Tutorial by Nooh Ayub

(Yolo v5 for Face Mask Detection Pretrained Model)

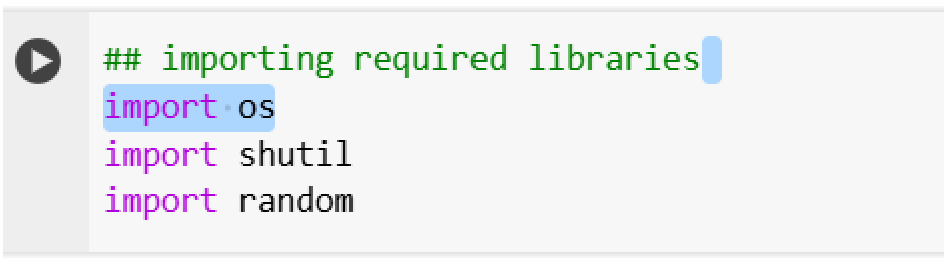
Objectives:

- Understand Pre trained model
- Loading pre trained model from Ultralytics github Repsoitory
- Evaluating the model on face mask test dataset:

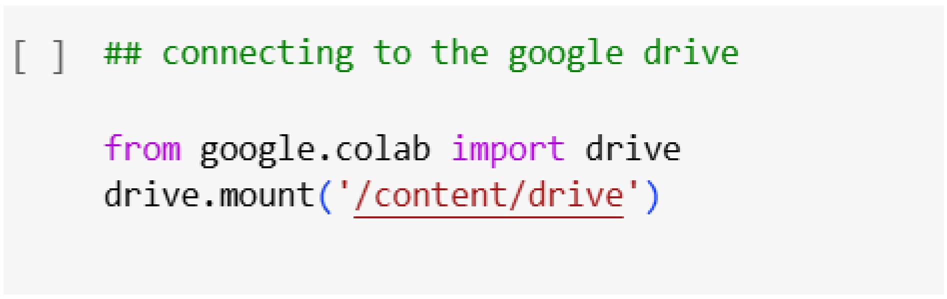
Here's a step by step breakdown:

Step 1 —Open GoogleColab and import Libraries:

We will import the important libraries to get started.

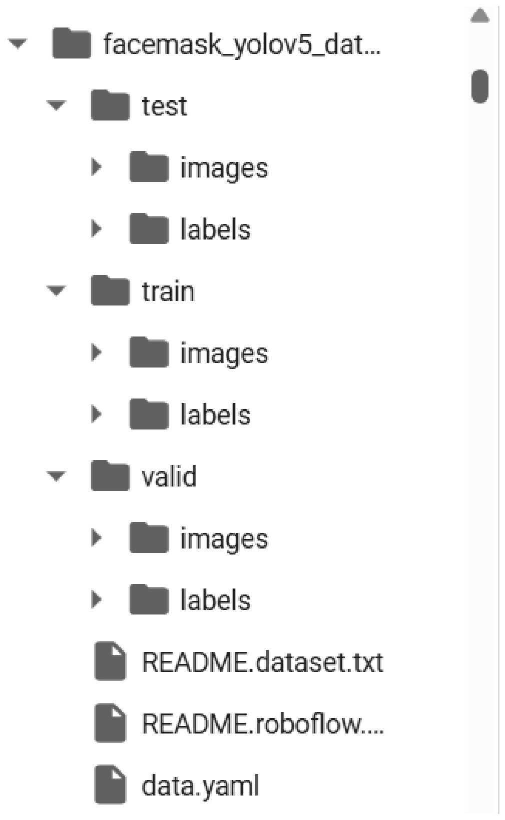


```
## importing required libraries
import os
import shutil
import random
```



```
[ ] ## connecting to the google drive

    from google.colab import drive
    drive.mount('/content/drive')
```

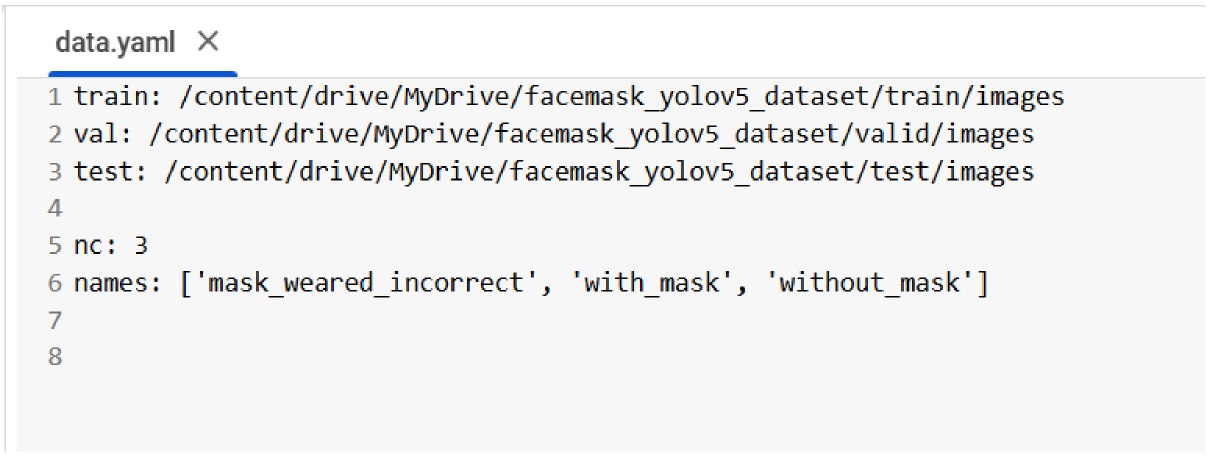
Step 2 — Upload the Dataset to Google drive and it should appear in the Following Tree Structure:



```
facemask_yolov5_dat...
    test
        images
        labels
    train
        images
        labels
    valid
        images
        labels
    README.dataset.txt
    README.roboflow....
    data.yaml
```

Step 3 — Please modify the data.yaml by replacing train and test paths with correct paths:

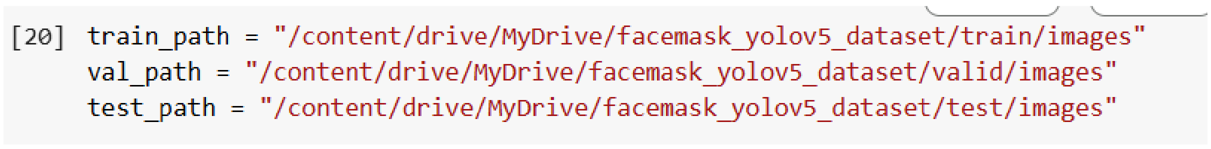
(a) Adjust the correct labels of your classes and specify the correct number of classes according to your dataset



data.yaml

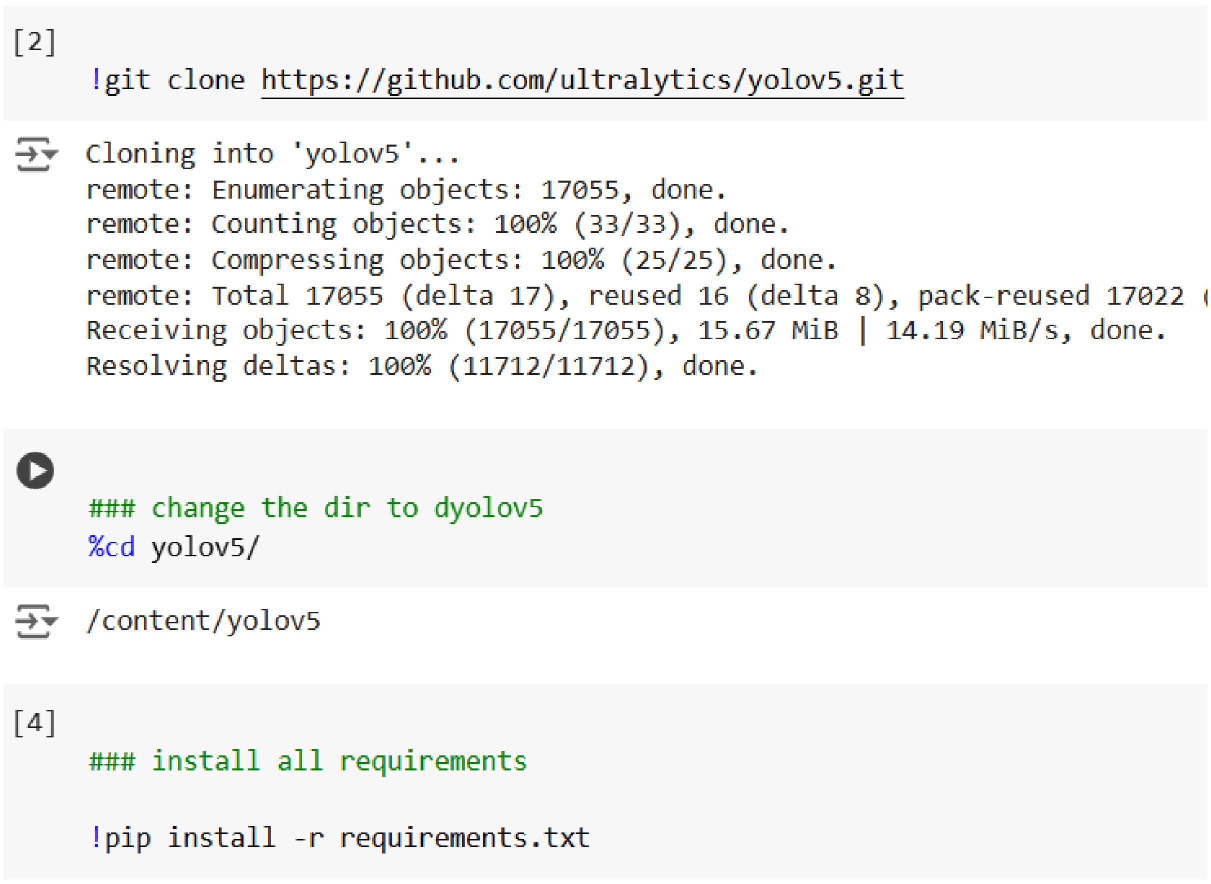
```
train: /content/drive/MyDrive/facemask_yolov5_dataset/train/images
val: /content/drive/MyDrive/facemask_yolov5_dataset/valid/images
test: /content/drive/MyDrive/facemask_yolov5_dataset/test/images

nc: 3
names: ['mask_weared_incorrect', 'with_mask', 'without_mask']


```

Step 4 — Specify the Correct train and test paths, Please cross check the Paths to avoid further errors:



```
[20] train_path = "/content/drive/MyDrive/facemask_yolov5_dataset/train/images"
     val_path = "/content/drive/MyDrive/facemask_yolov5_dataset/valid/images"
     test_path = "/content/drive/MyDrive/facemask_yolov5_dataset/test/images"
```

Step 5 —Next install the pretrained model from github (Ultralytics) and change the directory to yolov5 and pip install the requirements.txt to avoid dependency issues:



```
[2]
     !git clone https://github.com/ultralytics/yolov5.git
```

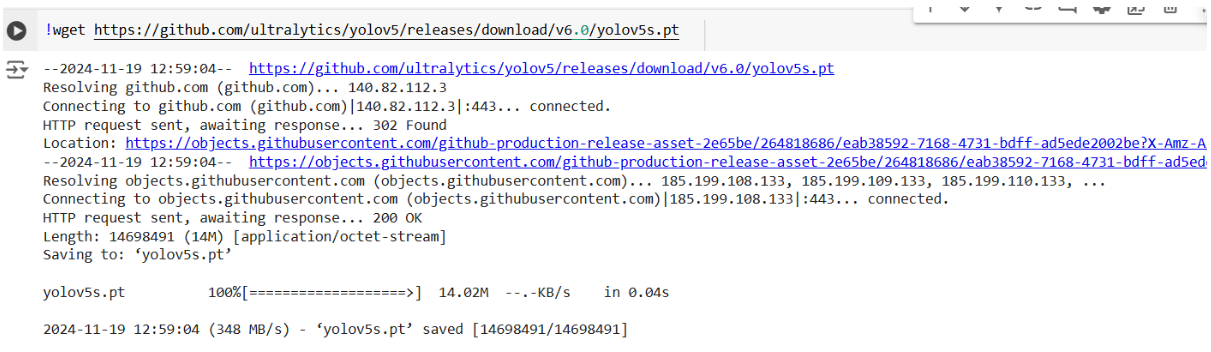
```
Cloning into 'yolov5'...
remote: Enumerating objects: 17055, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 17055 (delta 17), reused 16 (delta 8), pack-reused 17022 (
Receiving objects: 100% (17055/17055), 15.67 MiB | 14.19 MiB/s, done.
Resolving deltas: 100% (11712/11712), done.
```

```
### change the dir to dyolov5
%cd yolov5/
```

```
/content/yolov5
```

```
[4]
     ### install all requirements

     !pip install -r requirements.txt
```

Step 6 — install the desired yolov5 model according to your need e.g we are using yolov5s.pt model:



```
!wget https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s.pt
```

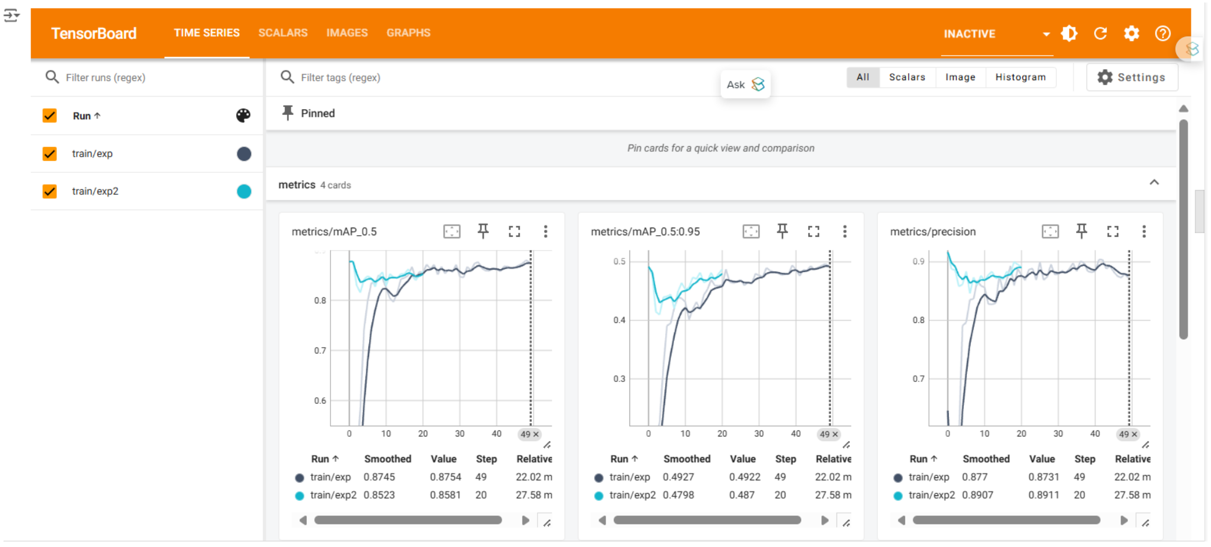
```
--2024-11-19 12:59:04--  https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s.pt
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/eab38592-7168-4731-bdff-ad5ede2002be?X-Amz-A
--2024-11-19 12:59:04--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/eab38592-7168-4731-bdff-ad5ed
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14698491 (14M) [application/octet-stream]
Saving to: ‘yolov5s.pt’

yolov5s.pt          100%[===================>]  14.02M  --.-KB/s    in 0.04s

2024-11-19 12:59:04 (348 MB/s) - ‘yolov5s.pt’ saved [14698491/14698491]
```

Note: to install yolov5l just replace yolov5s.pt with yolov5l.pt when downloading

Step 7 — Next first run the Tensorboard to evaluate the results and training statistics of your model during training. Below is the Command for it and the output is also shown:

```
[ ]  %load_ext tensorboard
     %tensorboard --logdir runs
```

(a) OUTPUTS: (following output will only show after the training has started by running the command from the next step)



metrics/mAP_0.5

Run ↑	Smoothed	Value	Step	Relative
● train/exp	0.8745	0.8754	49	22.02 m
● train/exp2	0.8523	0.8581	20	27.58 m

metrics/mAP_0.5:0.95

Run ↑	Smoothed	Value	Step	Relative
● train/exp	0.4927	0.4922	49	22.02 m
● train/exp2	0.4798	0.487	20	27.58 m

metrics/precision

Run ↑	Smoothed	Value	Step	Relative
● train/exp	0.877	0.8731	49	22.02 m
● train/exp2	0.8907	0.8911	20	27.58 m

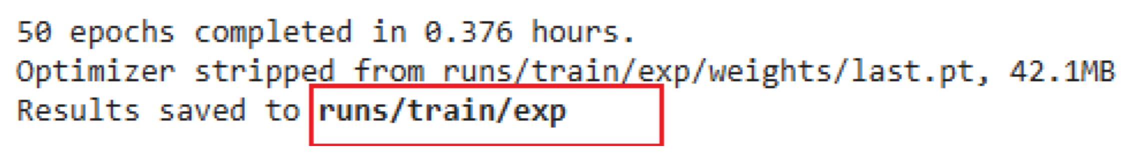
Step 8 — Now run the train command

```
!python train.py --img 416 --batch 8 --epochs 40 --data /content/drive/MyDrive/facemask_yolov5_dataset/data.yaml
--weights /content/yolov5/yolov5/yolov5s.pt --nosave –cache
```

For yolov5l use the following command :

```
!python train.py --img 416 --batch 8 --epochs 40 --data /content/drive/MyDrive/facemask_yolov5_dataset/data.yaml
--weights /content/yolov5/yolov5l.pt --nosave --cache
```

Step 9 — after Training has concluded, the saved weights of the model will show in the following folder:



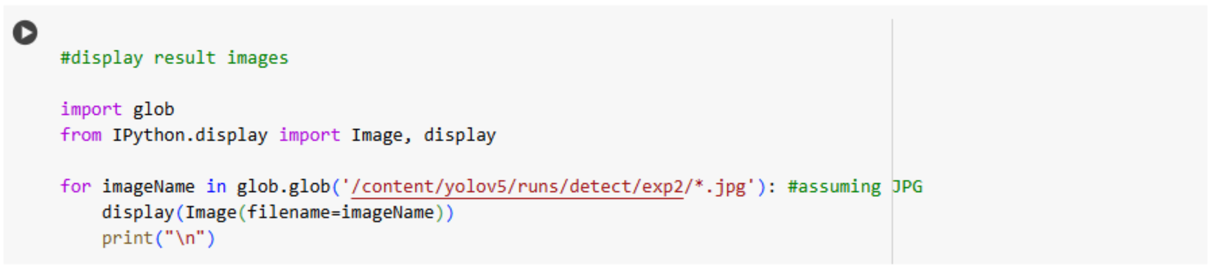
```
50 epochs completed in 0.376 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 42.1MB
Results saved to runs/train/exp
```

Step 10 — now run inference on the test data by using the previous weights (*i.e last.pt*) saved after training:

```
!python detect.py --weights /content/yolov5/yolov5/yolov5/runs/train/exp/weights/last.pt --img 416 --conf 0.4 --
source /content/drive/MyDrive/facemask_yolov5_dataset/test/images
```

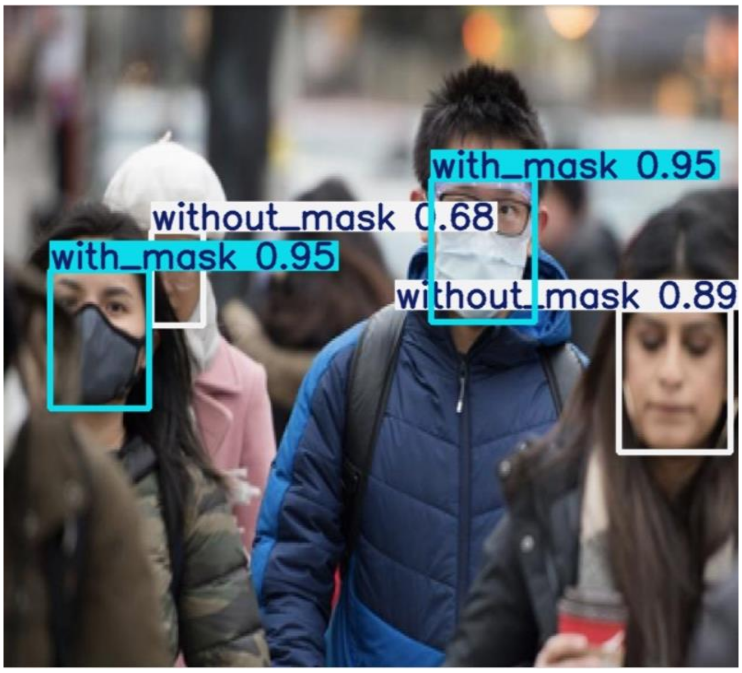
Note : specify the correct paths to your weights and the test folder, in your case it could be different

Step 11 — Now display the Image Results by running the following command (please change the exp2 to exp1 if you have done the training once... training again will save it in exp2 folder):



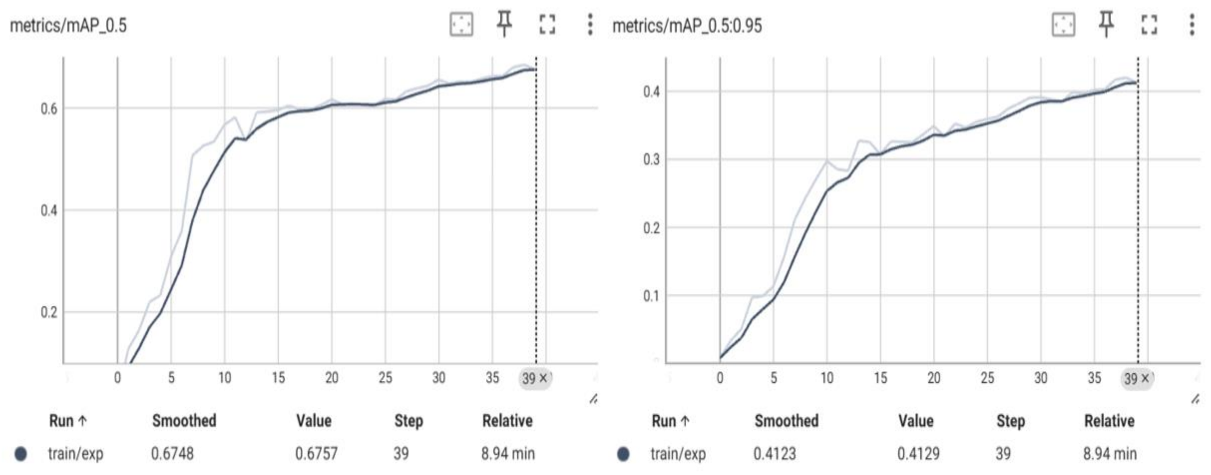
```
#display result images

import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp2/*.jpg'): #assuming JPG
    display(Image(filename=imageName))
    print("\n")
```

<u>SOME TEST RESULTS:</u>



Step 12 — Following is the mAP score comparisons for the yolov5s and yolov5l models after 40 epochs on the training dataset:

(a) Yolov5s

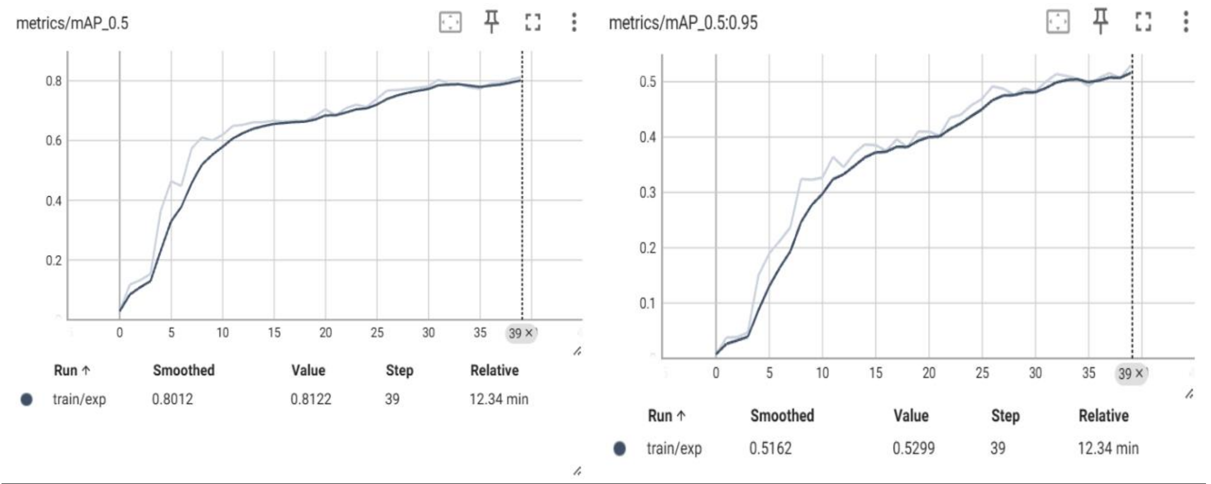


Run ↑	Smoothed	Value	Step	Relative
● train/exp	0.6748	0.6757	39	8.94 min

Run ↑	Smoothed	Value	Step	Relative
● train/exp	0.4123	0.4129	39	8.94 min

mAP score = 0.6757

(b) Yolov5l



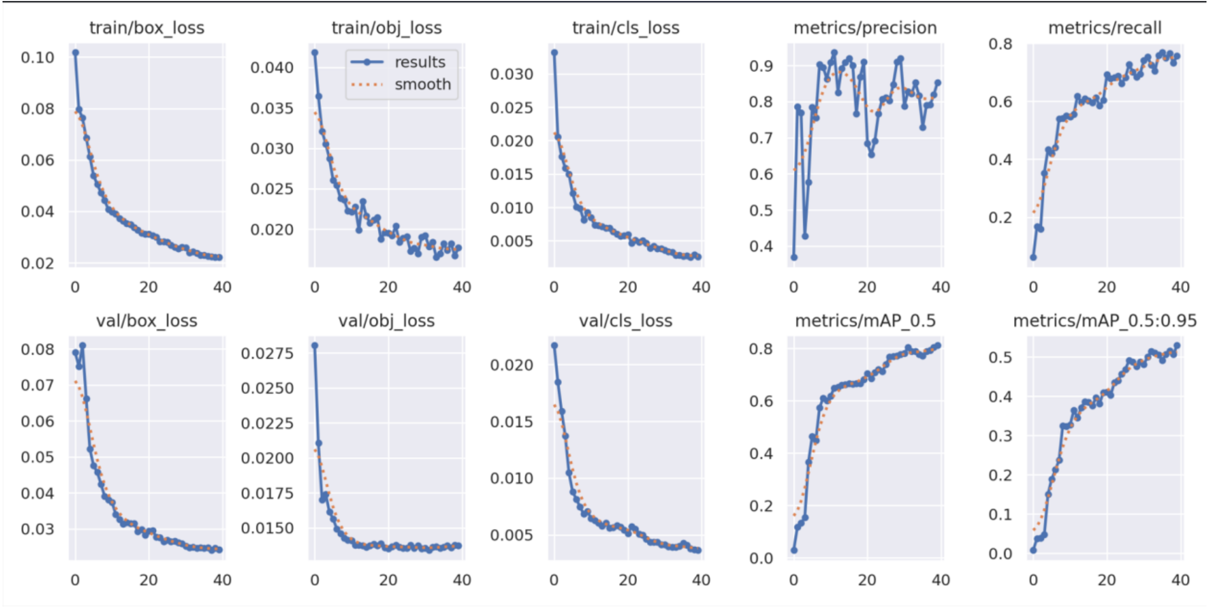
Run ↑	Smoothed	Value	Step	Relative
● train/exp	0.8012	0.8122	39	12.34 min

Run ↑	Smoothed	Value	Step	Relative
● train/exp	0.5162	0.5299	39	12.34 min

mAP score = 0.8122

Conclusion: clearly the yolov5l model wins in this case with an mAP score of 0.8122 since yolov5l has more layers

Loss curves for yolov5l:


train/box_loss
train/obj_loss
train/cls_loss
metrics/precision
metrics/recall
results
smooth
val/box_loss
val/obj_loss
val/cls_loss
metrics/mAP_0.5
metrics/mAP_0.5:0.95