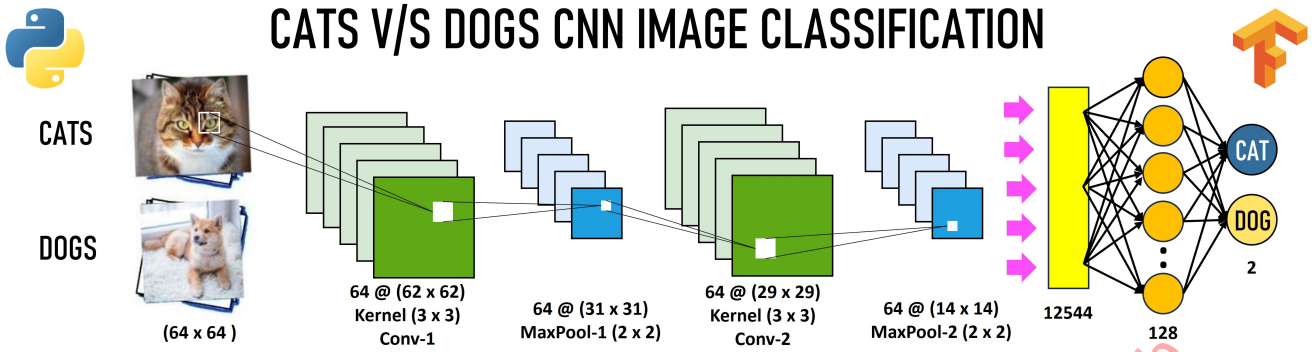

In [1]:
```
# Import required libraries
import os
import pickle
import cv2 as cv
import numpy as np
import tensorflow as tf
from tensorflow import keras
```

In [2]:
```
# Import dataset
DIR = r'C:\Users\Rav\Desktop\CNN-CATDOG-TF\dataset'
CATEGORY = ['cat', 'dog']
```

In [3]:
```
# Labeling of the data i.e. cat or dog. (Take some time, dataset is large)
IMG_SIZE = 64

data = []

for category in CATEGORY:
    folder = os.path.join(DIR, category)
    label = CATEGORY.index(category)
    for img in os.listdir(folder):
        img_path = os.path.join(folder, img)
        img_arr = cv.imread(img_path)
        img_arr = cv.resize(img_arr, (IMG_SIZE, IMG_SIZE))
        data.append([img_arr, label])
```

In [4]:
```
# Check length data
print(len(data))
```

```
25000
```

In [5]:
```
# Creating X and y for features and labels

X = []
y = []

for features, label in data:
    X.append(features)
    y.append(label)
```

In [6]:
```
# Convert X and y into array
X = np.array(X)
y = np.array(y)
```

In [7]:
```
# Now save data into a file
pickle.dump(X, open('X.pkl', 'wb')) #.pkl = extension
pickle.dump(y, open('y.pkl', 'wb')) # wb = write in binary
```

In [8]:
```
# load saved data
X = pickle.load(open('X.pkl', 'rb')) # rb = read in binary
y = pickle.load(open('y.pkl', 'rb'))
```

In [9]:
```
# Feature scaling technique
X = X / 255
```

In [10]:
```
print(X.shape)
```

```
(25000, 64, 64, 3)
```

In [11]:
```
# Design CNN Model
def modelCNN():
    model = keras.Sequential([# Input layer
                              keras.layers.Input(shape=(64, 64, 3)),

                              # Convolutional layer 1
                              keras.layers.Conv2D(64, (3, 3), activation='relu'),
                              keras.layers.MaxPooling2D(2, 2),

                              # Convolutional layer 2
                              keras.layers.Conv2D(64, (3, 3), activation='relu'),
                              keras.layers.MaxPooling2D(2, 2),

                              # Flatten layer - Output of convolution
                              keras.layers.Flatten(),

                              # Hidden layer 1
                              keras.layers.Dense(128, activation='relu'),

                              # Output layer
                              keras.layers.Dense(2, activation='softmax')],
                             name="CNN-CATDOG")
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['ac
    return model
```

In [12]:
```
model = modelCNN()
model.summary()
```

```
Model: "CNN-CATDOG"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 64)        1792

 max_pooling2d (MaxPooling2  (None, 31, 31, 64)        0
 D)

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        36928

 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 64)        0
 g2D)

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 128)               1605760

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 1644738 (6.27 MB)
Trainable params: 1644738 (6.27 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [13]:
```
history = model.fit(X, y, epochs=10, shuffle=True)
```

```
Epoch 1/10
782/782 [==============================] - 134s 170ms/step - loss: 0.6253 - accuracy: 0.
6489
Epoch 2/10
782/782 [==============================] - 129s 165ms/step - loss: 0.5003 - accuracy: 0.
7557
Epoch 3/10
782/782 [==============================] - 129s 165ms/step - loss: 0.4348 - accuracy: 0.
7942
Epoch 4/10
782/782 [==============================] - 129s 166ms/step - loss: 0.3677 - accuracy: 0.
8329
Epoch 5/10
782/782 [==============================] - 129s 166ms/step - loss: 0.3020 - accuracy: 0.
8670
Epoch 6/10
782/782 [==============================] - 131s 167ms/step - loss: 0.2235 - accuracy: 0.
9066
Epoch 7/10
782/782 [==============================] - 129s 165ms/step - loss: 0.1405 - accuracy: 0.
9454
Epoch 8/10
782/782 [==============================] - 131s 167ms/step - loss: 0.0827 - accuracy: 0.
9706
Epoch 9/10
782/782 [==============================] - 133s 169ms/step - loss: 0.0563 - accuracy: 0.
9802
Epoch 10/10
782/782 [==============================] - 129s 165ms/step - loss: 0.0366 - accuracy: 0.
9876
```

In [14]:
```
loss, acc = model.evaluate(X, y)
print("Loss : " + str(loss*100))
print("Accuracy : " + str(acc*100))
```

```
782/782 [==============================] - 34s 44ms/step - loss: 0.0211 - accuracy: 0.99
39
Loss : 2.106366492807865
Accuracy : 99.38799738883972
```

In [15]:
```
# Save a CNN model with extension .keras
model.save('catdog.keras')
```

In [16]:
```
# Predictions on images, taken from internet
CATEGORY = ['Cat', 'Dog']

def image(path):
    img = cv.imread(path)
    new_arr = cv.resize(img, (64, 64))
    new_arr = np.array(new_arr)
    new_arr = new_arr.reshape(1, 64, 64, 3)
    return new_arr

model = tf.keras.models.load_model('catdog.keras')
```

In [17]:
```
prediction = model.predict([image('sample\cat1.jpg')])
print(CATEGORY[prediction.argmax()])
```

```
1/1 [==============================] - 0s 161ms/step
Cat
```

In [18]:
```
prediction = model.predict([image('sample\cat2.jpg')])
print(CATEGORY[prediction.argmax()])
```

```
1/1 [==============================] - 0s 31ms/step
Cat
```

In [19]:
```
prediction = model.predict([image('sample\dog1.jpg')])
print(CATEGORY[prediction.argmax()])
```

```
1/1 [==============================] - 0s 31ms/step
Dog
```

In [20]:
```
prediction = model.predict([image('sample\dog2.jpg')])
print(CATEGORY[prediction.argmax()])
```

```
1/1 [==============================] - 0s 31ms/step
Dog
```