Imports Used

- **Sequential** is used to create layer by layer Convolution Neural Network with Single input
- **Conv2D** is used to perform the first step => Convolution operation
- **MaxPooling2D** performs pooling operation. Here instead of using mean or min poling we used maxpooling as we have to give importance to the maximum weight pixel in the region of concern
- **Flatten** will convert the 2D array into 1D array
- **Dense** forms a fully conected neural network

In [1]:

```
# 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
```

```
/home/user/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion
of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it
will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

Creation and Compilation of Classifier

In the below, Sequential() creates a classifier to which the various layers are added. Following describes the layers and their input parameters.

1. **Convolution Operation**
   - The first parameter in Conv2D is 32 which indicates the number of filters used.
   - The second parameter (3,3) is the shape of filter
   - The third paraeter indicates the shape of input image size which is 64x64x3 , 3 here stands for RGB
   - The fourth parameter is the activation function. Here it is **"RELU"**
2. **Maxpooling Operation**
   The size of the pool is 2x2. Very small size is considered inorder to avoid pixel loss, so that we get precise location of where the feature is located.
3. **Flatten Operation**
   This will convert 2D into 1D array so that it could be used in fully connected neural network. i.e., the next layer
4. **Fully Connected Layer**
   In this the flattened output of the previous layer is given as the input. This layer can be considered as hidden layer as it lies inbetween the falttened inout and the ouput layer.
   - The first parameter units indicates the number of hidden nodes that are present in this layer.
   - The second parameter indicates the activation function used. In this case it is **"RELU"**
5. **Output Layer**
   This layer contains only one node, as it is a binary classifier we get to know if the image contains CAT or DOG. Sigmoid activation function is used here as it can easily reduce the output to 0 or 1.

After adding all the required layers we compile them in the end.

- Optimizer parameter "adam" indincates Stochoistic Gradient Descent Algorithm
- Loss function choosen is Binary Cross Entropy
- "Accuracy" is the performance metrics chosen

In [2]:

```
classifier = Sequential()
#Add Convolution layer
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))
#Add Max Pooling layer
classifier.add(MaxPooling2D(pool_size = (2, 2)))
#Perform Flattening
classifier.add(Flatten())
#Add Fully Connected Network
classifier.add(Dense(units = 128, activation = 'relu'))
#Final Output layer
classifier.add(Dense(units = 1, activation = 'sigmoid'))
#compile the classsifier
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

In [3]:

```
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('training_set',target_size = (64, 64),batch_size = 32,class_mode = 'binary')
validation_set = train_datagen.flow_from_directory('validation_set',target_size = (64, 64),batch_size = 32,class_mode = 'binary')
test_set = test_datagen.flow_from_directory('test_set',target_size = (64, 64),batch_size = 32,class_mode = 'binary')
```

```
Found 1600 images belonging to 2 classes.
Found 400 images belonging to 2 classes.
Found 2023 images belonging to 2 classes.
```

In [4]:

```
classifier.fit_generator(training_set,steps_per_epoch = 1600,epochs = 10,validation_data = validation_set,validation_steps = 400,max_queue_size=7,workers=3)
```

```
Epoch 1/10
1600/1600 [==============================] - 400s 250ms/step - loss: 0.5095 - acc: 0.7480 - val_loss: 0.5724 - val_acc: 0.7315
Epoch 2/10
1600/1600 [==============================] - 448s 280ms/step - loss: 0.3110 - acc: 0.8657 - val_loss: 0.6626 - val_acc: 0.7395
Epoch 3/10
1600/1600 [==============================] - 440s 275ms/step - loss: 0.1608 - acc: 0.9390 - val_loss: 0.8654 - val_acc: 0.7289
Epoch 4/10
1600/1600 [==============================] - 450s 282ms/step - loss: 0.0809 - acc: 0.9726 - val_loss: 1.1711 - val_acc: 0.7039
Epoch 5/10
1600/1600 [==============================] - 443s 277ms/step - loss: 0.0530 - acc: 0.9828 - val_loss: 1.4644 - val_acc: 0.7079
Epoch 6/10
1600/1600 [==============================] - 446s 279ms/step - loss: 0.0428 - acc: 0.9855 - val_loss: 1.5372 - val_acc: 0.7166
Epoch 7/10
1600/1600 [==============================] - 456s 285ms/step - loss: 0.0326 - acc: 0.9895 - val_loss: 1.5508 - val_acc: 0.7284
Epoch 8/10
1600/1600 [==============================] - 466s 291ms/step - loss: 0.0258 - acc: 0.9913 - val_loss: 1.7487 - val_acc: 0.7165
Epoch 9/10
1600/1600 [==============================] - 434s 271ms/step - loss: 0.0252 - acc: 0.9915 - val_loss: 1.8884 - val_acc: 0.7113
Epoch 10/10
1600/1600 [==============================] - 441s 276ms/step - loss: 0.0261 - acc: 0.9916 - val_loss: 1.7523 - val_acc: 0.7186
```

Out[4]:

```
<keras.callbacks.History at 0x7fcacc410668>
```

Testing the model

In [23]:

```
import numpy as np
from keras.preprocessing import image
import os
import glob
data_path = os.path.join('./test_set/dogs/','*jpg')
files = glob.glob(data_path)
dog,cat,total=0,0,0
for f in files:
    test_image = image.load_img(f, 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:
        dog+=1
    else:
        cat+=1
    total+=1
print("Total Dogs: ",total)
print("Dogs : ",dog)
print("Cats : ",cat)
print('----------------------')

data_path = os.path.join('./test_set/cats/','*jpg')
files = glob.glob(data_path)
dog,cat,total=0,0,0
for f in files:
    test_image = image.load_img(f, 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:
        dog+=1
    else:
        cat+=1
    total+=1
print("Total Cats: ",total)
print("Dogs : ",dog)
print("Cats : ",cat)
print('----------------------')
```

```
Total Dogs:  1012
Dogs :  856
Cats :  156
----------------------
Total Cats:  1011
Dogs :  559
Cats :  452
----------------------
```