In [1]:

```
from keras.models import Sequential
from keras.layers import Conv2D,Activation,MaxPooling2D,Dense,Flatten,Dropout
import numpy as np
```

```
Using TensorFlow backend.
```

# initializing a convolutional neural network using the sequential model of keras.

In [ ]:

In [2]:

```
classifier = Sequential()
```

In [3]:

```
classifier.add(Conv2D(32,(3,3),input_shape=(64,64,3)))
```

# Activation Layer

In [4]:

```
classifier.add(Activation('relu'))
```

### adding Pooling helps to reduce the dimensionality of each feature map and retains the essential information.

In [5]:

```
classifier.add(MaxPooling2D(pool_size =(2,2)))
```

In [ ]:

In [6]:

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

### adding drop out to avoid overfitting

In [7]:

```
classifier.add(Flatten())
```

In [8]:

```
classifier.add(Dense(64))
classifier.add(Activation('relu'))
```

In [9]:

```
classifier.add(Dropout(0.5))
```

### initializing 1 more fully connected layer.

### initializing 1 more fully connected layer

In [10]:
```
classifier.add(Dense(1))
```

### adding sigmoid so to convert data to probabilities

In [11]:
```
classifier.add(Activation('sigmoid'))
```

### viewing summary of how the classifier looks like

In [12]:
```
classifier.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
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
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 29, 29, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 32)        9248
_________________________________________________________________
activation_3 (Activation)    (None, 12, 12, 32)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                73792
_________________________________________________________________
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: 93,249
Trainable params: 93,249
Non-trainable params: 0
_________________________________________________________________
```

## compiling the model

In [13]:

```
classifier.compile(optimizer ='rmsprop',
                   loss ='binary_crossentropy',
                   metrics =['accuracy'])
```

In [ ]:

```

```

## Data augmentation

In [14]:

```
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)
```

In [ ]:

```

```

## setting train and test directories

In [15]:

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

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

```
Found 38 images belonging to 3 classes.
Found 20 images belonging to 3 classes.
```

## Training the classifier

In [16]:
```
from IPython.display import display
from PIL import Image
classifier.fit_generator(training_set,
                        steps_per_epoch =625,
                        epochs = 6,
                        validation_data =test_set,
                        validation_steps = 5000)
```

```
Epoch 1/6
625/625 [==============================] - 576s 922ms/step - loss: -474793622.8
512 - accuracy: 0.4994 - val_loss: -2675842560.0000 - val_accuracy: 0.5000
Epoch 2/6
625/625 [==============================] - 568s 909ms/step - loss: -1877960325
8.9033 - accuracy: 0.4998 - val_loss: -57959440384.0000 - val_accuracy: 0.5000
Epoch 3/6
625/625 [==============================] - 575s 919ms/step - loss: -16591919733
3.8167 - accuracy: 0.4999 - val_loss: -378296107008.0000 - val_accuracy: 0.5000
Epoch 4/6
625/625 [==============================] - 569s 910ms/step - loss: -76666381618
7.3406 - accuracy: 0.5001 - val_loss: -1475622207488.0000 - val_accuracy: 0.500
0
Epoch 5/6
625/625 [==============================] - 569s 910ms/step - loss: -24467907596
69.3628 - accuracy: 0.5000 - val_loss: -4237761970176.0000 - val_accuracy: 0.50
00
Epoch 6/6
625/625 [==============================] - 579s 926ms/step - loss: -62868027375
32.5234 - accuracy: 0.5000 - val_loss: -10177109557248.0000 - val_accuracy: 0.5
000
```

Out[16]:
```
<keras.callbacks.callbacks.History at 0x7f41b82095d0>
```

In [ ]:

In [17]:
```
from keras.models import Sequential
from keras.layers import Dense
```

In [ ]:

## saving the model

In [ ]:

In [18]:
```
classifier.save('catdog_cnn_model.h5')
```

## importing the model so we can test

In [19]:
```
from keras.models import load_model
classifier = load_model('catdog_cnn_model.h5')
```

# performing the test

In [22]:

```
import numpy as np
from keras.preprocessing import image
test_image =image.load_img('data/Train/dogs/11.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)
if result[0][0] >= 0.5:
    prediction = 'dog'
else:
    prediction = 'cat'
print(prediction)
```

```
dog
```

# our model has successfully classified our image

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: