实验一

加载预定义 VGG16 网络参数，并在 Kaggle 猫/狗数据集上进行微调和测试

1.加载 keras 模块

8

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Convolution2D, MaxPooling2D,ZeroPadding2D
from keras.layers import Activation, Dropout, Flatten, Dense, Input
from keras.utils import to_categorical
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input, VGG16
from keras import backend as K
import numpy as np
```

No output

加载预训练权值

定义新加入的 layers

9

```
img_width, img_height = 150, 150
if K.image_data_format() == 'channels_first':
    input_shape = (3, img_width, img_height)
else:
    input_shape = (img_width, img_height, 3)

# LOAD VGG16
input_tensor = Input(shape=input_shape)
model = VGG16(weights='imagenet',
                      include_top=False,
                      input_tensor=input_tensor)

top_model = Sequential()
top_model.add(Flatten(input_shape=model.output_shape[1:]))
top_model.add(Dense(256, activation='relu'))
top_model.add(Dropout(0.5))
top_model.add(Dense(2, activation='softmax'))
#top_model.load_weights(top_model_weights_path)
```

```
# CREATE AN "REAL" MODEL FROM VGG16
# BY COPYING ALL THE LAYERS OF VGG16
new_model = Sequential()
for l in model.layers:
    new_model.add(l)
new_model.add(top_model)
```

No output

设置不需微调的 layers 的 trainable 属性

并调用 compile 函数重新编译网络

10

```
from keras.optimizers import SGD
for layer in new_model.layers[:4]:
    layer.trainable = False
#compile the model with a SGD/momentum optimizer
# and a very slow learning rate.
new_model.compile(loss='categorical_crossentropy',
              optimizer=SGD(lr=1e-4, momentum=0.9),
              metrics=['accuracy'])
```

No output

定义 ImageDataGenerator

11

```
train_data_dir = r'C:\Users\coffe\Desktop\dogs-vs-cats\train'
validation_data_dir = r'C:\Users\coffe\Desktop\dogs-vs-cats\validation'
nb_train_samples = 10835
nb_validation_samples = 4000
epochs = 1
batch_size = 20


# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
```

```
# only rescaling
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')


Found 10835 images belonging to 2 classes.
Found 4000 images belonging to 2 classes.
```

训练模型

12

```
new_model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size)
Epoch 1/1
541/541 [==============================] - 253s 467ms/step - loss: 0.3200 -
accuracy: 0.8505 - val_loss: 0.1355 - val_accuracy: 0.9337
```

12

```
<keras.callbacks.callbacks.History at 0x1e0d921ff28>
```

使用训练后模型预测图像

16

```
import cv2
img = cv2.resize(cv2.imread(r'C:\Users\coffe\Desktop\dogs-vs-
cats\test\1.jpg'), (img_width, img_height)).astype(np.float32)
# img[:,:,0] -= 103.939
# img[:,:,1] -= 116.779
# img[:,:,2] -= 123.68
```

```
#img = img.transpose((2,0,1))
x = img_to_array(img)

x = np.expand_dims(x, axis=0)

#x = preprocess_input(x)

score = new_model.predict(x)


print(score)
[[0. 1.]]
```