In [1]:

```
#importing Necessary Libraries
from keras.preprocessing.image import ImageDataGenerator #generate Multiple data from a single pic
ture like rotating it
#shinking , zooming in, fliping Horizontaly or verticaly
from keras.models import Sequential
```

```
Using TensorFlow backend.
```

In [2]:

```
from keras.layers import Conv2D,MaxPooling2D #conv2d is to create convolutional layer where we can
extract features from the images
#Maxpolling for reducing the size of the picture
```

In [3]:

```
from keras.layers import Activation,Dropout,Flatten,Dense
#Droput used so the model doesnot overfit
#Flatten coverts 2d into 1d image which puts into neural networks (which takes only 1d array)
#Dense to create hidden layer or output layer
```

In [4]:

```
from keras.preprocessing import image #to import images and process them
```

In [5]:

```
#dimensions of our images
img_width ,img_height = 64,64
```

In [6]:

```
train_data_dir="C:/Users/Dell/Downloads/cat-and-dog/training_set/training_set"
test_data_dir="C:/Users/Dell/Downloads/cat-and-dog/test_set/test_set"
```

In [7]:

```
input_shape=(img_width,img_height,3)
```

In [8]:

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

In [9]:

```
test_datagen = ImageDataGenerator(rescale =1./255) #if you need you can mention others but here we
will keep it natural
```

In [10]:

```
train_generator= train_datagen.flow_from_directory(
                    train_data_dir,
                    target_size=(img_width,img_height),
                    batch_size= 32,
                    class_mode="binary")


# first it goes to the particular folder, get the data and then process it
```

```
Found 8005 images belonging to 2 classes.
```

In [11]:

```
test_generator=test_datagen.flow_from_directory(test_data_dir,target_size=(img_width,img_height),batch_size= 30,
                    class_mode="binary")
```

```
Found 2023 images belonging to 2 classes.
```

In [12]:

```
# we had import the libraries
#we had generated our data
# now we need to put the data into neural network
```

In [13]:

```
#making a neural network
```

In [14]:

```
model = Sequential()
model.add(Conv2D(32,(3,3),input_shape=(64,64,3),))# we extract 32 features by using (3,3) matrix
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
```

```
WARNING:tensorflow:From C:\Users\Dell\Anaconda3\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\Dell\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

In [15]:

```
model.add(Conv2D(64 ,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
```

In [16]:

```
model.add(Flatten()) #converts 2d into 1d
model.add(Dense(64)) # hidden layer and here 64 because earlier we got 64 inputs
model.add(Activation("relu"))

model.add(Dense(1)) #output layer
model.add(Activation("sigmoid"))
```

In [17]:

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
activation_1 (Activation)    (None, 62, 62, 32)        0
```

```
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 29, 29, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                802880
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_4 (Activation)    (None, 1)                 0
=================================================================
Total params: 822,337
Trainable params: 822,337
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

```
model.compile(loss= "binary_crossentropy", optimizer = "adam",metrics =["accuracy"])
```

In [19]:

```
#we have created neural network, now we need to insert our data
```

In [20]:

```
model.fit_generator(train_generator,steps_per_epoch=40,
                    epochs=20, validation_data =test_generator ,validation_steps=100)
```

```
WARNING:tensorflow:From C:\Users\Dell\Anaconda3\lib\site-
packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/20
40/40 [==============================] - 100s 2s/step - loss: 0.6969 - acc: 0.5211 - val_loss: 0.6
839 - val_acc: 0.6095
Epoch 2/20
40/40 [==============================] - 91s 2s/step - loss: 0.6863 - acc: 0.5500 - val_loss: 0.67
23 - val_acc: 0.6064
Epoch 3/20
40/40 [==============================] - 92s 2s/step - loss: 0.6537 - acc: 0.6297 - val_loss: 0.75
60 - val_acc: 0.5303
Epoch 4/20
40/40 [==============================] - 88s 2s/step - loss: 0.6477 - acc: 0.6187 - val_loss: 0.62
68 - val_acc: 0.6571
Epoch 5/20
40/40 [==============================] - 84s 2s/step - loss: 0.6194 - acc: 0.6680 - val_loss: 0.60
07 - val_acc: 0.6827
Epoch 6/20
40/40 [==============================] - 86s 2s/step - loss: 0.6177 - acc: 0.6672 - val_loss: 0.59
27 - val_acc: 0.6721
Epoch 7/20
40/40 [==============================] - 84s 2s/step - loss: 0.6099 - acc: 0.6590 - val_loss: 0.58
33 - val_acc: 0.6929
Epoch 8/20
40/40 [==============================] - 83s 2s/step - loss: 0.5922 - acc: 0.6891 - val_loss: 0.56
48 - val_acc: 0.7157
Epoch 9/20
40/40 [==============================] - 86s 2s/step - loss: 0.5778 - acc: 0.6945 - val_loss: 0.64
05 - val_acc: 0.6365
Epoch 10/20
```

```
40/40 [==============================] - 84s 2s/step - loss: 0.5974 - acc: 0.6867 - val_loss: 0.56
34 - val_acc: 0.7023
Epoch 11/20
40/40 [==============================] - 86s 2s/step - loss: 0.5835 - acc: 0.6891 - val_loss: 0.54
14 - val_acc: 0.7252
Epoch 12/20
40/40 [==============================] - 89s 2s/step - loss: 0.5731 - acc: 0.6930 - val_loss: 0.71
62 - val_acc: 0.6064
Epoch 13/20
40/40 [==============================] - 85s 2s/step - loss: 0.5779 - acc: 0.7059 - val_loss: 0.54
59 - val_acc: 0.7390
Epoch 14/20
40/40 [==============================] - 82s 2s/step - loss: 0.5403 - acc: 0.7258 - val_loss: 0.52
68 - val_acc: 0.7382
Epoch 15/20
40/40 [==============================] - 87s 2s/step - loss: 0.5486 - acc: 0.7188 - val_loss: 0.53
52 - val_acc: 0.7286
Epoch 16/20
40/40 [==============================] - 86s 2s/step - loss: 0.5722 - acc: 0.6969 - val_loss: 0.52
12 - val_acc: 0.7442
Epoch 17/20
40/40 [==============================] - 88s 2s/step - loss: 0.5323 - acc: 0.7242 - val_loss: 0.52
76 - val_acc: 0.7404
Epoch 18/20
40/40 [==============================] - 92s 2s/step - loss: 0.5625 - acc: 0.7016 - val_loss: 0.53
03 - val_acc: 0.7305
Epoch 19/20
40/40 [==============================] - 44s 1s/step - loss: 0.5266 - acc: 0.7413 - val_loss: 0.51
50 - val_acc: 0.7519
Epoch 20/20
40/40 [==============================] - 33s 827ms/step - loss: 0.5092 - acc: 0.7547 - val_loss: 0
.5072 - val_acc: 0.7529
```

Out[20]:

```
<keras.callbacks.History at 0x18ee2514c88>
```

In [22]:

```
# now we are saving our weights so that we can use this weight in future
```

In [23]:

```
model.save_weights("first_try.h5")
```

In [24]:

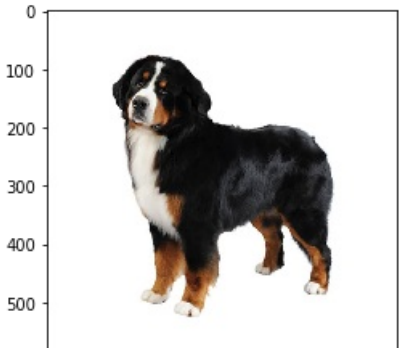
```
testing =image.load_img("C:/data/WORKING_Bernese-Mountain-Dog.jpg",target_size=(64,64))
```

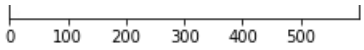
In [30]:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
plt.imshow(mpimg.imread("C:/data/WORKING_Bernese-Mountain-Dog.jpg"))
```

Out[30]:

```
<matplotlib.image.AxesImage at 0x18ee5a8d8d0>
```





In [26]:

```
testing=image.img_to_array(testing)
```

In [27]:

```
import numpy as np
```

In [28]:

```
img_pred = np.expand_dims(testing,axis=0)
```

In [29]:

```
result = model.predict(img_pred)
print(result)

if result[0][0]==1:
    prediction ="Dog"
else:
    prediction = "cat"

print(prediction)
```

```
[[1.]]
Dog
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: