YOLOv3 to TensorFlow Lite Conversion

In the previous article, we created a YOLOv3 custom object detection model with Transfer Learning. Let's now go a step ahead and convert it into a TensorFlow Lite model. So, let's begin.

Step 1: Gathering YOLOv3 model files.

A YOLOv3 trained setup typically consists of the following files –

a) **classes.txt**: Labels of the model.
b) ***.weights:** Stores the weights of the neural network.
c) ***.cfg:** YOLOv3 configuration file.

During the model training, the network weights are saved after every 1000 iterations so that if in case the training gets interrupted for some reason, it can be continued from where it was left last time to save a lot of computations and time, of course.

In the figure below, there are three ***.weights** files.

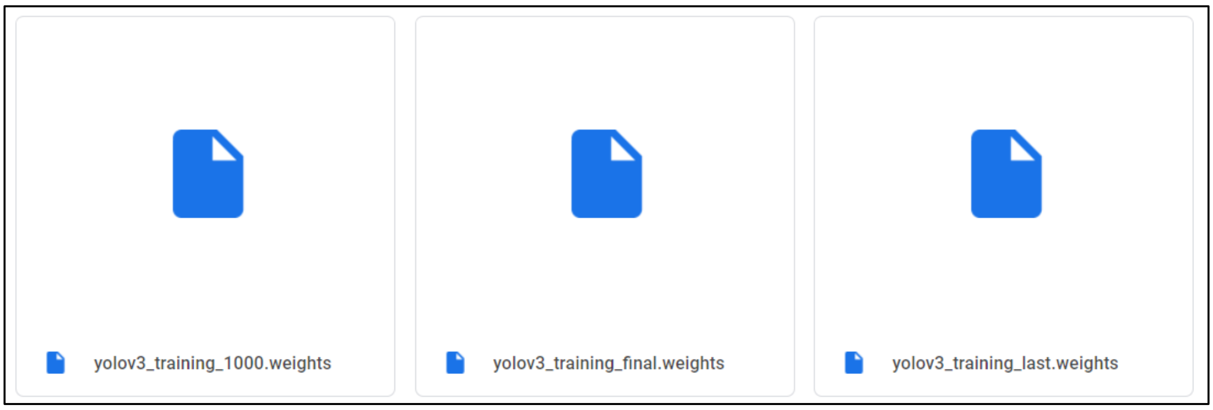

yolov3_training_1000.weights
yolov3_training_final.weights
yolov3_training_last.weights

Fig-1. YOLOv3 *.weights files

The **yolov3_training_1000.weights** file corresponds to the weights of first 1000 iterations, **yolov3_training_final.weights** file corresponds to the final weights generated after the training was completed and **yolov3_training_last.weights** file corresponds to the last saved weights just before the training was interrupted.

Note: Depending upon the size of your model, the number of ***.weights** file generated would vary. When the training is completed successfully, the last saved weights and the final weights of the model would be same. In other words, **yolov3_training_final.weights** and **yolov3_training_last.weights** are exactly the same at the end of the model training.

So, we'll need two files – **classes.txt** and **yolov3_training_last.weights.** Let's put these files inside a folder and name the folder as **"YOLOv3_TFLite"** for the sake of convenience during conversion.

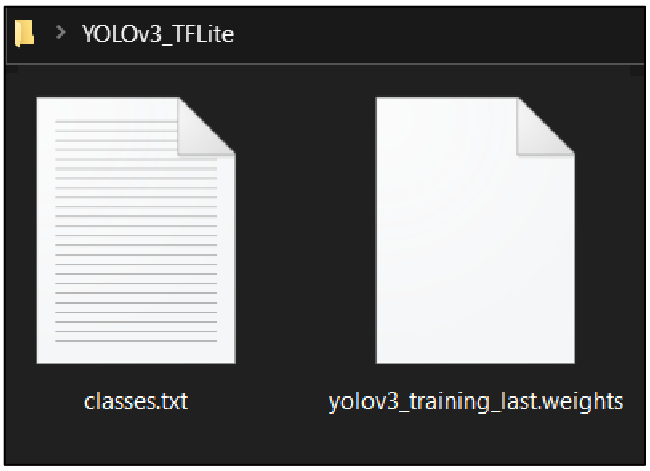
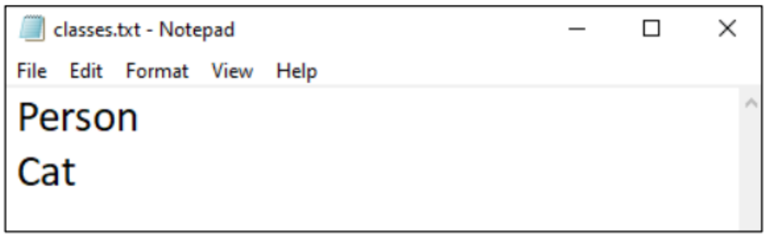


Fig-2. YOLOv3_TFLite directory

For your information, the model was trained for two classes – Person and Cat. The **classes.txt** file we are dealing with looks something like the following.



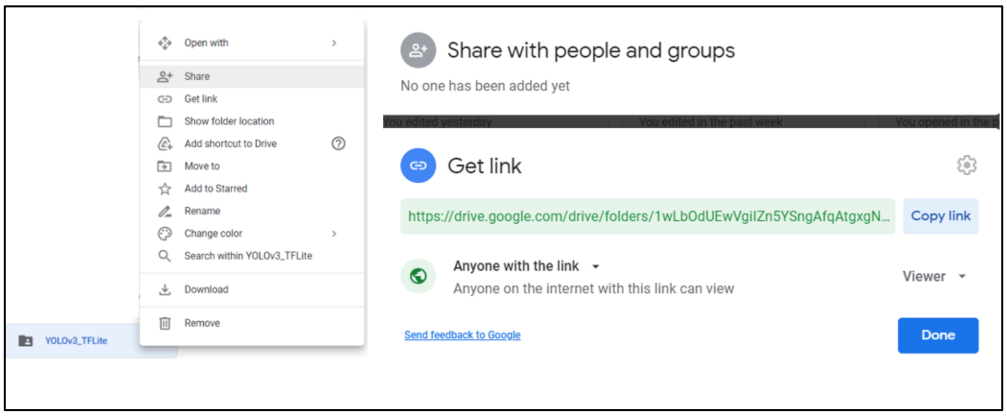
Person
Cat

Fig-3. classes.txt

Step 2: Upload the YOLOv3_TFLite folder on Google Drive.

Sign in to your Google account and open Google Drive. Upload the **YOLOv3_TFLite** folder there.

Now, right click on the folder and select the Share option. Change the permission to "Anyone with the link" as follows.



Share with people and groups

No one has been added yet

Get link

https://drive.google.com/drive/folders/1wLbOdUEwVgilZn5YSngAfqAtgxgN...

Anyone with the link

Anyone on the internet with this link can view

Step 3: Converting YOLOv3 model to TensorFlow Lite.

Now, the actual process of converting YOLOv3 model into TensorFlow Lite begins.

a. Setting up Google Colab.

The process of model conversion is an overkill task with the installation of certain libraries and different versions of TensorFlow. So, to make it easy, we will use Google Colab to ensure the process is smooth without worrying about manual installation of libraries on local machine and possibility of compatibility issues.

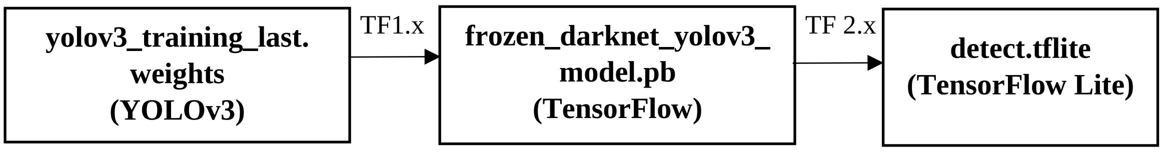
(i) Clone or download ZIP from the following GitHub repository on your local machine.

```
git clone https://github.com/NSTiwari/YOLOv3-to-
TensorFlow-Lite-Conversion
```

(ii) Open Google Colab and upload the **YOLOv3_to_TFLite_Conversion.ipynb** file from the downloaded repository.

b. Model Conversion.

The process of converting ***.weights** to ***.tflite** is illustrated below.


yolov3_training_last.weights (YOLOv3)
TF1.x
frozen_darknet_yolov3_model.pb (TensorFlow)
TF 2.x
detect.tflite (TensorFlow Lite)

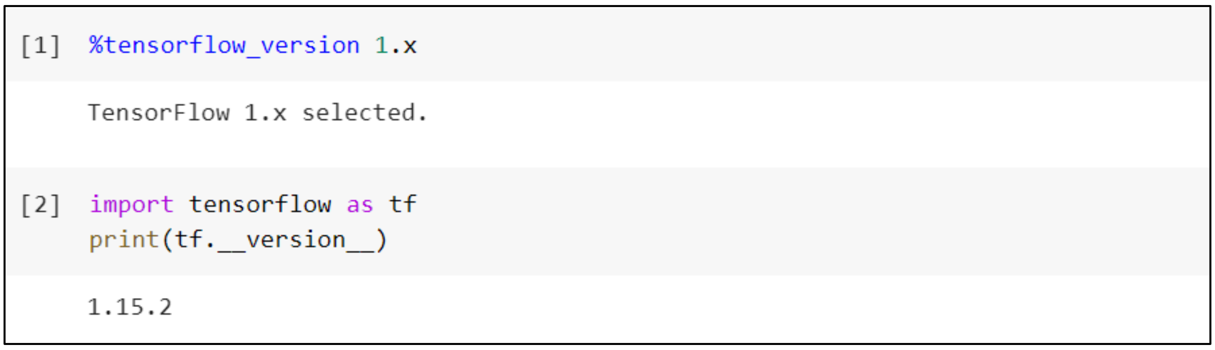
We first convert the **yolov3_training_last.weights** file into **frozen_darknet_yolov3_model.pb** which is a **protocol buffer** file. Information such as graph definition and weights of the model are frozen into one single file; hence the name "frozen model". This step involves the use of TensorFlow 1.x.

Next, the **frozen_darknet_yolov3_model.pb** is then converted into **detect.tflite**; a TensorFlow Lite version of the original model. This involves the use of TensorFlow 2.x.

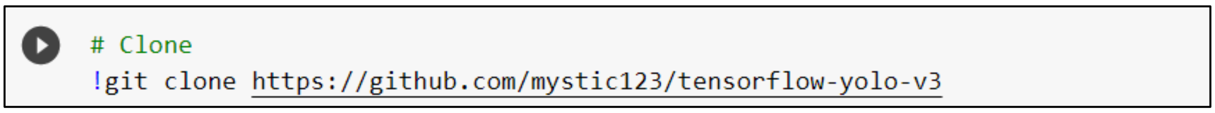
Let's see how we can do this ourselves by running the notebook cells one-by-one as follows.

(i) Setup TensorFlow 1.x.

Install TF 1.x required for the first sub-step of conversion.

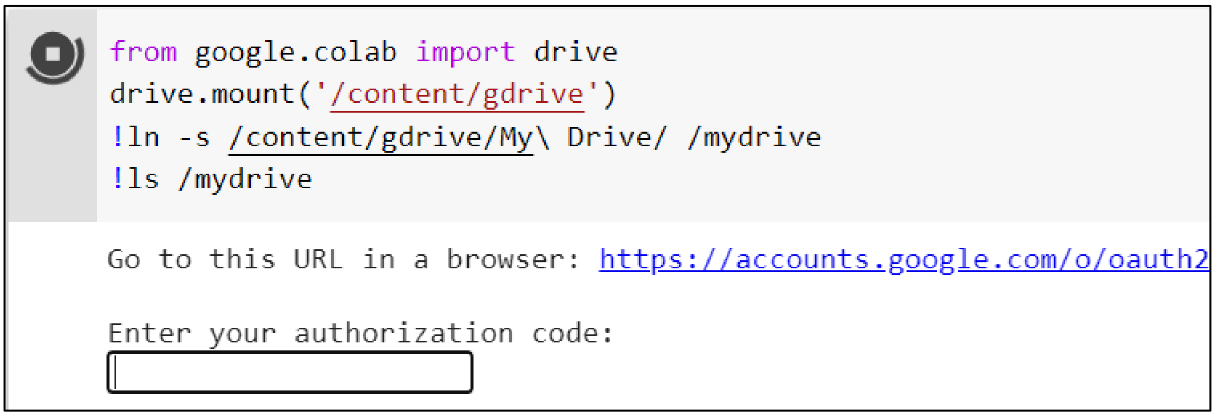

```
[1] %tensorflow_version 1.x

    TensorFlow 1.x selected.

[2] import tensorflow as tf
    print(tf.__version__)

    1.15.2
```

(ii) Clone the tensorflow-yolo-v3 repository.

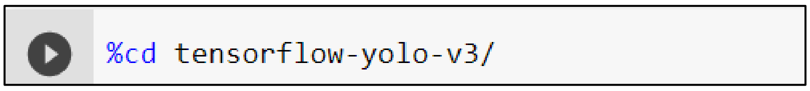

```
# Clone
!git clone https://github.com/mystic123/tensorflow-yolo-v3
```

(iii-a) Mount your Google Drive on Google Colab.

This is to provide access to the **YOLOv3_TFLite** folder you had uploaded on Google Drive in Step 2. Click on the link highlighted in blue and copy the authorization code that appears in a new tab. Paste it in the box below as shown.

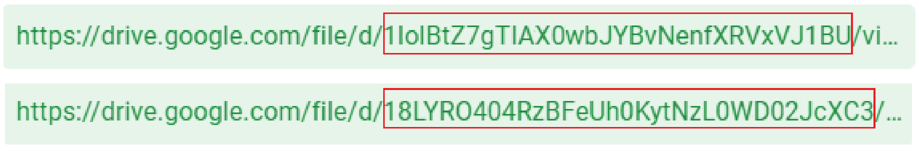

```
from google.colab import drive
drive.mount('/content/gdrive')
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive

Go to this URL in a browser: https://accounts.google.com/o/oauth2

Enter your authorization code:
```

(iii-b) Navigate to tensorflow-yolo-v3 repository.


```
%cd tensorflow-yolo-v3/
```

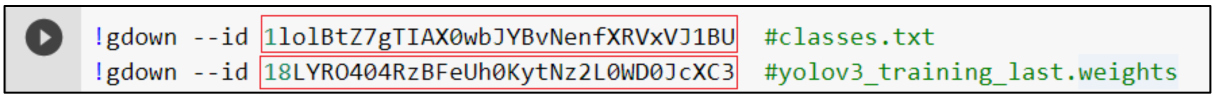
(iv) Download yolov3_training_last.weights and classes.txt files.

Open the **YOLOv3_TFLite** folder on your Google Drive. Right click on the **classes.txt** file and select Get link option. Copy the text highlighted in red as shown below. This is the unique id of the file.



```
https://drive.google.com/file/d/1lolBtZ7gTIAX0wbJYBvNenfXRVxVJ1BU/vi...
https://drive.google.com/file/d/18LYRO404RzBFeUh0KytNzL0WD02JcXC3/...
```

Paste the id in the figure as shown below. Do this for both the files.

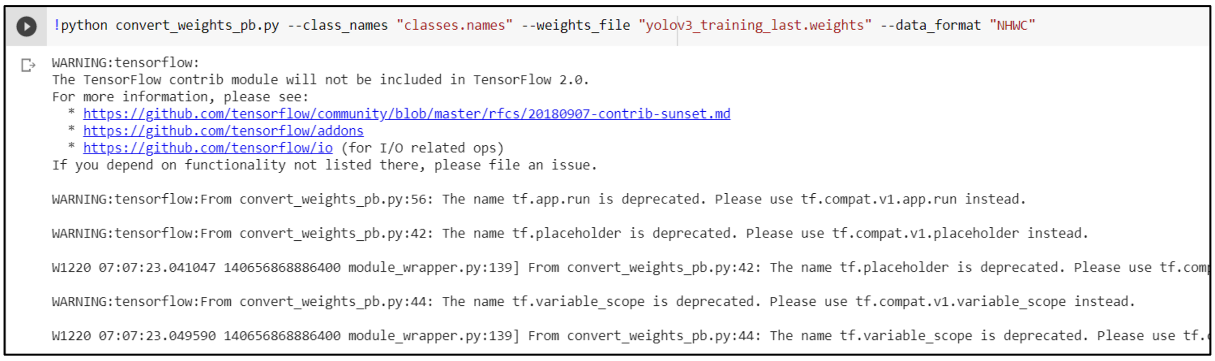


```
!gdown --id 1lolBtZ7gTIAX0wbJYBvNenfXRVxVJ1BU  #classes.txt
!gdown --id 18LYRO404RzBFeUh0KytNz2L0WD0JcXC3  #yolov3_training_last.weights
```

(v) Convert YOLOv3 to TensorFlow.

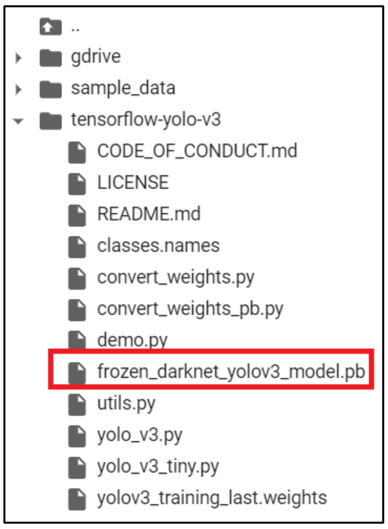
The cell below converts the YOLOv3 weights into a frozen model. The reason behind this is, for deploying the model on mobile, desktop or even browser-based applications, the acceptable format is ***.pb.**

On running the cell, you will see certain warnings as shown below. Don’t worry, that wouldn’t affect your model.



```
!python convert_weights_pb.py --class_names "classes.names" --weights_file "yolov3_training_last.weights" --data_format "NHWC"

WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.

WARNING:tensorflow:From convert_weights_pb.py:56: The name tf.app.run is deprecated. Please use tf.compat.v1.app.run instead.

WARNING:tensorflow:From convert_weights_pb.py:42: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

W1220 07:07:23.041047 140656868886400 module_wrapper.py:139] From convert_weights_pb.py:42: The name tf.placeholder is deprecated. Please use tf.com

WARNING:tensorflow:From convert_weights_pb.py:44: The name tf.variable_scope is deprecated. Please use tf.compat.v1.variable_scope instead.

W1220 07:07:23.049590 140656868886400 module_wrapper.py:139] From convert_weights_pb.py:44: The name tf.variable_scope is deprecated. Please use tf.
```

At the end of cell execution, a file named as **frozen_darknet_yolov3_model.pb** would be generated in the **tensorflow-yolo-v3** repository as shown below. This is the TensorFlow version of the original YOLOv3 model.

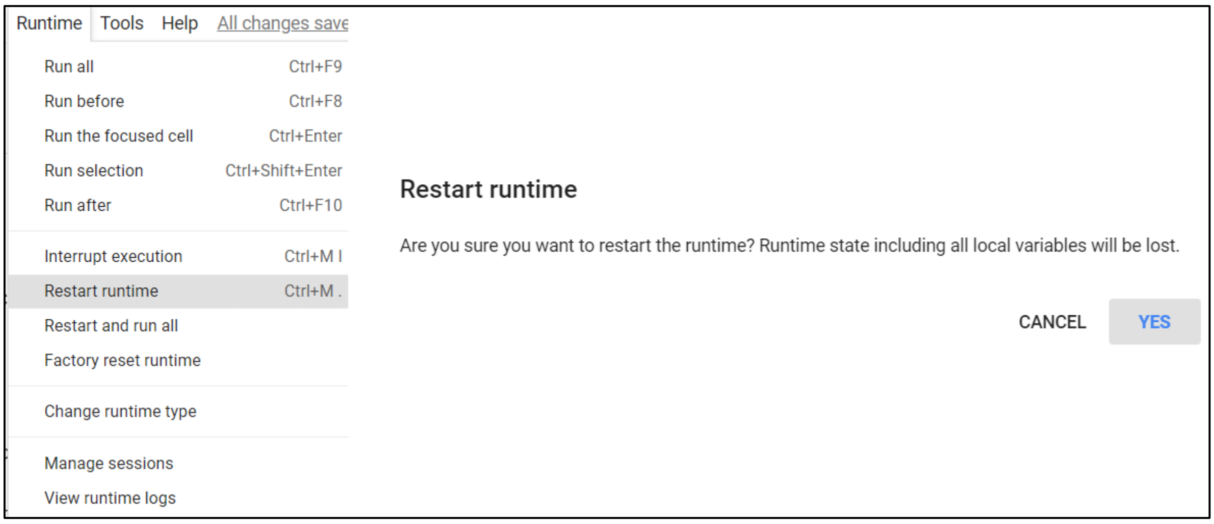


```
..
▸ gdrive
▸ sample_data
▾ tensorflow-yolo-v3
    CODE_OF_CONDUCT.md
    LICENSE
    README.md
    classes.names
    convert_weights.py
    convert_weights_pb.py
    demo.py
    frozen_darknet_yolov3_model.pb
    utils.py
    yolo_v3.py
    yolo_v3_tiny.py
    yolov3_training_last.weights
```

Great, we are halfway done. Let's now move ahead with the next part.

(vi-a) Restart runtime.

The next part involves the use of TensorFlow 2.x. So, we will need to restart the runtime to clear TensorFlow 1.x. Click on Runtime on the menu bar and select Restart runtime.

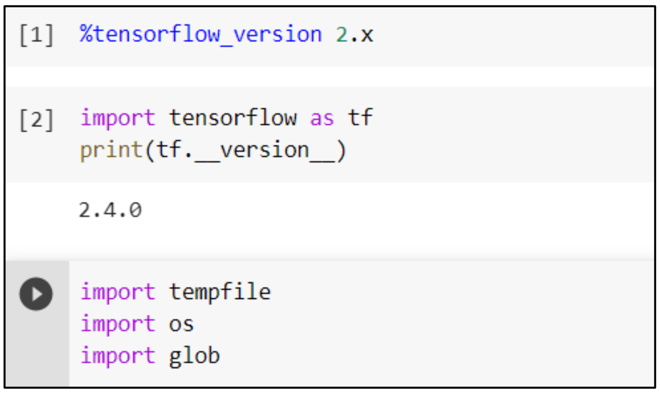


Restart runtime

Are you sure you want to restart the runtime? Runtime state including all local variables will be lost.

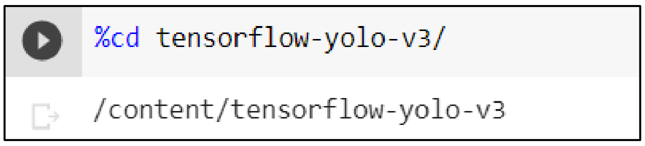
(vi-b) Setup TensorFlow 2.x.

Setup TensorFlow 2.x and other libraries required for the next steps.



```
[1] %tensorflow_version 2.x

[2] import tensorflow as tf
    print(tf.__version__)

    2.4.0

    import tempfile
    import os
    import glob
```

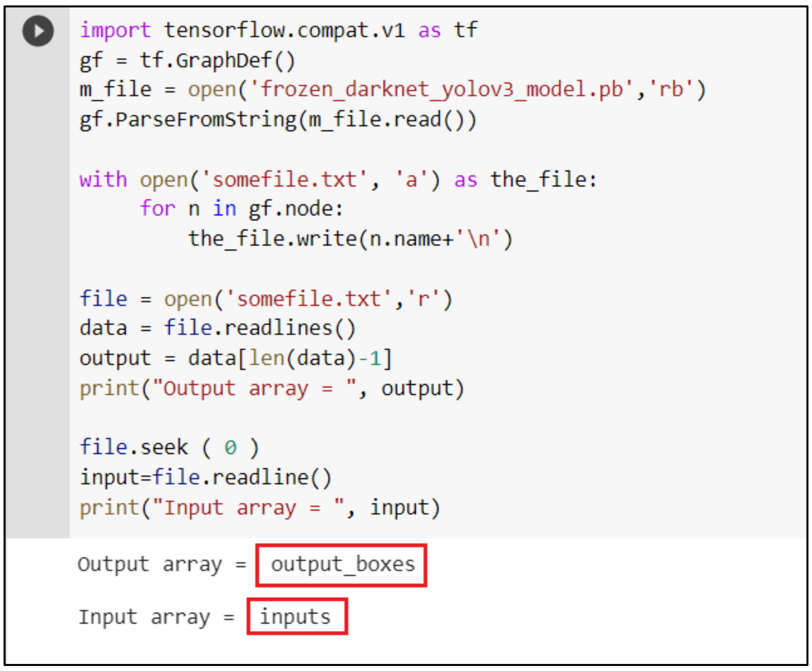
(vii-a) Navigate to tensorflow-yolo-v3 repository.



```
%cd tensorflow-yolo-v3/

/content/tensorflow-yolo-v3
```

(vii-b) Check input and output nodes of neural network.

Now, this step is important. In order to convert ***.pb** to ***.tflite**, it is necessary to know the input and output nodes of the neural network trained as they will be passed as parameters to the **TFLiteConverter** function. In case you don't, then run the cell below.

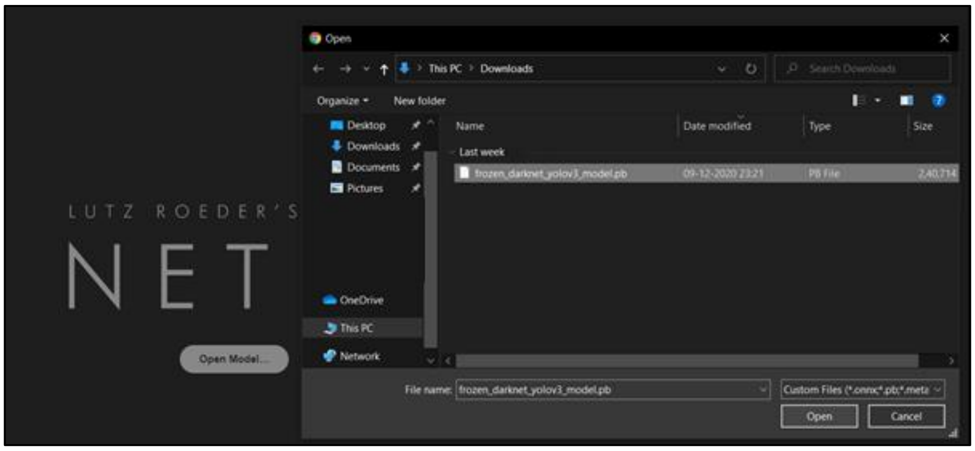


```
import tensorflow.compat.v1 as tf
gf = tf.GraphDef()
m_file = open('frozen_darknet_yolov3_model.pb','rb')
gf.ParseFromString(m_file.read())

with open('somefile.txt', 'a') as the_file:
     for n in gf.node:
         the_file.write(n.name+'\n')

file = open('somefile.txt','r')
data = file.readlines()
output = data[len(data)-1]
print("Output array = ", output)

file.seek ( 0 )
input=file.readline()
print("Input array = ", input)
```

```
Output array = output_boxes

Input array = inputs
```

The texts highlighted in red suggest that **"inputs"** is the input node and "**output_boxes"** is the output node of your original YOLOv3 model.

Alternatively, you can use **Netron** to visualize your model. To do that, download the **frozen_darknet_yolov3_model.pb** from the **tensorflow-yolo-v3** repository (obtained at Step (v)) on your local machine. Go to `www.netron.app` and click on Open Model. Choose the downloaded ***.pb** file.

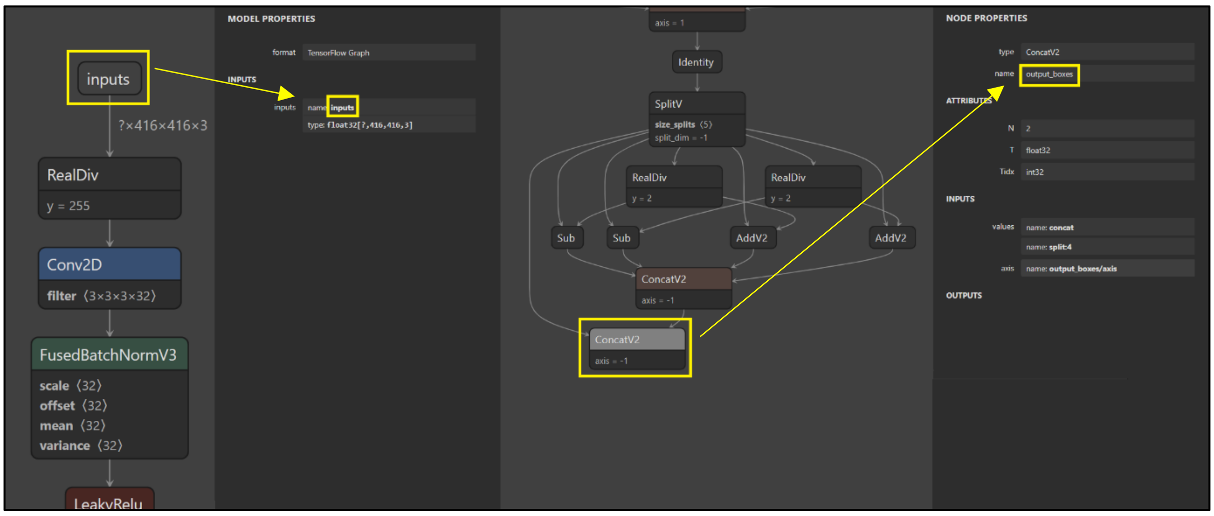


Name	Date modified	Type	Size
Last week			
frozen_darknet_yolov3_model.pb	09-12-2020 23:21	PB File	2,40,714

File name: frozen_darknet_yolov3_model.pb

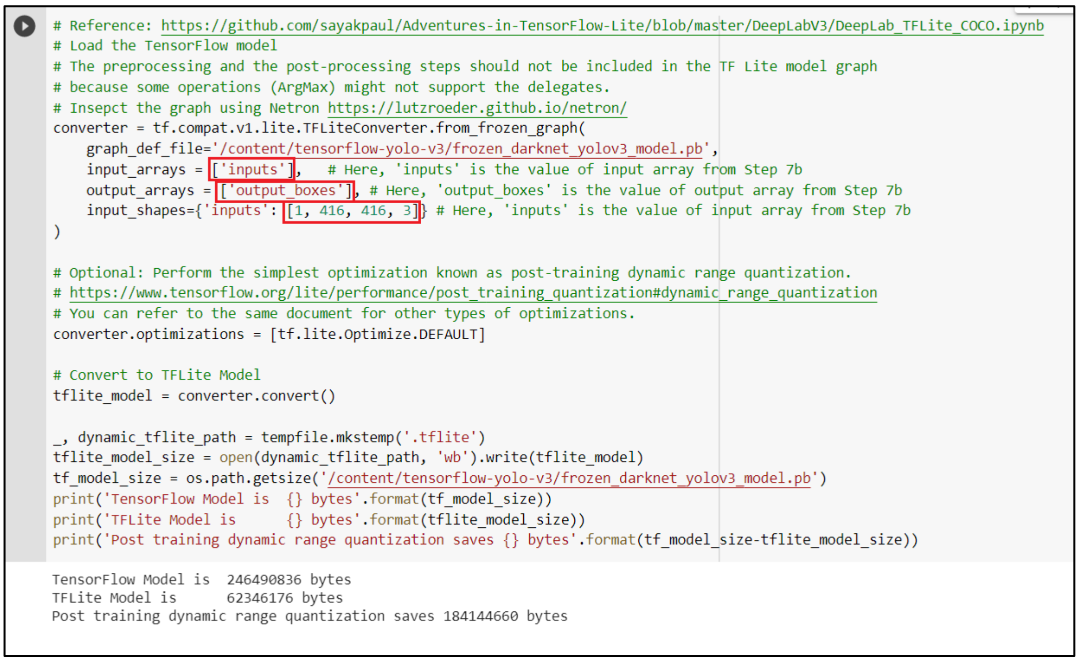
Custom Files (*.onnx;*.pb;*.meta

In the figure below, we can visualize the neural network structure of our model. The first node highlighted in yellow is the inputs layer and its name is **inputs.** The input size is 416 x 416 x 3 i.e., input images would be of dimension 416 x 416 with 3 channels (RGB). The last node of the network also highlighted in yellow, is the ConcatV2 layer and its name is **output_boxes.** Hence, we have correctly verified the input and output nodes.



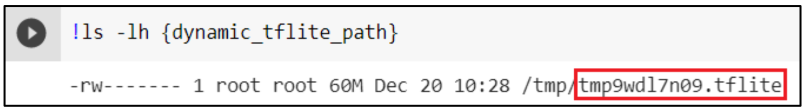
(viii) Convert TensorFlow model to TensorFlow Lite.

We are good to go for the next step. Run the cell below to convert ***.pb** to ***.tflite**. Note that the value of **input_arrays, output_arrays** and **input_shapes** parameters highlighted in red were derived in Step (vii-b). The TensorFlow model is then optimized by **quantization** before it is converted into TensorFlow Lite.



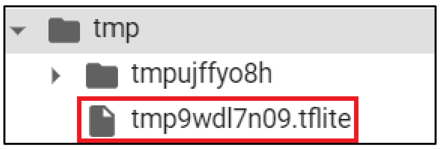
```
# Reference: https://github.com/sayakpaul/Adventures-in-TensorFlow-Lite/blob/master/DeepLabV3/DeepLab_TFLite_COCO.ipynb
# Load the TensorFlow model
# The preprocessing and the post-processing steps should not be included in the TF Lite model graph
# because some operations (ArgMax) might not support the delegates.
# Insepct the graph using Netron https://lutzroeder.github.io/netron/
converter = tf.compat.v1.lite.TFLiteConverter.from_frozen_graph(
    graph_def_file='/content/tensorflow-yolo-v3/frozen_darknet_yolov3_model.pb',
    input_arrays = ['inputs'],    # Here, 'inputs' is the value of input array from Step 7b
    output_arrays = ['output_boxes'], # Here, 'output_boxes' is the value of output array from Step 7b
    input_shapes={'inputs': [1, 416, 416, 3]} # Here, 'inputs' is the value of input array from Step 7b
)

# Optional: Perform the simplest optimization known as post-training dynamic range quantization.
# https://www.tensorflow.org/lite/performance/post_training_quantization#dynamic_range_quantization
# You can refer to the same document for other types of optimizations.
converter.optimizations = [tf.lite.Optimize.DEFAULT]

# Convert to TFLite Model
tflite_model = converter.convert()

_, dynamic_tflite_path = tempfile.mkstemp('.tflite')
tflite_model_size = open(dynamic_tflite_path, 'wb').write(tflite_model)
tf_model_size = os.path.getsize('/content/tensorflow-yolo-v3/frozen_darknet_yolov3_model.pb')
print('TensorFlow Model is  {} bytes'.format(tf_model_size))
print('TFLite Model is      {} bytes'.format(tflite_model_size))
print('Post training dynamic range quantization saves {} bytes'.format(tf_model_size-tflite_model_size))
```

```
TensorFlow Model is  246490836 bytes
TFLite Model is      62346176 bytes
Post training dynamic range quantization saves 184144660 bytes
```



```
!ls -lh {dynamic_tflite_path}

-rw------- 1 root root 60M Dec 20 10:28 /tmp/tmp9wdl7n09.tflite
```

The TensorFlow Lite model is saved as a temporary file inside the **tmp** folder as show below.

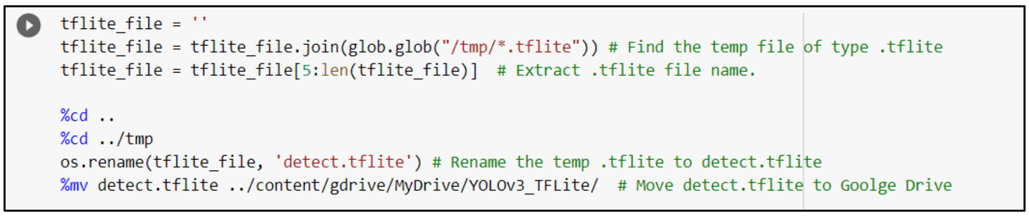


It can be observed that the size of the TensorFlow model before optimization and conversion was approximately 246 MB. The resulting TensorFlow Lite model is approximately 62 MB in size. This will slightly reduce the accuracy but there is a great reduction in the model size as well, so, that is a good tradeoff.

(ix) Move the TF Lite model to the YOLOv3_TFLite folder on Google Drive.

When a Google Colab session is terminated, all the files generated during the session would get deleted. Thus, we move the TensorFlow Lite model from the **tmp** folder to the **YOLOv3_TFLite** folder on Google Drive to ensure we have it saved permanently which can later be downloaded on local machine as and when required.

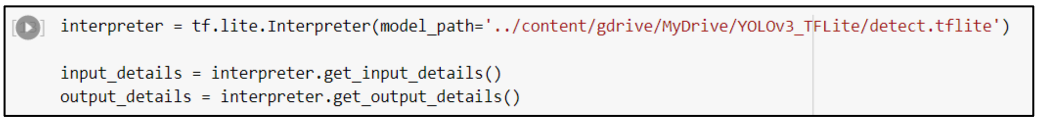
The cell below automatically finds the file with **.tflite** extension in the **tmp** folder and renames it as **detect.tflite** before it is moved to the **YOLOv3_TFLite** folder on Google Drive.



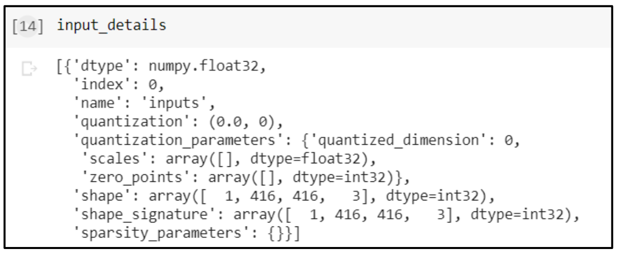
```
tflite_file = ''
tflite_file = tflite_file.join(glob.glob("/tmp/*.tflite")) # Find the temp file of type .tflite
tflite_file = tflite_file[5:len(tflite_file)]  # Extract .tflite file name.

%cd ..
%cd ../tmp
os.rename(tflite_file, 'detect.tflite') # Rename the temp .tflite to detect.tflite
%mv detect.tflite ../content/gdrive/MyDrive/YOLOv3_TFLite/  # Move detect.tflite to Goolge Drive
```

(x) Model Inference.

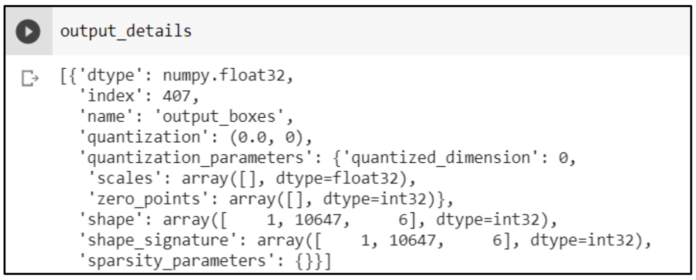
Run the cell below to interpret the TF Lite model.



```
interpreter = tf.lite.Interpreter(model_path='../content/gdrive/MyDrive/YOLOv3_TFLite/detect.tflite')

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```



```
[14] input_details

[{'dtype': numpy.float32,
  'index': 0,
  'name': 'inputs',
  'quantization': (0.0, 0),
  'quantization_parameters': {'quantized_dimension': 0,
   'scales': array([], dtype=float32),
   'zero_points': array([], dtype=int32)},
  'shape': array([  1, 416, 416,   3], dtype=int32),
  'shape_signature': array([  1, 416, 416,   3], dtype=int32),
  'sparsity_parameters': {}}]
```

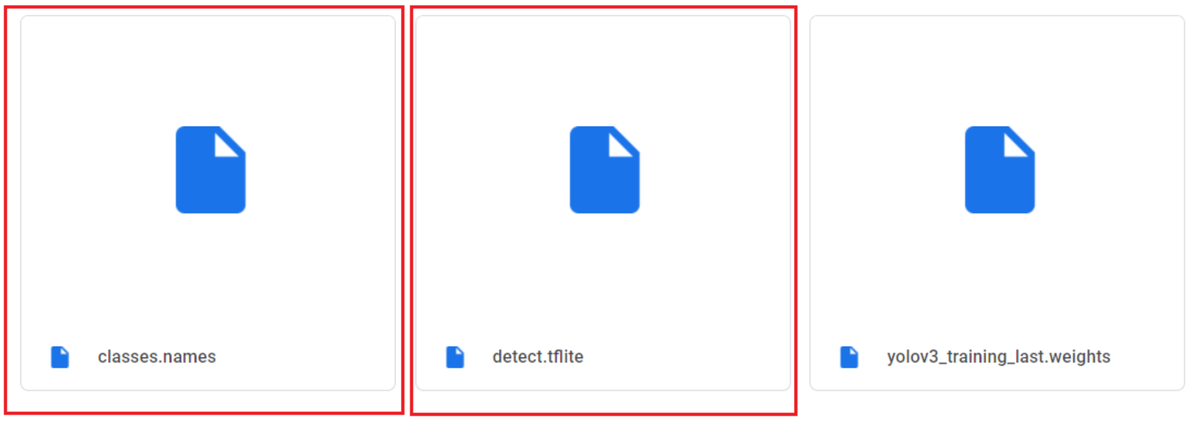


```
output_details
```

```
[{'dtype': numpy.float32,
  'index': 407,
  'name': 'output_boxes',
  'quantization': (0.0, 0),
  'quantization_parameters': {'quantized_dimension': 0,
   'scales': array([], dtype=float32),
   'zero_points': array([], dtype=int32)},
  'shape': array([    1, 10647,     6], dtype=int32),
  'shape_signature': array([    1, 10647,     6], dtype=int32),
  'sparsity_parameters': {}}]
```

Step 4: Testing the model.

Now, we are all set to test our model. We'll need two files for testing – **classes.names** and **detect.tflite.**

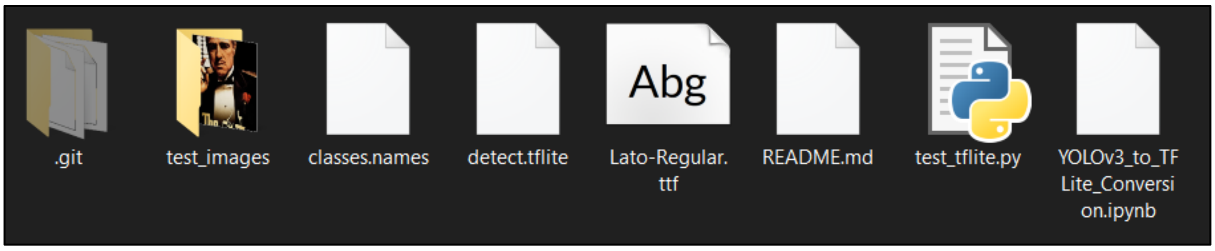


(i) Download **classes.names** and **detect.tflite** files from the **YOLOv3_TFLite** folder on Google Drive and save them into the **YOLOv3-to-TensorFlow-Lite-Conversion** repository on your local machine.

(ii) Create a new folder called **test_images** inside the **YOLOv3-to-TensorFlow-Lite-Conversion** repository and save some images inside it which you would like to test the model on.

(iii) Open the **test_tflite.py** file and edit Line 151 by replacing <your_test_image> with the name of image file you want to test.

The overall directory should look like this.

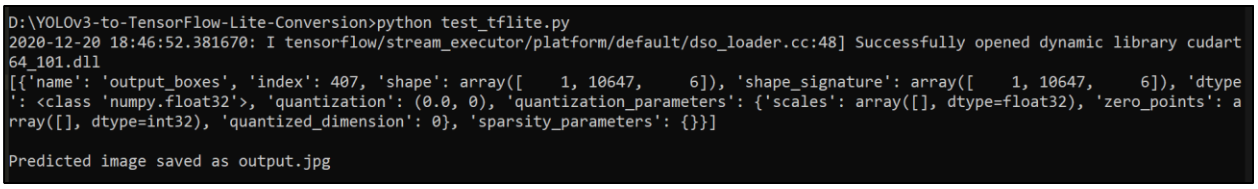


Before you test the model, make sure you have installed TensorFlow 2.x, OpenCV, NumPy and PIL libraries on your machine. If not, run the following commands on command prompt.

```
pip install tensorflow==2.3.1
pip install opencv-python
pip install opencv-contrib-python
pip install pillow
pip install numpy
```

Now, you are good to test the TF Lite model. Open command prompt and navigate to **YOLOv3-to-TensorFlow-Lite-Conversion** directory. Run the following command.

```
python test_tflite.py
```



```
D:\YOLOv3-to-TensorFlow-Lite-Conversion>python test_tflite.py
2020-12-20 18:46:52.381670: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cudart
64_101.dll
[{'name': 'output_boxes', 'index': 407, 'shape': array([    1, 10647,     6]), 'shape_signature': array([    1, 10647,     6]), 'dtype
': <class 'numpy.float32'>, 'quantization': (0.0, 0), 'quantization_parameters': {'scales': array([], dtype=float32), 'zero_points': a
rray([], dtype=int32), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]

Predicted image saved as output.jpg
```

The predicted image is then saved as **output.jpg** in the same directory.

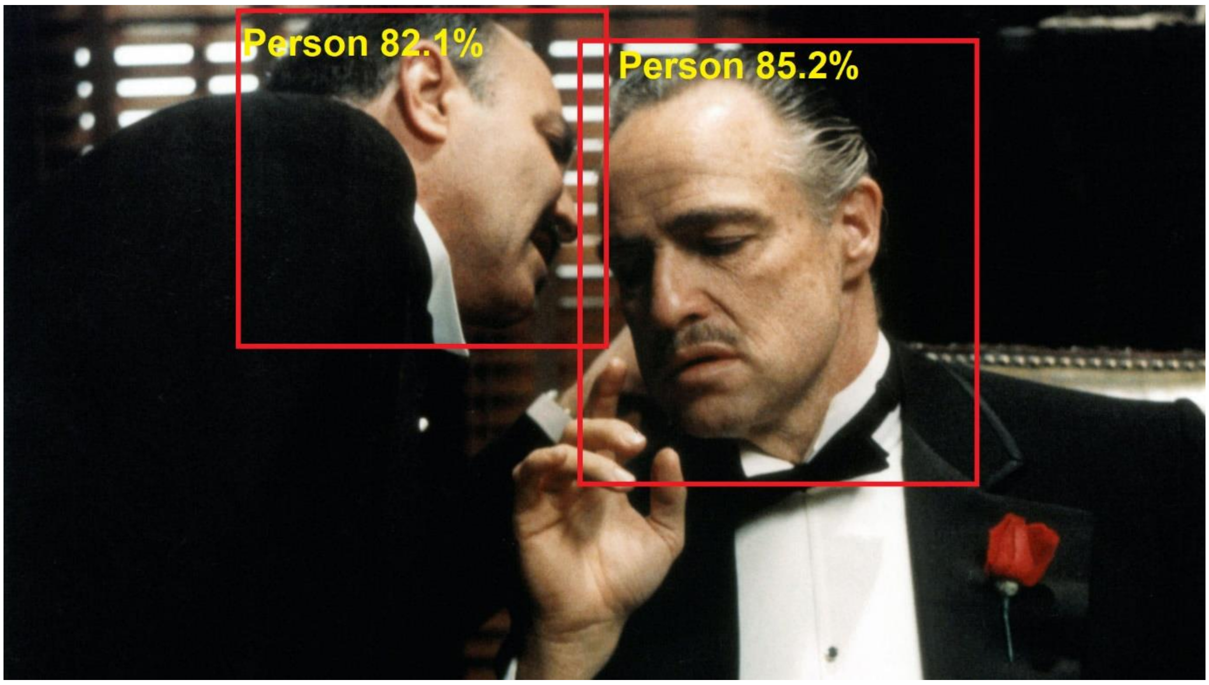


Fig-4. output.jpg

Congratulations, you have successfully completed your YOLOv3 model into a TensorFlow Lite model.