The task is to build a model to predict the category of an animal: dog or cat?

In [2]:
```
# 4 steps are required to build a CNN:
#1.Convolution,
#2.Max pooling,
#3.Flattening, and
#4.Full connection


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

In [3]:
```
classifier = Sequential()
# Convolution is a linear operation involving the multiplication of weights with
# The multiplication is performed between an array of input data and a 2D array o
# It is represented like ana array of 0 and 1
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))

#The pooling operation provides spatial variance making the system capable of rec
# pooling basically helps reduce the number of parameters and computations preser
classifier.add(MaxPooling2D(pool_size = (2, 2)))

#The output from the final Pooling layer which is flattened is the input of the f
classifier.add(Flatten())
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

In [4]:
```
# Compile the CNN by choosing an SGD algorithm, a loss function, and performance
# We use binary_crossentropy for binary classification, and use categorical_cross

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

In [5]:
```
# Image augmentation is a method of applying different kinds of transformation to
# The images are different from each other in certain aspects because of shifting
# So, we are using the Keras ImageDataGenerator class to augment our images.
#create an object of ImageDataGenerator, for augmenting train set
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)


##create another object of ImageDataGenerator, for augmenting test set
test_datagen = ImageDataGenerator(rescale = 1./255)
```

In [ ]:
```
# we will use flow_from_directory(directory) method from Keras Official website t
# This is why we structured the data folders in a specific way so that the class
```

In [6]:
```
#apply image augmentation on train set by resizing all images to 64x64 and creati
training_set = train_datagen.flow_from_directory(r'C:\Users\M.komala\OneDrive\Des
                                                 target_size = (64, 64),
                                                 batch_size = 30,
                                                 class_mode = 'binary')
```

```
Found 8000 images belonging to 2 classes.
```

In [7]:
```
#apply image augmentation on test set by resizing all images to 64x64 and creatir
test_set = test_datagen.flow_from_directory(r'C:\Users\M.komala\OneDrive\Desktop\
                                            target_size = (64, 64),
                                            batch_size = 30,
                                            class_mode = 'binary')
```

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

In [12]:
```
classifier.fit_generator(training_set,
                         steps_per_epoch = (8000/32),
                         epochs = 25,
                         validation_data = test_set,
                         validation_steps = (2000/32))
```

```
C:\Users\M.komala\AppData\Local\Temp\ipykernel_9924\3171078237.py:1: UserWarnin
g: `Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  classifier.fit_generator(training_set,

Epoch 1/25
250/250 [==============================] - 626s 2s/step - loss: 0.6608 - accura
cy: 0.6036 - val_loss: 0.7445 - val_accuracy: 0.5519
Epoch 2/25
250/250 [==============================] - 338s 1s/step - loss: 0.5988 - accura
cy: 0.6781 - val_loss: 0.5671 - val_accuracy: 0.7159
Epoch 3/25
250/250 [==============================] - 227s 910ms/step - loss: 0.5724 - acc
uracy: 0.7012 - val_loss: 0.5456 - val_accuracy: 0.7376
Epoch 4/25
250/250 [==============================] - 170s 678ms/step - loss: 0.5490 - acc
uracy: 0.7156 - val_loss: 0.5535 - val_accuracy: 0.7302
Epoch 5/25
250/250 [==============================] - 55s 220ms/step - loss: 0.5453 - accu
racy: 0.7187 - val_loss: 0.5311 - val_accuracy: 0.7519
Epoch 6/25
250/250 [==============================] - 43s 173ms/step - loss: 0.5317 - accu
racy: 0.7311 - val_loss: 0.5499 - val_accuracy: 0.7265
Epoch 7/25
250/250 [==============================] - 44s 174ms/step - loss: 0.5282 - accu
racy: 0.7323 - val_loss: 0.5257 - val_accuracy: 0.7460
Epoch 8/25
250/250 [==============================] - 122s 490ms/step - loss: 0.5114 - acc
uracy: 0.7418 - val_loss: 0.5525 - val_accuracy: 0.7370
Epoch 9/25
250/250 [==============================] - 176s 703ms/step - loss: 0.4993 - acc
uracy: 0.7503 - val_loss: 0.5439 - val_accuracy: 0.7429
Epoch 10/25
250/250 [==============================] - 50s 200ms/step - loss: 0.4884 - accu
racy: 0.7613 - val_loss: 0.5401 - val_accuracy: 0.7339
Epoch 11/25
250/250 [==============================] - 45s 178ms/step - loss: 0.4880 - accu
racy: 0.7611 - val_loss: 0.5189 - val_accuracy: 0.7497
Epoch 12/25
250/250 [==============================] - 43s 173ms/step - loss: 0.4837 - accu
racy: 0.7665 - val_loss: 0.5193 - val_accuracy: 0.7513
Epoch 13/25
250/250 [==============================] - 43s 173ms/step - loss: 0.4676 - accu
racy: 0.7757 - val_loss: 0.6363 - val_accuracy: 0.6804
Epoch 14/25
250/250 [==============================] - 42s 168ms/step - loss: 0.4712 - accu
racy: 0.7744 - val_loss: 0.5953 - val_accuracy: 0.7095
Epoch 15/25
250/250 [==============================] - 43s 171ms/step - loss: 0.4575 - accu
racy: 0.7826 - val_loss: 0.5240 - val_accuracy: 0.7508
```

```
Epoch 16/25
250/250 [==============================] - 43s 174ms/step - loss: 0.4515 - accu
racy: 0.7809 - val_loss: 0.5183 - val_accuracy: 0.7640
Epoch 17/25
250/250 [==============================] - 43s 170ms/step - loss: 0.4494 - accu
racy: 0.7826 - val_loss: 0.5390 - val_accuracy: 0.7455
Epoch 18/25
250/250 [==============================] - 46s 186ms/step - loss: 0.4371 - accu
racy: 0.7931 - val_loss: 0.5479 - val_accuracy: 0.7376
Epoch 19/25
250/250 [==============================] - 43s 170ms/step - loss: 0.4346 - accu
racy: 0.7937 - val_loss: 0.5461 - val_accuracy: 0.7476
Epoch 20/25
250/250 [==============================] - 44s 174ms/step - loss: 0.4358 - accu
racy: 0.7933 - val_loss: 0.5381 - val_accuracy: 0.7471
Epoch 21/25
250/250 [==============================] - 44s 174ms/step - loss: 0.4177 - accu
racy: 0.8029 - val_loss: 0.5336 - val_accuracy: 0.7619
Epoch 22/25
250/250 [==============================] - 43s 172ms/step - loss: 0.4177 - accu
racy: 0.8040 - val_loss: 0.5137 - val_accuracy: 0.7640
Epoch 23/25
250/250 [==============================] - 43s 170ms/step - loss: 0.4083 - accu
racy: 0.8147 - val_loss: 0.5379 - val_accuracy: 0.7603
Epoch 24/25
250/250 [==============================] - 42s 168ms/step - loss: 0.4039 - accu
racy: 0.8123 - val_loss: 0.5353 - val_accuracy: 0.7646
Epoch 25/25
250/250 [==============================] - 43s 171ms/step - loss: 0.3999 - accu
racy: 0.8160 - val_loss: 0.5255 - val_accuracy: 0.7630
```

Out[12]: <keras.callbacks.History at 0x1a6736029b0>

In [13]:
```
import numpy as np
```

In [ ]:
```
# Predict the image 1
```

In [19]:
```
from keras.preprocessing import image
```

In [25]:
```
import tensorflow as tf
```

In [28]:
```
import keras
```

In [55]:
```
test_image =  keras.utils.load_img(r'C:\Users\M.komala\OneDrive\Desktop\keras\CNN
                                   target_size = (64, 64))
```

In [56]:
```
test_image
```

Out[56]:

In [40]:
```
from keras.utils import np_utils
```

In [57]:
```
#add channel dimension for image
test_image = tf.keras.utils.img_to_array(test_image)
```

In [59]:
```
##add batch dimension for image
test_image = np.expand_dims(test_image, axis = 0)
```

In [60]:
```
result = classifier.predict(test_image)
```

```
1/1 [==============================] - 0s 203ms/step
```

In [61]:
```
result
```

Out[61]:
```
array([[1.]], dtype=float32)
```

In [64]:
```
if result [0][0] ==0:
    print('It is a Cat Image')
else:
    print('It is a Dog Image')
```

```
It is a Dog Image
```

In [63]:
```
training_set.class_indices

# when the output is '0' it is cat
# when the output is '1' it is Dog
```

Out[63]:
```
{'cats': 0, 'dogs': 1}
```

In [ ]:
```
# Predict 2nd Image
```

In [71]:
```
test_image1 =  keras.utils.load_img(r'C:\Users\M.komala\OneDrive\Desktop\keras\CN
                                    target_size = (64, 64))

test_image1 = tf.keras.utils.img_to_array(test_image1)
```

In [74]:
```
##add batch dimension for image
test_image1 = np.expand_dims(test_image1, axis = 0)
```

In [76]:
```
result = classifier.predict(test_image1)
```

```
1/1 [==============================] - 0s 219ms/step
```

In [77]:
```
result
```

Out[77]:
```
array([[0.]], dtype=float32)
```

In [78]:
```
if result [0][0] ==0:
    print('It is a Cat Image')
else:
    print('It is a Dog Image')
```

```
It is a Cat Image
```