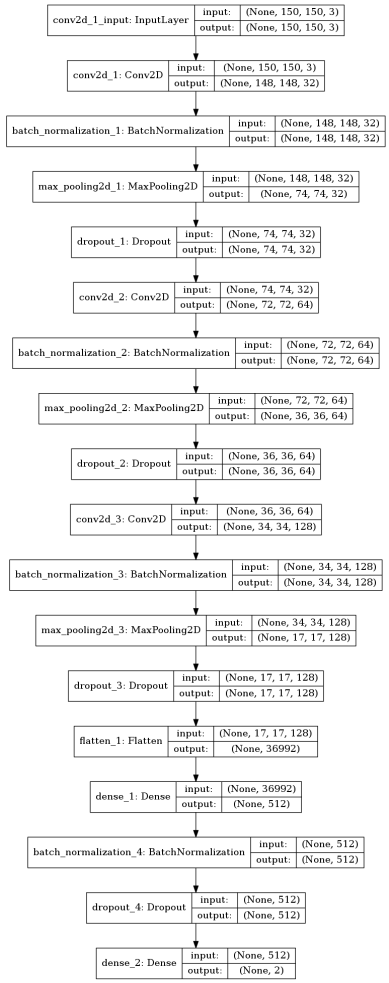
压缩前后体积对比



可以看到，图像体积大幅减小。同时，图像虽有失真，但依旧可以判断分类为狗。两种压缩方法虽在压缩体积上有相似的效果。但图像经两种不同的方法压缩后图像会出现不同。经DCT变换后图像呈现“晕染”的效果。经K-L变换后图像呈现“马赛克”效果。 这是由于两种算法的根本区别所导致。 DCT变换过滤掉了图片中高频的部分，所以图片还保留原有轮廓，但清晰度降低。而K-L变换时，由于将预先将图像分块，所以呈现“马赛克”状。

3.3 训练模型

模型的任务是判断输入图片的分类，所以选择卷积神经网络（CNN）作为模型。训练数据集选自Kaggle数据科学竞赛平台数据集中包含25000张图片（12500张猫的图片，125000张狗的图片）。而网络具体参数如下。在的已有成功基础上经过改动参数调试多次得到。



由于此网络有较大的深度，由于电脑性能受限。不能选择整个数据集训练（实测训练一个epoch耗时半个小时以上，整个训练过程过程预计将达到10个小时）。所以实际过程中，随机在数据集中选择400张图片（猫狗各200张）进行训练。在训练之前对图片进行预处理，分割成大小为150，并将R，G，B三通道分解最终得到（150,150,3）的数组。由于是二分类问题损失函数（Loss Function）选为Cross Entrop。为了防止结果过拟合，在网络中添加Dropout层与normalization层。而每一个卷积层使用的卷积核大小（filter）及步长（stride）均经过调整。 整个训练过程使用Python中的Keras深度学习框架。

4.实验过程及结果

进行实验时，随机在关键词为“猫”，“狗”的搜索结果中，各随机选取10张图片（共20张）作为测试集。之后将这20张图片分别经过DCT变换压缩，和K-L变换压缩总共得到40张压缩后的图片。

第一步 将未压缩的20张图片放入算法，查看结果

In [6]:
```
for i in range(1,11):
    image_path="/root/cat/Dog/dog"+str(i)+".jpg"

    image = tf.keras.preprocessing.image.load_img(image_path, target_size=(SIZE, SIZE))
    image = tf.keras.preprocessing.image.img_to_array(image)
    image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
    image = np.expand_dims(image, axis=0)
    if (catmodel.predict(image)>0.5):
        print("dog")
    else:
        print("cat")
```

```
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
```

In [3]:
```
catmodel=load_model("cats_and_dogs_small_4.h5")
```

In [5]:
```
for i in range(1,11):
    image_path="/root/cat/Cat/cat"+str(i)+".jpg"

    image = tf.keras.preprocessing.image.load_img(image_path, target_size=(SIZE, SIZE))
    image = tf.keras.preprocessing.image.img_to_array(image)
    image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
    image = np.expand_dims(image, axis=0)
    if (catmodel.predict(image)>0.5):
        print("dog")
    else:
        print("cat")
```

```
cat
cat
cat
cat
cat
cat
cat
cat
cat
cat
```

可以看到，20张图片（10张猫，10张狗）识别正确率为100%。这证明了算法的准确性。

第二步 将经过K-L算法压缩后的图片放入算法识别，查看结果

In [9]:
```
print("result")
for i in range(1,11):
    image_path="/root/cat/img/dog_pca/dog_pca"+str(i)+".jpg"

    image = tf.keras.preprocessing.image.load_img(image_path, target_size=(SIZE, SIZE))
    image = tf.keras.preprocessing.image.img_to_array(image)
    image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
    image = np.expand_dims(image, axis=0)

    if (catmodel.predict(image)>0.5):
        print("dog")
    else:
        print("cat")
```

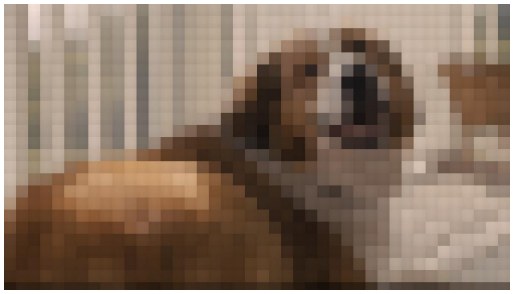
```
result
dog
cat
dog
dog
dog
dog
dog
dog
dog
dog
```

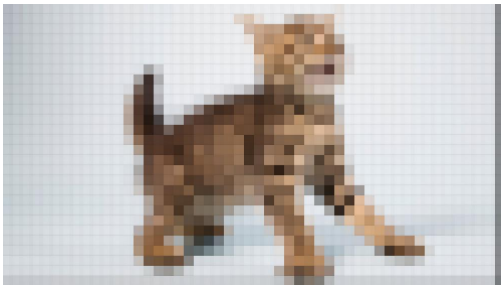
In [10]:
```
print("result")
for i in range(1,11):
    image_path="/root/cat/img/cat_pca/cat_pca"+str(i)+".jpg"

    image = tf.keras.preprocessing.image.load_img(image_path, target_size=(SIZE, SIZE))
    image = tf.keras.preprocessing.image.img_to_array(image)
    image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
    image = np.expand_dims(image, axis=0)

    if (catmodel.predict(image)>0.5):
        print("dog")
    else:
        print("cat")
```

```
result
cat
cat
cat
cat
cat
cat
cat
cat
cat
dog
```

结果显示，同样的图片经过压缩后再经算法识别，出现了误分类的情况，但大部分图片已经可以被算法正确识别。

以下是两张被误分类的照片。人类依旧可以判断出分类。但算法在处理压缩图像时未能分辨。





第三步 将经过DCT算法压缩后的图片放入算法识别，查看结果

In [12]:
```
print("result")
for i in range(1,11):
    image_path="/root/cat/img/dog_dct/dog_dct"+str(i)+".jpg"

    image = tf.keras.preprocessing.image.load_img(image_path, target_size=(SIZE, SIZE))
    image = tf.keras.preprocessing.image.img_to_array(image)
    image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
    image = np.expand_dims(image, axis=0)

    if (catmodel.predict(image)>0.5):
        print("dog")
    else:
        print("cat")
```

```
result
dog
cat
dog
dog
dog
dog
cat
dog
dog
dog
```

In [11]:
```
print("result")
for i in range(1,11):
    image_path="/root/cat/img/cat_dct/cat_dct"+str(i)+".jpg"

    image = tf.keras.preprocessing.image.load_img(image_path, target_size=(SIZE, SIZE))
    image = tf.keras.preprocessing.image.img_to_array(image)
    image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
    image = np.expand_dims(image, axis=0)

    if (catmodel.predict(image)>0.5):
        print("dog")
    else:
        print("cat")
```

```
result
cat
cat
cat
cat
cat
cat
cat
cat
dog
dog
```

结果显示，算法误分类的图片个数增加。

以下为误分类的图片

第二张图片过于模糊，因此未能识别。而第三四张图片较好分辨。进一步检查算法的原始输出

In [9]:
```
print("result")
for i in range(1,11):
    image_path="/root/cat/img/cat_dct/cat_dct"+str(i)+".jpg"

    image = tf.keras.preprocessing.image.load_img(image_path, target_size=(SIZE, SIZE))
    image = tf.keras.preprocessing.image.img_to_array(image)
    image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
    image = np.expand_dims(image, axis=0)
    print(catmodel.predict(image))

#     if (catmodel.predict(image)>0.5):
#         print("dog")
#     else:
#         print("cat")
```

```
result
[[0.01709601]]
[[0.00426771]]
[[2.3389624e-05]]
[[7.460505e-07]]
[[1.8851372e-06]]
[[0.01038889]]
[[1.7795485e-06]]
[[1.1544432e-06]]
[[0.5286942]]
[[0.51328343]]
```

算法对于这里两张图片的判别输出均十分接近0.5（盲猜的概率），这表示算法并不确定其分类。

5.结论及思考

由上述结果，可以得出一个粗略的结论：图像经过压缩后，会有部分信息损失，进而可能会导致算法无法正确识别出结果。而为了得到更准确的结论，下一步需要扩大试验规模，用更多图片在更多算法上进行测试。 本次实验帮助我理解了识别算法和压缩算法的本质。 在传统的模式识别领域，特征提取是一项重要的工作，他与信号处理密不可分。往往需要在空域、频域上计算各种特征。进而运用这些特征而去判断所属的类别。而新兴的深度学习模型本质上实现了**特征提取自动化** 。但算法自动提取的特征往往未知。 本次实验中所使用的CNN中的**卷积层的本质是二维图像滤波器** 它的作用是自定的提取并

学习图像的部分特征（信息）。 训练好的模型在处理输入图像时，首先对输入图像进行卷积从而自动提取提取，过滤出图像中的特定信息后，经过全连接层进行种类判别。

在电脑中存储信息需要物理空间。压缩算法的目的是在去除掉部分信息以换取节约的空间，这不可避免的会造成信息损失。而这可能会使某些**会被模型提取出的信息**被删去。 因此，压缩后的图像在进入算法时，算法提取出的信息与原图像差别较大。导致识别结果出错。 大部分图片压缩过程未把此部分信息删去，所以识别结果依然准确。

6.附录

使用软件

- **MATLAB2019b**
- **Python 所用包为：Numpy, Tensorflow, Keras, Pandas**

程序1：DCT压缩算法

```
clc,clear
ratio = 0.03;
Image = imread('C:\Users\赵呈亮\Desktop\Dog\dog10.jpg');
for k=1:3%分为RGB三个通道 分别处理
doubleData = im2double(Image(:,:,k));%读取图像
dctSeries = dct2(doubleData);%以fft的方式进行dct运算

[rows, cols] = size(doubleData);
for i = 1:rows%按行列检查
    for j = 1:cols
        if (i+j>(rows+cols)*ratio)
            dctSeries(i,j) = 0;%过滤掉高频部分
        end
    end
end
y = idct2(dctSeries);%反dct变换，还原图像

imge(:,:,k)=y;%写入每个通道
end
imshow(imge);%展示图片
imwrite(imge,strcat("dog_dct10.jpg"))%存储图片
imge(:,:,:)=0;%变量清0
fprintf('压缩比例为%f时', ratio)
```

程序2：K-L压缩算法

```
function [I_pca,ratio,contribution]=pcaimage(I,pset,block)

X=im2col(double(I),block,'distinct')';
[n,p]=size(X);
m=min(pset,p);%防止主成分个数超过范围

[E,D]=eig(X'*X);%计算协方差矩阵的特征向量与特征值对角矩阵
for i=1:size(E,2)%按列计数
    [~,idx]=max(abs(E(:,i)));
    E(:,i)=V(:,i)*sign(E(idx,i));
end
```

```
[lambda,locs]=sort(diag(D),'descend');%locs排序前的索引
E=E(:,locs);
coef=E(:,1:m);%变换系数矩阵
score=X*V(:,1:m);%主成分分析得分
contribution=sum(lambda(1:m))/sum(lambda);%主成分分析中的贡献率


X=score*coef';%反变换
I_pca=cast(col2im(X',block,size(I),'distinct'),class(I));%合并
ratio=n*p/(n*m+p*m);%计算压缩率
```

```
clc; clear all; close all;
%for k=1:10
k=1
I=imread(strcat('C:\Users\赵呈亮\Desktop\Cat\cat',num2str(k),'.jpg'));
R=I(:,:,1);%分别读取RGB三通道
G=I(:,:,2);
B=I(:,:,3);
figure('Units', 'Normalized', 'Position', [0 0 1 1]);
p=1;
[IR,~,~]=pcaimage(R,p,[20, 20]);
[IG,~,~]=pcaimage(G,p,[20, 20]);
[IB,ratio,contribution]=pcaimage(B,p,[20, 20]);%三通道分别进行K-L变换
Img=cat(3,IR,IG,IB);#合并三通道
imshow(Img)
title(['主成分个数=',num2str(p)]);
imwrite(Img,strcat("cat_pca",num2str(k),".jpg"))
%end
```

程序3：模型训练（部分参考https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data）

```
import os
import cv2
import re
import random
import numpy as np
import pandas as pd#pandas
from keras import layers, models, optimizers#
from keras import backend as K
from sklearn.model_selection import train_test_split#导入需要的包
img_width = 150#图像裁剪后的大小
img_height = 150
CHANNELS=3
TRAIN_DIR = '../root/IMG/'
train_dogs = [TRAIN_DIR+i for i in os.listdir(TRAIN_DIR) if "dog" in i]
train_cats = [TRAIN_DIR+i for i in os.listdir(TRAIN_DIR) if "cat" in i]# 读取训练集 区分猫狗
train_img = train_dogs[:200] + train_cats[:200]#选取图片各200
random.shuffle(train_img)
test_images =  test_img[:25]

def read_img(file_path):
    img = cv2.imread(file_path, cv2.IMREAD_COLOR) #cv2.IMREAD_GRAYSCALE
```

```
    return cv2.resize(img, (img_width, img_height),
interpolation=cv2.INTER_CUBIC)#将图像转为数组


def prep_data(images):
    count = len(images)
    data = np.ndarray((count, CHANNELS, img_width, image_height),
dtype=np.uint8)

    for i, image_file in enumerate(images):
        image = read_img(image_file)
        data[i] = image.T
        if i%250 == 0: print('Processed {} of {}'.format(i, count))

    return data

train = prep_data(train_img)
labels = []
for i in train_images:
    if 'dog' in i:
        labels.append(1)#狗为1
    else:
        labels.append(0)#猫为0



from keras.models import Sequential
from keras.layers import Conv2D,  Dropout, MaxPooling2D,, Dense, Activation,
BatchNormalization
#定义序列模型
cat_dog_model = Sequential()#初始化

cat_dog_model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150,
3)))#第一层卷积
cat_dog_model.add(BatchNormalization())#防止过拟合 normalization层
cat_dog_model.add(MaxPooling2D(pool_size=(2, 2)))#pooling层 缩小卷积层大小
cat_dog_model.add(Dropout(0.25))#随机失活层 防止过拟合

cat_dog_model.add(Conv2D(64, (3, 3), activation='relu'))#第二层卷积 64个（3,3）的卷
积核
cat_dog_model.add(BatchNormalization())
cat_dog_model.add(MaxPooling2D(pool_size=(2, 2)))
cat_dog_model.add(Dropout(0.25))

cat_dog_model.add(Conv2D(128, (3, 3), activation='relu'))#第三层卷积 128个（3,3）的
卷积核
cat_dog_model.add(BatchNormalization())
cat_dog_model.add(MaxPooling2D(pool_size=(2, 2)))
cat_dog_model.add(Dropout(0.25))#随机失活的概率为0.25

cat_dog_model.add(Flatten())#将二维化为列向量
cat_dog_model.add(Dense(256, activation='relu'))#犬类阶层 普通的神经网络结构
cat_dog_model.add(BatchNormalization())
cat_dog_model.add(Dropout(0.5))
cat_dog_model.add(Dense(1, activation='softmax')) # 最终输出0-1的数 <0.5判断为猫
>0.5判断为狗
```

```
cat_dog_model.compile(loss='categorical_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])#编译模型 评价指标为正确率

cat_dog_model.fit(train,label,batch_size=16,nb_epoch=10,shuffle=True)#训练模型
cat_dog_model.save("CATVSDOG.h5")#保存模型
```

```
)
Epoch 1/50
1333/1333 [==============================] - 1741s 1s/step - loss: 0.7494 - accuracy: 0.6291 - val_loss: 0.5394 - v
al_accuracy: 0.7305
Epoch 2/50
/root/anaconda3/envs/jupyter_notebook/lib/python3.7/site-packages/keras/callbacks/callbacks.py:1042: RuntimeWarning
: Reduce LR on plateau conditioned on metric `val_acc` which is not available. Available metrics are: val_loss,val_
accuracy,loss,accuracy,lr
  (self.monitor, ','.join(list(logs.keys()))), RuntimeWarning
 958/1333 [====================>.........] - ETA: 7:31 - loss: 0.5669 - accuracy: 0.7164
```

由于电脑性能受限。运行时间过于漫长

程序4：识别结果

```
from keras import backend as K
from keras.models import load_model
import tensorflow as tf
import numpy as np
SIZE=150

print("result")
for i in range(1,11):
    image_path="/root/cat/img/cat_dct/cat_dct"+str(i)+".jpg"#读取图片
    image = tf.keras.preprocessing.image.load_img(image_path, target_size=(SIZE,
SIZE))#将输入图像处理成数组
    image = tf.keras.preprocessing.image.img_to_array(image)
    image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
    image = np.expand_dims(image, axis=0)
    print(catmodel.predict(image))#打印输出值

     if (catmodel.predict(image)>0.5):#判断猫狗分类
         print("dog")
     else:
         print("cat")
```