dog vs cat

February 27, 2022

1 Libraries

[1]:
```
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from keras.preprocessing import image
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
```

2 Dataset

2.1 Preprocessing the Training set

[2]:
```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
training_set = train_datagen.flow_from_directory('dataset/training_set',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
```

```
Found 8000 images belonging to 2 classes.
```

[3]:
```
# target_size 150, 150 or 256, 256
```

2.2 Preprocessing the Test set

[4]:
```
test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = test_datagen.flow_from_directory('dataset/test_set',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

```
Found 2000 images belonging to 2 classes.
```

[5]:
```
# target size 150,150 or 256, 256 (same training set)
```

3 Building a Convolutional Neural Network (CNN)

[6]:
```
cnn = tf.keras.models.Sequential()
```

3.1 Convolution

[7]:
```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu',␣
 ↪input_shape=[64, 64, 3]))
```

[8]:
```
# 64,64 because we used earlier in processing in test and train dataset
# has to match
# for black images change the last digit 3 into 1
# kerner_size = 3 or 5 or 7
```

3.2 Pooling

[9]:
```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

3.3 Second Convolutional Layer

[10]:
```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

3.4 Flattening

[11]:
```
cnn.add(tf.keras.layers.Flatten())
```

3.5 Full Connection

[12]:
```
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
```

[13]:
```
# hidden neuron = 128
```

3.6 Output Layer

[14]:
```
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

4 Training CNN

4.1 Compiling the CNN

[15]:
```
cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =␣
 ↪['accuracy'])
```

4.2 Training the CNN on the Training set and evaluating it on the Test set

[ ]:

```
cnn.fit(x = training_set, validation_data = test_set, epochs = 25)
```

```
Epoch 1/25
 63/250 [======>…] - ETA: 2:06 - loss: 0.7072 - accuracy:
0.5188
```

5 Prediction

[ ]:

```
test_image = image.load_img('dataset/single_prediction/cat_or_dog_1.jpg',␣
 ↪target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = cnn.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
    prediction = 'dog'
else:
    prediction = 'cat'

print(prediction)

# if result[0][0] > 0.5:
#          prediction = ‘dog’
```

[ ]:

```
# train and set er sime same same hote hbe 64, 64
# result first 0 means batch
# scond 0 pic er index
```

[ ]:

```
img=mpimg.imread('dataset/single_prediction/cat_or_dog_1.jpg')
plt.figure(figsize=(20, 20))
plt.axis('off')
plt.imshow(img)
```