In [1]:

```
import tensorflow as tf
import keras
```

In [2]:

```
from keras.preprocessing.image import ImageDataGenerator
```

In [3]:

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

In [4]:

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

```
Found 8048 images belonging to 2 classes.
```

In [5]:

```
training_set.class_indices
```

Out[5]:

```
{'cats': 0, 'dogs': 1}
```

In [6]:

```
test_datagen = ImageDataGenerator(rescale = 1/255)

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

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

Modelling - Convolution Neural Network

Initialising the CNN

In [7]:

```
from keras.models import Sequential
classifier = Sequential()
```

Step 1 - Convolution

In [8]:

```
from keras.layers import Conv2D
classifier.add(Conv2D(input_shape=[64, 64, 3],
                      filters=32, kernel_size=3,activation='relu'))
```

Step 2 - Max Pooling

In [9]:

```
from keras.layers import MaxPooling2D
classifier.add(MaxPooling2D(pool_size=2, strides=2))
```

Step 3 - Flattening

In [10]:

```
from keras.layers import Flatten
classifier.add(Flatten())
```

Step 4 - Full Connection

In [11]:

```
from keras.layers import Dense

# hidden layer with 128 neurons
classifier.add(Dense(units = 128, activation = 'relu'))

# Output Layer with 1 neuron
classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

Training the CNN Model with train data & Testing the model with test data

In [12]:

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

In [13]:

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

```
Epoch 20/25
252/252 [==============================] - 49s 194ms/step - loss: 0.2559 - accuracy: 0.8914 - val_loss: 0.6285 - val_accu
racy: 0.7400
Epoch 21/25
252/252 [==============================] - 46s 181ms/step - loss: 0.2442 - accuracy: 0.8990 - val_loss: 0.6706 - val_accu
racy: 0.7495
Epoch 22/25
252/252 [==============================] - 49s 196ms/step - loss: 0.2252 - accuracy: 0.9062 - val_loss: 0.7430 - val_accu
racy: 0.7345
Epoch 23/25
252/252 [==============================] - 51s 201ms/step - loss: 0.2089 - accuracy: 0.9172 - val_loss: 0.8784 - val_accu
racy: 0.7120
Epoch 24/25
252/252 [==============================] - 46s 182ms/step - loss: 0.1988 - accuracy: 0.9200 - val_loss: 0.8082 - val_accu
racy: 0.7360
Epoch 25/25
252/252 [==============================] - 49s 196ms/step - loss: 0.1885 - accuracy: 0.9277 - val_loss: 0.8695 - val_accu
racy: 0.7340
```

Out[13]:

Evalution

- **Making a single prediction**

In [14]:

```
import numpy as np
from PIL import Image
```

In [15]:

```
#load the data
test_image = Image.open("dataset/single_prediction/cat_or_dog_1.jpg")

# Data Preprocessing
test_image = test_image.resize((64,64))
test_image = np.array(test_image)
test_image = np.expand_dims(test_image,axis=0)

# Prediction
result= classifier.predict(test_image)

# Evaluation
if result[0][0] == 1:
    print("Dog")
else:
    print("Cat")
```

```
1/1 [==============================] - 1s 801ms/step
Dog
```