Object detection using Yolov3 on Google Colab

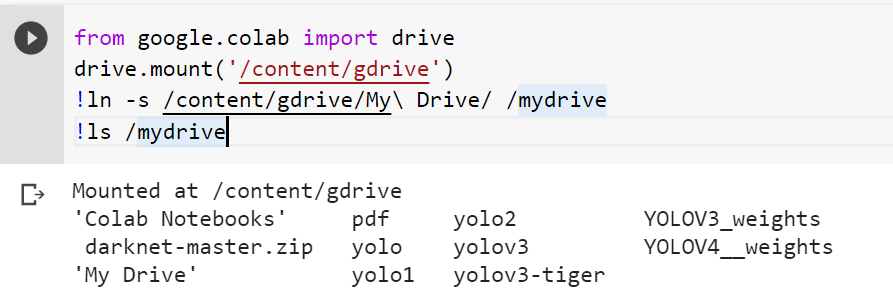
In this project we will learn how to train [YOLOv3](https://pjreddie.com/darknet/yolo/) on a custom dataset using the [Darknet](https://pjreddie.com/darknet/) framework. After following this will be having enough knowledge about object detection and you can just tune it according to your need and play.

**What is Object detection and why it is getting some much attention?**

* Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning and deep learning to produce meaningful results. When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this intelligence using a computer.
* The use cases are endless, be it Tracking objects, Video surveillance, Pedestrian detection, Anomaly detection, People Counting, Self-driving cars or Face detection, the list goes on.
* After labeling all the images place the images and labels files in the same folder and make a zip file of that folder and upload it to google drive.
* Now open google colab and run the cells one by one

**Step-1:**

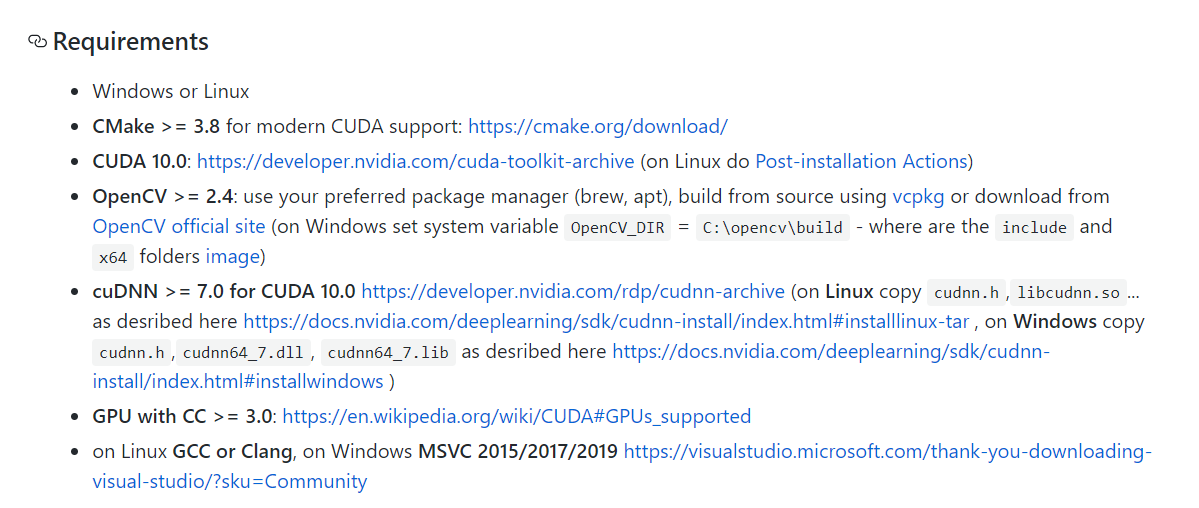
Open google colab new notebook and mount your drive. And check the folders present in the drive by using ls command.



**Step-2:**

Installing Darknet

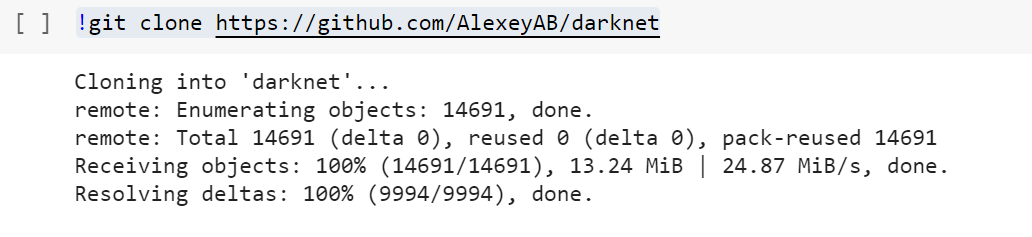
Darknet is an open-source framework that supports Object Detection and Image Classification tasks in the form of Convolutional Neural Networks. Darknet is mainly known for its implementation of the YOLO algorithm (You Only Look Once), which has demonstrated state of the art performance when it comes to real-time object detection.



Darknet is an open-source neural network framework supports CPU and GPU computation. Darknet can also by downloaded directly in google colab for this project we will directly download in colab but just download it either from chrome or using cmd and save it in a directory.

**For downloading using Command prompt use the below command:**

git clone <https://github.com/pjreddie/darknet>



Preparing YoloV3 configuration files

### **I) Configuration Files**

YOLO is entirely plug-n-play, that is, you can configure YOLO to detect any type of objects. In fact, you can also modify the CNN architecture itself and play around with it. YOLO does this by making use of configuration files under **cfg/**. The configuration files end with .cfg extension, which YOLO can parse.

These configuration files consist of mainly:

* CNN Architectures (layers and activations)
* Anchor Boxes
* Number of classes
* Learning Rate
* Optimization Technique
* input size
* probability score threshold
* batch sizes

**Now we have to make changes to the file we have downloaded, This include steps as follows:**

* Open darknet-master folder which we have just downloaded and from that open cfg folder now in the cfg folder make a copy of the file yolov3.cfg now rename the copy file to yolo-obj.cfg
* Open the file yolo-obj.cfg and change max\_batches to (classes\*2000),if you have 6 object classes change max\_batches=12000.
* Then change the line steps to (0.8\*max\_batches ,0.9\*max\_batches) ie; if you have 6 classes steps=9600,10800.
* Set network size width=416 height=416
* Change line classes=80 to your number of objects in each of 3 yolo layers. i.e since in this project we are having only one class we will use classes=1
* Change [filters=255] to filters=(classes + 5)x3 in the 3 convolutional layer immediately before each 3 yolo layers. The number 5 is the count of parameters center\_x, center\_y, width, height, and objectness Score.
* If you have 6 classes filters=33
* save the file and upload it to the drive

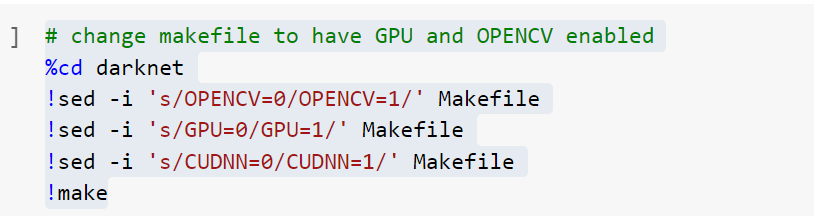
**Step-3:**

change the directory to darknet and change makefile to have GPU and OPENCV enabled.

If you are using GPU, change GPU from 0 to 1 and if you have installed OpenCV set OPENCV to “1”, otherwise not need.

Save the Makefile and run the command in the root directory to start the build process.

Python is written in C so c/c++ extensions (cython) is make execution faster. and yeah you are right cython allows you to code c/c++ on python script. 2) Make is not specific for python. when you build a envinorment you need dependencies, env variables, folders or scripts to run. When we look at the mentioned makefile <[github.com/AlexeyAB/darknet/blob/…](https://github.com/AlexeyAB/darknet/blob/9c1b9a2cf6363546c152251be578a21f3c3caec6/Makefile#L1)> it specifies the environment variables, and paths, ensure compile tools are right



**Step-4:**

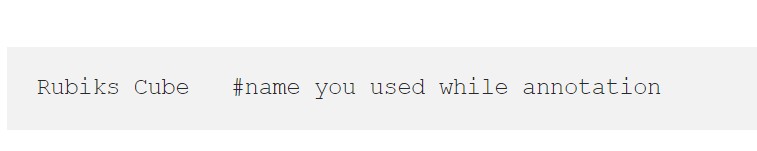
We need to create “**obj.names**” which its name implies that it contains names of classes, and the file “**obj.data**” which contains parameters needed for training. Here we directly create the two files in colab for this project. Replace tiger with the classes on which you are training. Change the backup folder (drive folder) to the folder where you want to save the weights. Place the files in the same folder as mentioned in obj.data file



**Sample info present in obj.data**



**Sample info present in obj. names**



**Step-5:**

### II) Weights

Each configuration has corresponding pre-trained weights. Here, only YOLO V3 is referenced.

#### Full Weight This is the weight trained on full 9000+ classes.

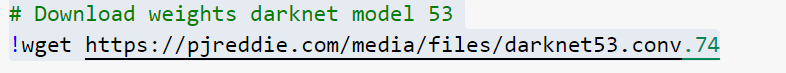
#### Tiny Weight

This is the weight trained on only 80 classes.

Installing Darknet.53.conv.74

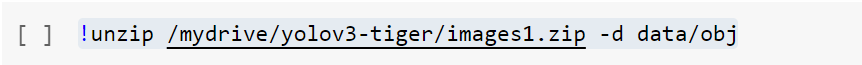
darknet53.conv.74.weights contains the weights for the DarkNet network\* originally trained for classification on the ImageNet dataset, which is used as the pre-trained feature extractor (backbone) for YOLOv3. To use this for detection the additional weights which are only present in the YOLOv3 network are randomly initialized prior to training.

\*darknet53.conv.74.weights refers to the fact that only the weights for the convolutional layers are included, excluding the weights for the final fully connected layer which outputs the class probabilities for classification. There is a separate file called darknet53.weights which includes *all* the weights.

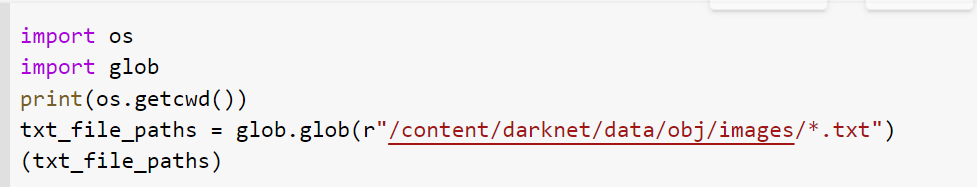


**Step-6:**

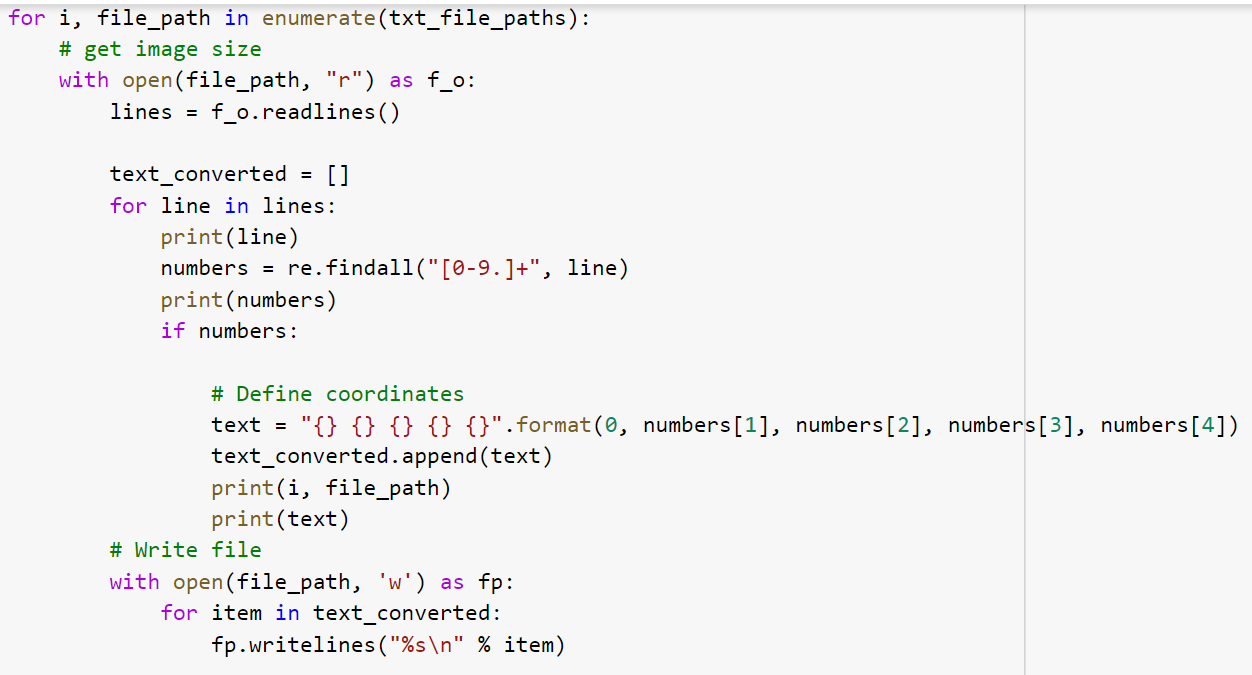
Unzip the image folder and store it in present data directory which is a sub directory of current working directory.



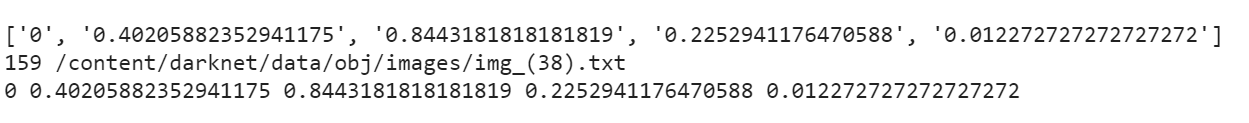
Store the paths of the txt files in the variable txt\_file\_paths



by using the above txt\_file\_paths we will format the data and then again write to the same file.



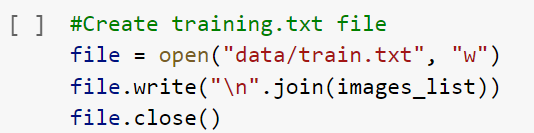
**Sample output of the above code**



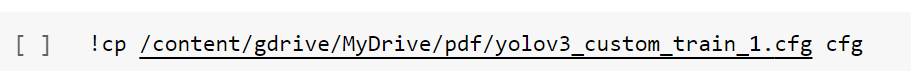
**Creating a list of all the image paths**



writing the paths of all the images in the train.txt file and storing the train.txt file in data folder of cwd.



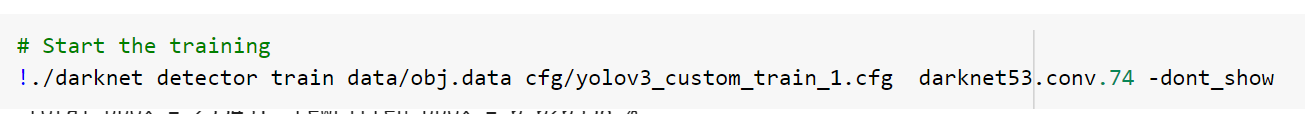
**Copying the cfg file from our google drive and storing into the cfg folder of cwd.**



**Step-7:**

**Start Training**

Time for the fun part! Enter the following command into your terminal and watch your GPU do what it does best.



**References:**

**For understanding about the cfg file:**

<https://stackoverflow.com/questions/50390836/understanding-darknets-yolo-cfg-config-files/50696918>

**queries related to darknet53 and retraining the model**

<https://github.com/pjreddie/darknet/issues/1878#:~:text=darknet53.-,conv.,data%20you%20can%20use%20darknet53>.

**What is darknet?**

<https://www.predictiveanalyticstoday.com/darknet/#:~:text=8.0-,Darknet%20is%20an%20open%20source%20neural,written%20in%20C%20and%20CUDA.&text=The%20framework%20features%20You%20Only,%25%20on%20COCO%20test%2Ddev>.

Why Yolov3 was introduced?

<https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>