实验一

自定义深度卷积申请网络，并在 Kaggle 猫/狗数据集上进行训练和测试

1.加载 keras 模块

25

```
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

首先尝试：自定义深度 CNN 模型结构

然后尝试：定义 Alex/VGG 网络结构

4

```
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(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
            optimizer='rmsprop',
            metrics=['accuracy'])
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-
packages\keras\backend\tensorflow_backend.py:4070: The name tf.nn.max_pool
is deprecated. Please use tf.nn.max_pool2d instead.

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
```

查看 model 架构


```
model.summary()
Model: "sequential_2"
_________________________________________________________________
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_1 (Dense)             (None, 64)               1183808
_________________________________________________________________
activation_4 (Activation)    (None, 64)               0
_________________________________________________________________
dropout_1 (Dropout)         (None, 64)               0
_________________________________________________________________
dense_2 (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 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

9

```
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 [==============================] - 88s 163ms/step - loss: 0.5732 -
accuracy: 0.7074 - val_loss: 0.6979 - val_accuracy: 0.7228
```

9

```
<keras.callbacks.callbacks.History at 0x1f2a2772a58>
```

使用训练后模型预测图像

35

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

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

#x = preprocess_input(x)

score = model.predict(x)


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