

Weed Detection Using YOLOV5

YOLO is a popular real-time object detection algorithm. The reason of YOLO's popularity is its speed. Unlike other object detection algorithms that perform region proposal and classification separately, YOLO performs both tasks in a single forward pass of the network, making it faster and more efficient. Hence it can be used in computer vision applications which are real time. For example a self-driving car might need to detect an obstacle using image from camera. Hence to avoid crash or accident this decision needs to be executed in real time.

YOLOv5 which was released in 2020 by Ultralytics is one of the variations of YOLO. In this article we will train YOLOV5 for weed detection. Weed detection technology can help farmers to precisely target weed infestations and reduce the use of herbicides, resulting in cost savings and a more sustainable farming approach.

The first step in any object detection method is data collection and annotation. After collecting data you have to identify the number of classes in your data. In the data which we have collected we have 7 classes as namely karela tori bhindi horseweed herb paris grass and small weed. Some of them are as follows.

Crop Types:
Karela:



Tori:



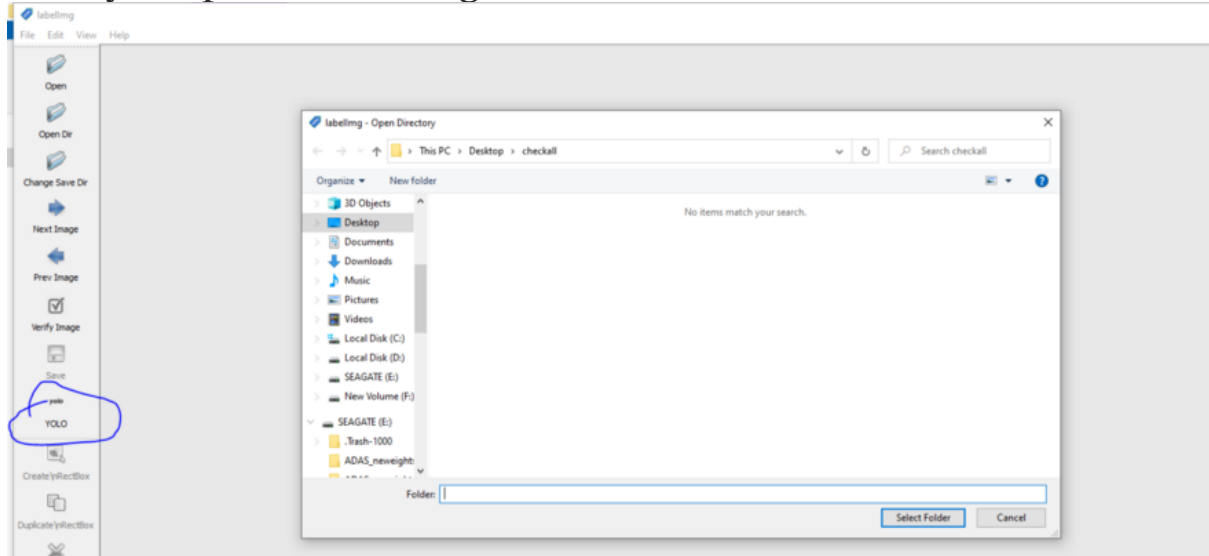
Bhindi:



Once you have identified the classes the next step is to annotate the data. There are many tools available for data annotation online and offline. Upto my research i have found LabelImg to be better. Its

lightweight and easy to use. LabelImg can be downloaded from [here](#). The direct exe file for windows can be found [here](#).

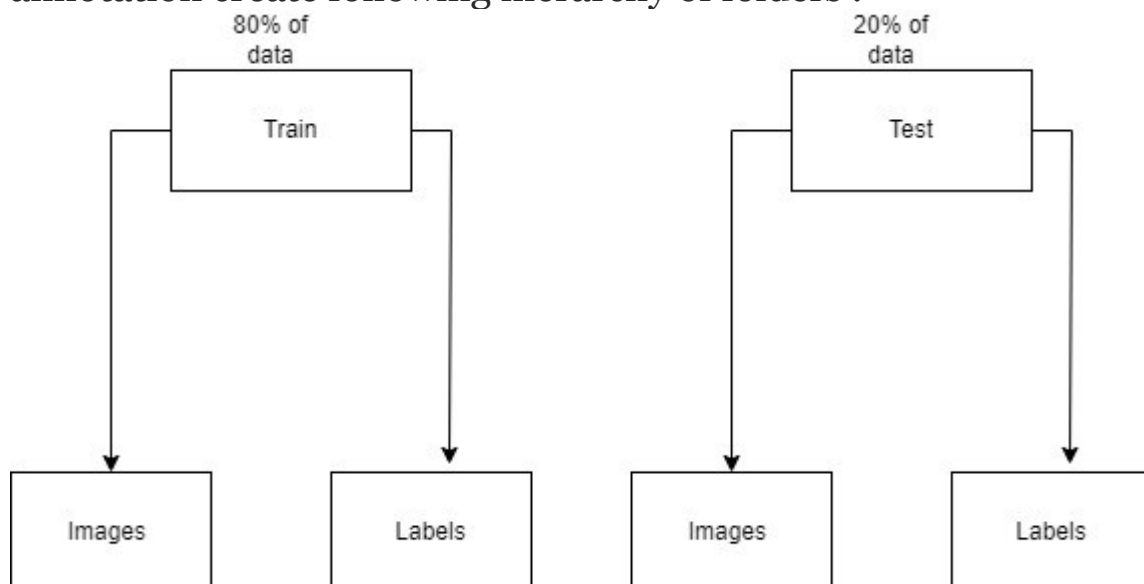
Once you open it LabelImg looks like the follows.



Select YOLO option as we are using YOLOV5 which supports .txt format. Other models requires the data to be in other formats. Next open the classes file and write the names of classes in order you want. The first class will be assigned the number 0 second name 1 and so .

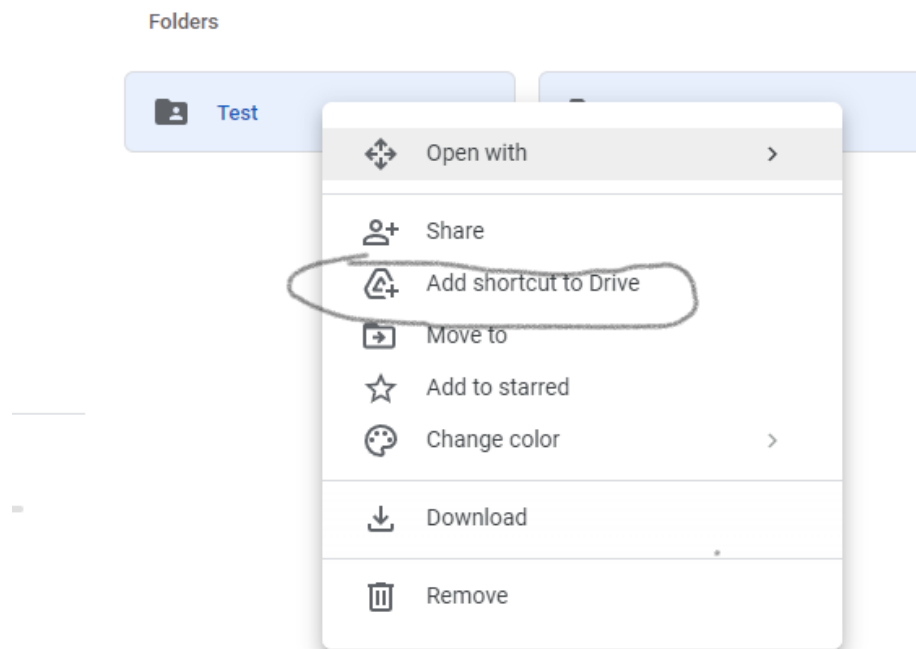


After creating bounding box. Assign class name and save the file.
Move to next one. In this annotate all the images. Once done
annotation create following hierarchy of folders .

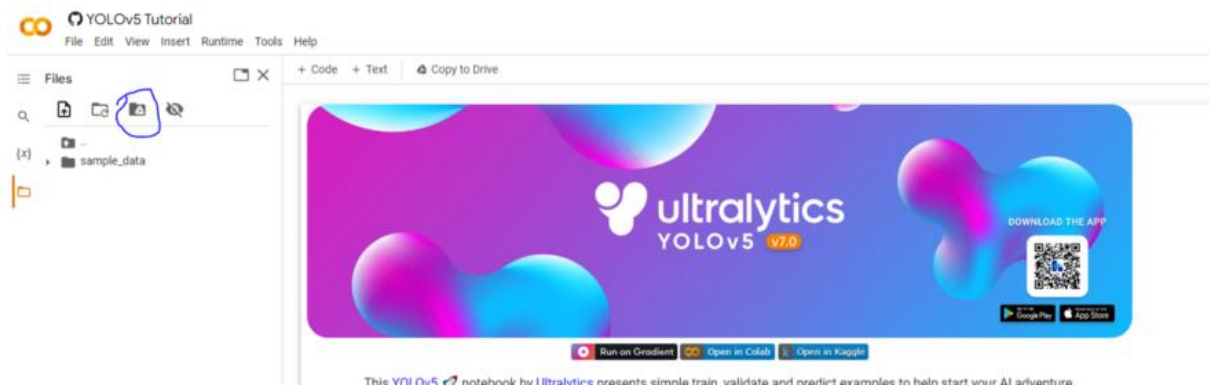


if you have 100 images then you will 80 images for training and 20 for testing. Once you have arranged your files like this upload them on your drive. I have uploaded my annotated files [here](#).

To use my data you will have to open it and add a shortcut to your drive. The better approach will be creating a weededetection named folder and add the shortcut there.

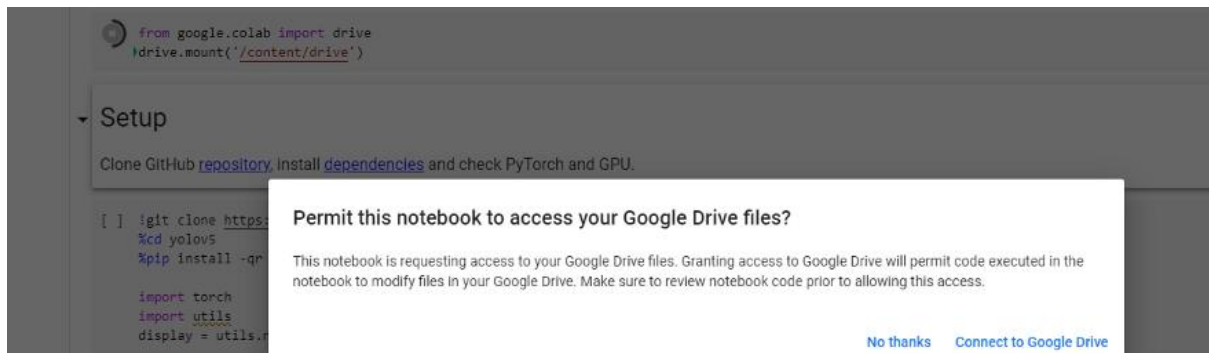


After creating shortcut open YOLOV5 Ultralytics official colab tutorial notebook which can be found [here](#). After opening the notebook the next step is to mount you drive which can be done as follows



click on the mount drive icon

This will create a code snippet as follows run it. Allow this notebook to access your google drive files by clicking on Connect to Google drive.



After allowing it you will be able to access your drive files from colab environment.

The next step is to create .yaml file according to your custom dataset. In yaml file you have to describe the classes in order you annotated them. For example let's say in your dataset you have 0 for cat and 1 for dog . You will follow this order in Yaml too. The YAML file for weed dataset is given as follows

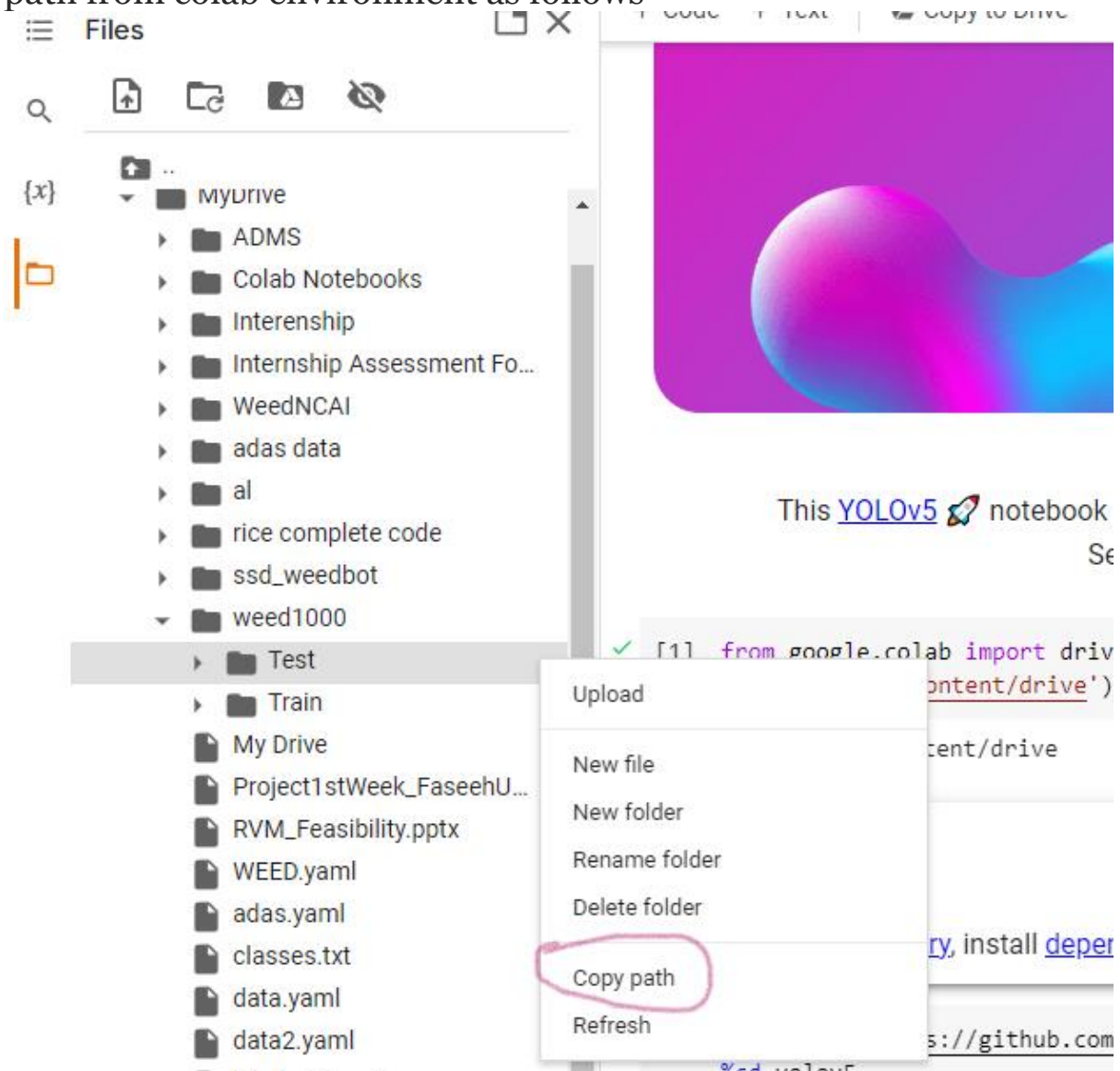
```
train: relative path to the folder where train data is
val: relative path to the folder where test data is
# Classes

nc: 7 # number of classes
```



```
names: ['herb paris', 'karela', 'small weed', 'grass', 'tori', 'horseweed', 'Bhindi'] # class names
```

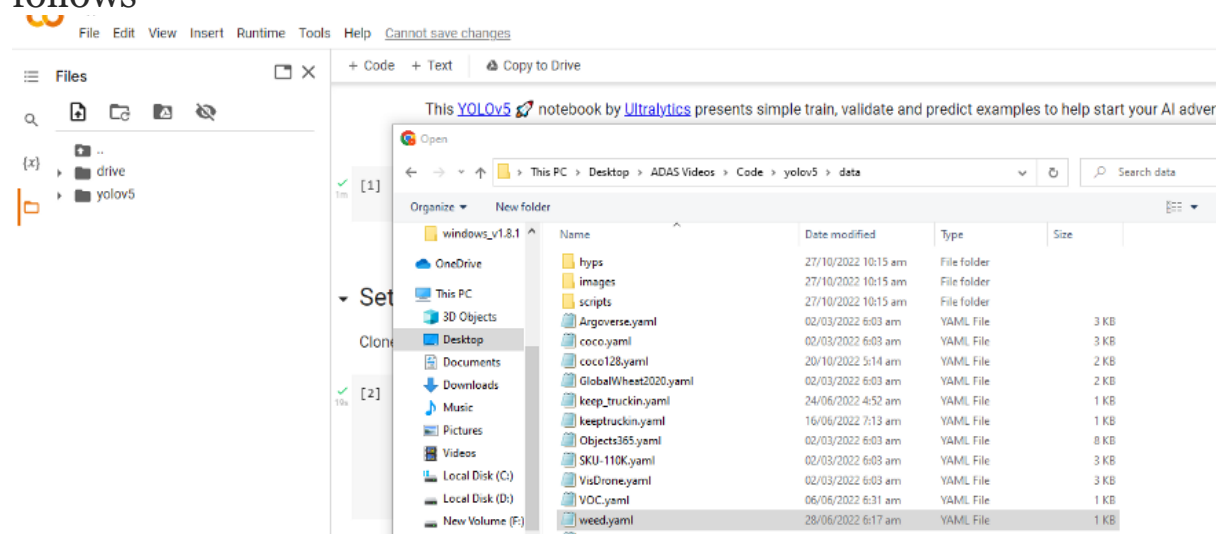
To create this yaml file open text editor and copy the content. Now you have to add the train folder path to it. For that navigate to train path from colab environment as follows



Paste this path in front of train as follows. Do the same val. Note in the given scenario val is same as test. So you will copy path of test folder.

```
train: /content/drive/MyDrive/weed1000/Test
```

once you have created yaml file. You can give it any name ending with .yaml. In my case i have named it as weed.yaml. Now we will upload this file to either our notebook environment or drive. As follows



Upload weed.yaml file

Once you are done with it. You just have to run two cells. Which are as follows

```
!git clone https://github.com/ultralytics/yolov5 # clone YOLO github repository
%cd yolov5
%pip install -qr requirements.txt # install

import torch
import utils
display = utils.notebook_init() # checks
```


The above cell will clone Ultralytics YOLOv5 repo to your notebook and install requirements. The as we have already created yaml file the next step is to start the training.

```
!python train.py --img 640 --batch 16 --epochs 3 --data <pathto weed.yamlfile> --weights yolov5s.pt --cache
```

Note next to data you have to paste the path to weed.yaml file which we have created. so for my case the code will be as follows

```
!python train.py --img 640 --batch 16 --epochs 3 --data /content/weed.yaml --weights yolov5s.pt --cache
```

Also next to weight we have written yolov5s.pt this means yolov5 small version. There are more versions , which include the followings

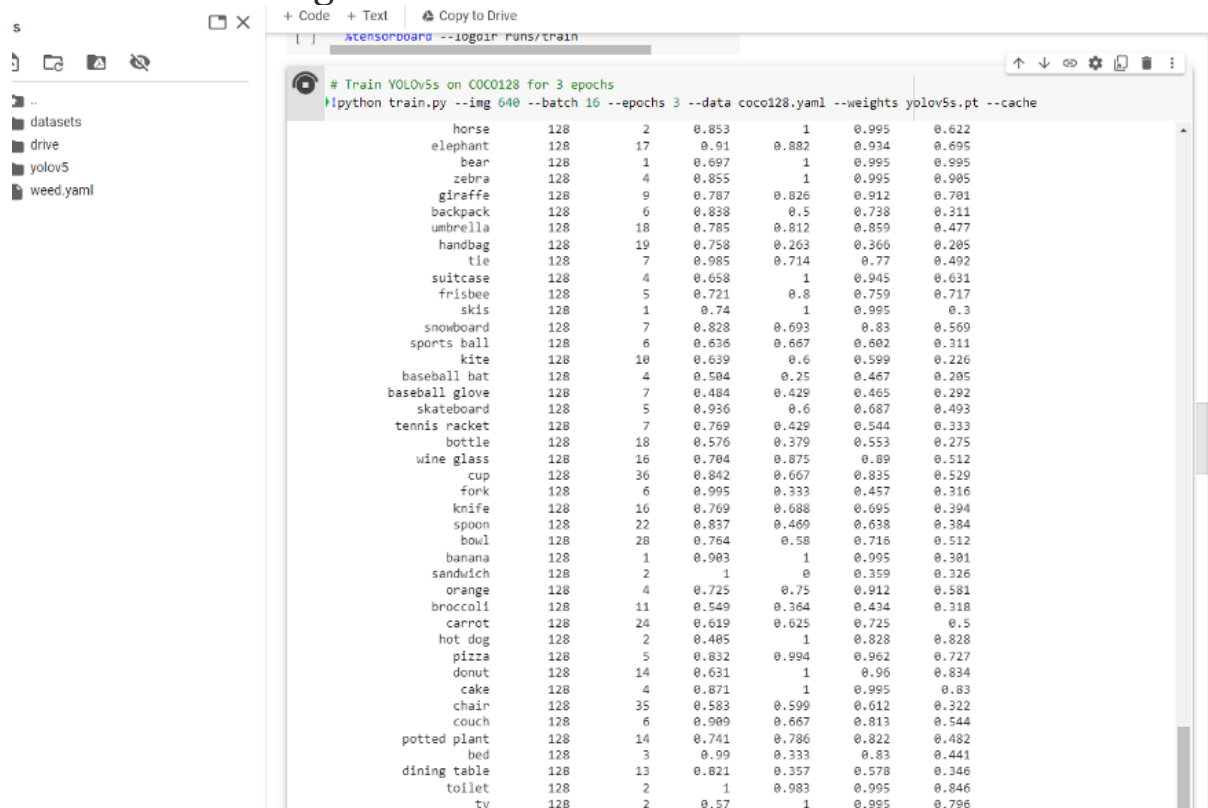
Pretrained Checkpoints

Model	size (pixels)	mAP ^{val} 50-95	mAP ^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6	1280	55.0	72.7	3136	26.2	19.4	140.7	209.8
+ TTA	1536	55.8	72.7	-	-	-	-	-

yolov5n nano being the lightest to yolov5l being the largest one.

There is always a trade off between these models. As speed increases

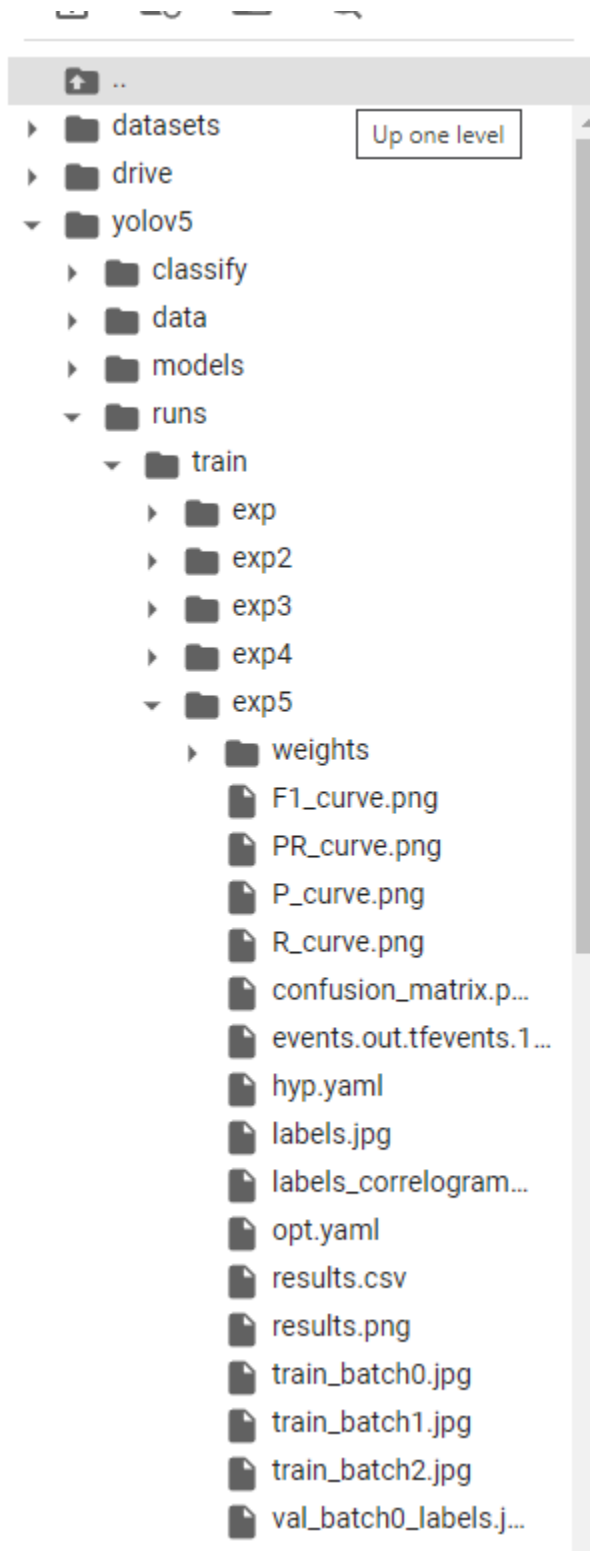
accuracy of the model decreases. Now when you will run the the above cell training will start as follows.



```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 16 --epochs 3 --data coco128.yaml --weights yolov5s.pt --cache
```

horse	128	2	0.853	1	0.995	0.622
elephant	128	17	0.91	0.882	0.934	0.695
bear	128	1	0.697	1	0.995	0.995
zebra	128	4	0.855	1	0.995	0.905
giraffe	128	9	0.787	0.826	0.912	0.701
backpack	128	6	0.838	0.5	0.738	0.311
umbrella	128	18	0.785	0.812	0.859	0.477
handbag	128	19	0.758	0.263	0.366	0.205
tie	128	7	0.985	0.714	0.77	0.492
suitcase	128	4	0.658	1	0.945	0.631
frisbee	128	5	0.721	0.8	0.759	0.717
skis	128	1	0.74	1	0.995	0.3
snowboard	128	7	0.828	0.693	0.83	0.569
sports ball	128	6	0.636	0.667	0.602	0.311
kite	128	10	0.639	0.6	0.599	0.226
baseball bat	128	4	0.504	0.25	0.467	0.205
baseball glove	128	7	0.484	0.429	0.465	0.292
skateboard	128	5	0.936	0.6	0.687	0.493
tennis racket	128	7	0.769	0.429	0.544	0.333
bottle	128	18	0.576	0.379	0.553	0.275
wine glass	128	16	0.704	0.875	0.89	0.512
cup	128	36	0.842	0.667	0.835	0.529
fork	128	6	0.995	0.333	0.457	0.316
knife	128	16	0.769	0.688	0.695	0.394
spoon	128	22	0.837	0.469	0.638	0.384
bowl	128	28	0.764	0.58	0.716	0.512
banana	128	1	0.903	1	0.995	0.301
sandwich	128	2	1	0	0.359	0.326
orange	128	4	0.725	0.75	0.912	0.581
broccoli	128	11	0.549	0.364	0.434	0.318
carrot	128	24	0.619	0.625	0.725	0.5
hot dog	128	2	0.405	1	0.828	0.828
pizza	128	5	0.832	0.994	0.962	0.727
donut	128	14	0.631	1	0.96	0.834
cake	128	4	0.871	1	0.995	0.83
chair	128	35	0.583	0.599	0.612	0.322
couch	128	6	0.909	0.667	0.813	0.544
potted plant	128	14	0.741	0.786	0.822	0.482
bed	128	3	0.99	0.333	0.83	0.441
dining table	128	13	0.821	0.357	0.578	0.346
toilet	128	2	1	0.983	0.995	0.846
tv	128	2	0.57	1	0.995	0.796

Once th trainig is finished you can see different training metrics in the folder runs/train



The weight folder contains the train model. To increase the performance you can increase epochs and datasize. Now to make a

prediction on a video or an image upload the video or image and run the following code.

```
!python detect.py --weights <path to trained model> --img 640 --conf 0.25  
--source <path to video or image> >
```

remember in the next to weight you will give the path of trained model and after the source you will write the path of video on which you want to perform detection using your trained model. Note download your trained model so you can use it for deployment or use.

```
!python detect.py --weights  
/content/yolov5/runs/train/exp5/weights/best.pt --img 640 --conf 0.25 --  
source /content/weed13858.jpg
```

Once you run this the output will be saved in the following folder

```
/content/yolov5/runs/detect
```

So that's all of it. By using the same procedure you can train YOLOV5 object detection for any object of your choice.