# 实验二
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

42

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils import to_categorical
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras import backend as K
import numpy as np
```

No output

定义 CNN 网络结构

43

```
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)


model = Sequential()
model.add(ZeroPadding2D((1,1),input_shape=(3,224,224)))
model.add(Convolution2D(64, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(64, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

model.compile(loss='binary_crossentropy',
            optimizer='rmsprop',
            metrics=['accuracy'])
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\ops\nn_impl.py:180:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops)
is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

加载预训练权值

```
import h5py
f = h5py.File('models/vgg/vgg16_weights.h5')
```

```
for k in range(f.attrs['nb_layers']):
    if k >= len(model_vgg.layers) - 1:
        # we don't look at the last two layers in the savefile (fully-
connected and activation)
        break
    g = f['layer_{}'.format(k)]
    weights = [g['param_{}'.format(p)] for p in range(g.attrs['nb_params'])]
    layer = model_vgg.layers[k]

    if layer.__class__.__name__ in ['Convolution1D', 'Convolution2D',
'Convolution3D', 'AtrousConvolution2D']:
        weights[0] = np.transpose(weights[0], (2, 3, 1, 0))

    layer.set_weights(weights)

f.close()
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 148, 148, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 72, 72, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 34, 34, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_7 (Dense)              (None, 64)                1183808
_________________________________________________________________
activation_4 (Activation)    (None, 64)                0
_________________________________________________________________
```

```
dropout_1 (Dropout)         (None, 64)             0
_________________________________________________________________
dense_8 (Dense)             (None, 1)              65
_________________________________________________________________
activation_5 (Activation)    (None, 1)              0
=================================================================
Total params: 1,212,513
Trainable params: 1,212,513
Non-trainable params: 0
_________________________________________________________________
```

定义新加入的 layers

```
top_model = Sequential()
top_model.add(Flatten(input_shape=model_vgg.output_shape[1:]))
top_model.add(Dense(256, activation='relu'))
top_model.add(Dropout(0.5))
top_model.add(Dense(1, activation='sigmoid'))

top_model.load_weights('models/bottleneck_40_epochs.h5')

model_vgg.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

```
for layer in model_vgg.layers[:25]:
    layer.trainable = False
#compile the model with a SGD/momentum optimizer
# and a very slow learning rate.
model_vgg.compile(loss='binary_crossentropy',
              optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
              metrics=['accuracy'])
```

No output

定义 ImageDataGenerator

```
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='binary')

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

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

训练模型

```
model_vgg.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 [==============================] - 288s 533ms/step - loss: 0.6526 -
accuracy: 0.6134 - val_loss: 0.4863 - val_accuracy: 0.7115
```

46

```
<keras.callbacks.callbacks.History at 0x2b7dcbaa2e8>
```

使用训练后模型预测图像

48

```
import cv2
img = cv2.resize(cv2.imread(r'C:\Users\coffe\Desktop\dogs-vs-
cats\test\7.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 = model_vgg.predict(x)


print(score)
[[0.99895906]]
```