Khalid Dhabbah

For this increment, I was diving deeply into deep learning in order to understand how these models work starting from training them until we use them. This is because I could not know how to deploy them, so I decided to start from the beginning. The model I took to study is the CNN model.

**Summary**

First of all, to train the CNN model, a Convolutional Neural Network should be involved in order to do some significant steps, which are embedding, convolution, pooling, flattening, full connection. Also, loss and accuracy are playing a significant role. The loss is the number of errors in prediction, so it should be decreased, and the accuracy should be increased.

The big picture is that when we pass a photo, for example, it is going pass through all these layers in order to get the most significant factors in this photo, and get it in one single flattening data, and using the pooling layer to resize the picture. Then it depends on the model for what it trained for, we could label them. For instance, after training the model using a cat and dog pictures, this model is going to be for those two categories.

As a result, to get the labels, we should define them. For example, if the output is 0, it means this picture is for a cat, and if the output is 1, it means it is for a dog.

Here is how to get the output from a pre-trained CNN model we received using Jupyter Notebook.

**First, Loading the model:**

#Tensorflow and leras should be imported

import tensorflow as tf

from keras.models import load\_model

model = load\_model('CNN011019-223859\_model.h5')

**Second, there are many functions we can use to get information about the model**, such as model.summary(): to get all the layers.

**Third, save a json file for this model for future developments.**

# save as JSON

json\_string = model.to\_json()

with open('CNN011019-223859\_model.json', 'w') as file:

file.write(json\_string)

**Fourth, compiling the model**

model.compile(loss='binary\_crossentropy',

optimizer='Adam',

metrics=['accuracy'])

**Fifth, passing a picture through the layers**

#pip3 install opencv-contrib-python --user

import cv2

import numpy as np

img = cv2.imread('1.jpg')

print(img.shape)

img = cv2.resize(img,(64,64))

print(img.shape)

img = np.reshape(img,[1,64,64,3])

**Finally, Getting the output**

classes = model.predict\_classes(img)

print(\*classes)