In [1]:
```
import zipfile
zip_ref = zipfile.ZipFile('/content/catvsdog.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

In [2]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization, Flatten, Der
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense, Dropout
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report,accuracy_score
```

In [3]:
```
img_size=(64,64)
img_size_ch=(64,64,3)
train_datagen = ImageDataGenerator(rescale=1./255)
train_data = train_datagen.flow_from_directory(
    '/content/catvsdog/train',
    target_size=img_size,
    batch_size=32,
    class_mode='binary',
    shuffle=True
)
test_datagen = ImageDataGenerator(rescale=1./255)
test_data = test_datagen.flow_from_directory(
    '/content/catvsdog/test',
    target_size=img_size,
    batch_size=32,
    class_mode='binary',
    shuffle=False
)

print(train_data)
print(test_data)
# print(valid_data)
```

```
Found 300 images belonging to 2 classes.
Found 300 images belonging to 2 classes.
<keras.preprocessing.image.DirectoryIterator object at 0x7f4e9b817580>
<keras.preprocessing.image.DirectoryIterator object at 0x7f4f24617400>
```

In [4]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=img_size_ch))
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

```
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 12, 12, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 128)        0
 2D)

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 256)               1179904

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 1,273,409
Trainable params: 1,273,409
Non-trainable params: 0
_________________________________________________________________
```

In [5]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [6]:
```
model.fit(train_data, epochs=10, validation_data=test_data)
```

```
Epoch 1/10
10/10 [==============================] - 8s 616ms/step - loss: 0.7108 - accuracy: 0.5133 -
val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 2/10
10/10 [==============================] - 5s 551ms/step - loss: 0.6952 - accuracy: 0.4833 -
val_loss: 0.6918 - val_accuracy: 0.5000
Epoch 3/10
10/10 [==============================] - 4s 406ms/step - loss: 0.6921 - accuracy: 0.5267 -
val_loss: 0.6914 - val_accuracy: 0.6100
Epoch 4/10
10/10 [==============================] - 6s 658ms/step - loss: 0.6872 - accuracy: 0.5400 -
val_loss: 0.6845 - val_accuracy: 0.6100
Epoch 5/10
10/10 [==============================] - 7s 742ms/step - loss: 0.6798 - accuracy: 0.5633 -
val_loss: 0.6889 - val_accuracy: 0.5133
Epoch 6/10
10/10 [==============================] - 5s 539ms/step - loss: 0.6684 - accuracy: 0.6367 -
val_loss: 0.6640 - val_accuracy: 0.6200
Epoch 7/10
10/10 [==============================] - 6s 609ms/step - loss: 0.5956 - accuracy: 0.7100 -
val_loss: 0.7028 - val_accuracy: 0.6067
Epoch 8/10
10/10 [==============================] - 6s 678ms/step - loss: 0.5761 - accuracy: 0.6967 -
val_loss: 0.6741 - val_accuracy: 0.6200
Epoch 9/10
10/10 [==============================] - 4s 407ms/step - loss: 0.5041 - accuracy: 0.7667 -
val_loss: 0.6903 - val_accuracy: 0.5900
Epoch 10/10
10/10 [==============================] - 5s 515ms/step - loss: 0.4549 - accuracy: 0.7833 -
val_loss: 0.6939 - val_accuracy: 0.5967
```

Out[6]:
```
<keras.callbacks.History at 0x7f4e98608430>
```

In [7]:
```
# Calculate the accuracy
training_loss, training_accuracy = model.evaluate(train_data, verbose=2)
print("Training Accuracy = %.2f %% loss = %f" % (training_accuracy * 100, training_loss))
testing_loss, testing_accuracy = model.evaluate(test_data, verbose=2)
print("Testing Accuracy = %.2f %% loss = %f" % (testing_accuracy * 100, testing_loss))
```

```
10/10 - 2s - loss: 0.3982 - accuracy: 0.8433 - 2s/epoch - 212ms/step
Training Accuracy = 84.33 % loss = 0.398155
10/10 - 1s - loss: 0.6939 - accuracy: 0.5967 - 1s/epoch - 134ms/step
Testing Accuracy = 59.67 % loss = 0.693882
```

In [8]:
```
y_pred = np.round(model.predict(test_data))
```

```
10/10 [==============================] - 1s 123ms/step
```

In [9]:
```
print(classification_report(test_data.labels, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.64      0.45      0.53       150
           1       0.58      0.74      0.65       150

    accuracy                           0.60       300
   macro avg       0.61      0.60      0.59       300
weighted avg       0.61      0.60      0.59       300
```