In [1]:
```
# Part 1 - Building the CNN

# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

# Initialising the CNN
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'rel
u'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding a second convolutional layer
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
 ['accuracy'])

# Part 2 - Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory('dataset/training_set',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')

test_set = test_datagen.flow_from_directory('dataset/test_set',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')

classifier.fit_generator(training_set,
```

```
                         steps_per_epoch = 8000,
                         epochs = 25,
                         validation_data = test_set,
                         validation_steps = 2000)
```

```
Using TensorFlow backend.
```

```
Found 8000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
Epoch 1/25
8000/8000 [==============================] - 974s 122ms/step - loss: 0.3659 -
acc: 0.8287 - val_loss: 0.5775 - val_acc: 0.7975
Epoch 2/25
8000/8000 [==============================] - 1053s 132ms/step - loss: 0.1202
- acc: 0.9536 - val_loss: 0.8787 - val_acc: 0.8105
Epoch 3/25
8000/8000 [==============================] - 991s 124ms/step - loss: 0.0604 -
acc: 0.9781 - val_loss: 1.0725 - val_acc: 0.8040
Epoch 4/25
8000/8000 [==============================] - 900s 113ms/step - loss: 0.0446 -
acc: 0.9844 - val_loss: 1.1117 - val_acc: 0.7980
Epoch 5/25
8000/8000 [==============================] - 873s 109ms/step - loss: 0.0348 -
acc: 0.9880 - val_loss: 1.1672 - val_acc: 0.8030
Epoch 6/25
8000/8000 [==============================] - 880s 110ms/step - loss: 0.0281 -
acc: 0.9906 - val_loss: 1.3560 - val_acc: 0.8070
Epoch 7/25
8000/8000 [==============================] - 912s 114ms/step - loss: 0.0253 -
acc: 0.9915 - val_loss: 1.3496 - val_acc: 0.8070
Epoch 8/25
8000/8000 [==============================] - 925s 116ms/step - loss: 0.0229 -
acc: 0.9924 - val_loss: 1.4182 - val_acc: 0.7975
Epoch 9/25
8000/8000 [==============================] - 932s 117ms/step - loss: 0.0188 -
acc: 0.9939 - val_loss: 1.4855 - val_acc: 0.8020
Epoch 10/25
8000/8000 [==============================] - 911s 114ms/step - loss: 0.0177 -
acc: 0.9943 - val_loss: 1.3892 - val_acc: 0.8035
Epoch 11/25
8000/8000 [==============================] - 887s 111ms/step - loss: 0.0158 -
acc: 0.9947 - val_loss: 1.5062 - val_acc: 0.8010
Epoch 12/25
8000/8000 [==============================] - 889s 111ms/step - loss: 0.0149 -
acc: 0.9953 - val_loss: 1.6563 - val_acc: 0.7985
Epoch 13/25
8000/8000 [==============================] - 914s 114ms/step - loss: 0.0148 -
acc: 0.9953 - val_loss: 1.5940 - val_acc: 0.7895
Epoch 14/25
8000/8000 [==============================] - 931s 116ms/step - loss: 0.0133 -
acc: 0.9957 - val_loss: 1.5452 - val_acc: 0.8075
Epoch 15/25
8000/8000 [==============================] - 933s 117ms/step - loss: 0.0114 -
acc: 0.9963 - val_loss: 1.6685 - val_acc: 0.8000
Epoch 16/25
8000/8000 [==============================] - 937s 117ms/step - loss: 0.0116 -
acc: 0.9962 - val_loss: 1.5643 - val_acc: 0.8100
Epoch 17/25
8000/8000 [==============================] - 934s 117ms/step - loss: 0.0115 -
acc: 0.9964 - val_loss: 1.4501 - val_acc: 0.8165
Epoch 18/25
8000/8000 [==============================] - 933s 117ms/step - loss: 0.0104 -
acc: 0.9968 - val_loss: 1.5362 - val_acc: 0.8020
Epoch 19/25
```

```
8000/8000 [==============================] - 946s 118ms/step - loss: 0.0103 -
acc: 0.9969 - val_loss: 1.6427 - val_acc: 0.7975
Epoch 20/25
8000/8000 [==============================] - 859s 107ms/step - loss: 0.0101 -
acc: 0.9970 - val_loss: 1.6678 - val_acc: 0.8065
Epoch 21/25
8000/8000 [==============================] - 858s 107ms/step - loss: 0.0094 -
acc: 0.9972 - val_loss: 1.5751 - val_acc: 0.8030
Epoch 22/25
8000/8000 [==============================] - 867s 108ms/step - loss: 0.0091 -
acc: 0.9973 - val_loss: 1.6835 - val_acc: 0.7990
Epoch 23/25
8000/8000 [==============================] - 848s 106ms/step - loss: 0.0080 -
acc: 0.9975 - val_loss: 1.6175 - val_acc: 0.7980
Epoch 24/25
8000/8000 [==============================] - 845s 106ms/step - loss: 0.0088 -
acc: 0.9975 - val_loss: 1.6234 - val_acc: 0.8080
Epoch 25/25
8000/8000 [==============================] - 861s 108ms/step - loss: 0.0084 -
acc: 0.9974 - val_loss: 1.7599 - val_acc: 0.7995
```

Out[1]: <keras.callbacks.History at 0x1f43d4ad7f0>

In [7]:
```
import numpy as np
from keras.preprocessing import image
test_image = image.load_img('dataset/single_prediction/test3.jpg', target_size
 = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = classifier.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
    prediction = 'dog'
else:
    prediction = 'cat'

print(prediction)
```

```
cat
```