Configuration Manual

MSc Research Project
Masters in Artificial Intelligence (MScAI JAN24)

Rohit PrasannaKumar
Student ID: 23133872

School of Computing
National College of Ireland

Supervisor: Paul Stynes

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rohit Prasanna Kumar		
Student ID:	23133872		
Programme:	Masters in Artificial Intelligence (MScAIJAN24)	**Year:**	2024
Module:	Practicum		
Lecturer:	Paul Stynes		
Submission Due Date:	15/12/2024		
Project Title:	A Deep Learning Filtration Framework to Eliminate Not Safe For Work content in Digital media.		
Word Count:	1111 **Page Count:** 9		

Signature: Rohit PrasannaKumar

Date: 15/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rohit PrasannaKumar
Student ID:23133872

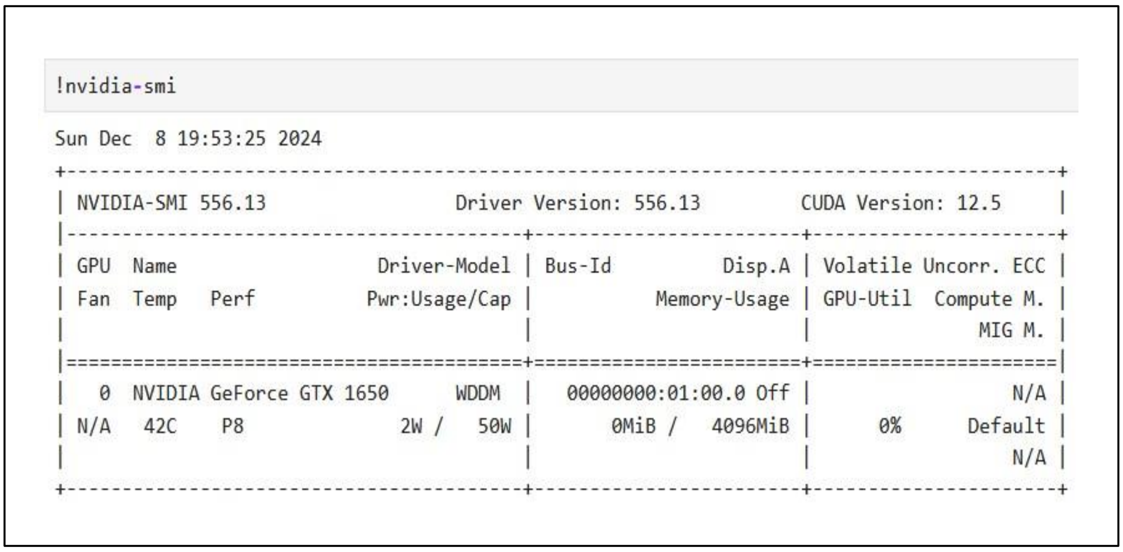
1 Introduction:

This manual provides the information on the requirements and environment setup required to reproduce the project ***"A Deep Learning Filtration Framework to Eliminate Not Safe for Work content in Digital media."*** To ensure the proper replication of the environment it would be requested to at least follow the minimum requirements and associated guidelines.

2 System Specifications:

2.1 Hardware Specifications:

Operating System: Windows-10-10.0.19045-SP0 64 bit
Processor: AMD Ryzen 5 4500 H
Threads: 12
Cores: 6
RAM: 16 GB
GPU: NVIDIA GeForce GTX 1650, 4096MiB With CUDA Compute Compatibility
Storage: 500 GB SSD (OS), 1 TB HDD.



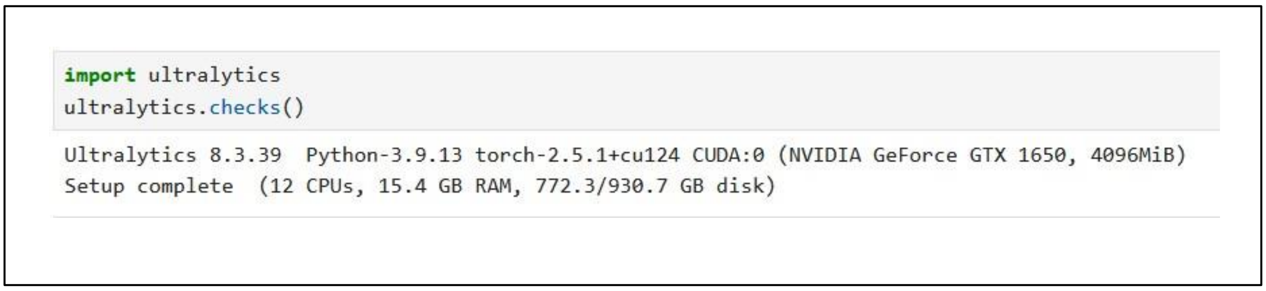
```
!nvidia-smi

Sun Dec  8 19:53:25 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 556.13                 Driver Version: 556.13         CUDA Version: 12.5     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                  Driver-Model | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce GTX 1650      WDDM  |   00000000:01:00.0 Off |                  N/A |
| N/A   42C    P8              2W /   50W |       0MiB /   4096MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

Fig.1: Nvidia System Management Interface

2.2 Software Specifications:

2.2.1 Programming Environment: Python version: Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)]



```
import ultralytics
ultralytics.checks()
```

```
Ultralytics 8.3.39  Python-3.9.13 torch-2.5.1+cu124 CUDA:0 (NVIDIA GeForce GTX 1650, 4096MiB)
Setup complete  (12 CPUs, 15.4 GB RAM, 772.3/930.7 GB disk)
```

Fig. 2: Python version check and hardware resource availability.

2.2.2 Libraries

Provided here are the list of libraries which are crucial to running the project.

1. numpy==1.26.4
2. opencv-contrib-python==4.10.0.84
3. opencv-python==4.10.0.84
4. opencv-python-headless==4.10.0.84
5. pandas==2.2.2
6. pickleshare==0.7.5
7. pillow==10.4.0
8. tensorboard==2.10.1
9. tensorboard-data-server==0.6.1
10. tensorboard-plugin-wit==1.8.1
11. tensorflow==2.10.1
12. tensorflow-estimator==2.10.0
13. tensorflow-hub==0.16.1
14. tensorflow-io-gcs-filesystem==0.31.0
15. tensorflow-text==2.10.0
16. tf-keras==2.15.0
17. torch==2.5.1+cu124
18. torchaudio==2.5.1+cu124
19. torchvision==0.20.1+cu124
20. tqdm==4.66.5
21. ultralytics==8.3.39
22. ultralytics-thop==2.0.10

The miscellaneous set of requirements will be provided in the requirements.txt file which would contain all the libraries. It is necessary as this experiment setup was run in a custom jupyter notebook which was accessible from Microsoft PowerShell. Ultralytics (Jocher, G. and Qiu, J. (2024)) is a necessary component for YOLO. So ensure proper installation of ultralytics and ensure it is working with pytorch-gpu.

2.2.3 Dependencies

Additionally, few of the libraries require custom configuration and installation of some associated software and setting up path in environment for the same. These would include.

1. Python [path set in environment variables]
2. CUDA ToolKit v11.4* [path set in environment variables]
3. cuDNN* (Nvidia cuDNN Documentation (2024)) [path set in environment variables]
4. ffmpeg (FFmpeg Documentation (2024))[path set in environment variables]
5. OpenCV (OpenCV Documentation (2024))

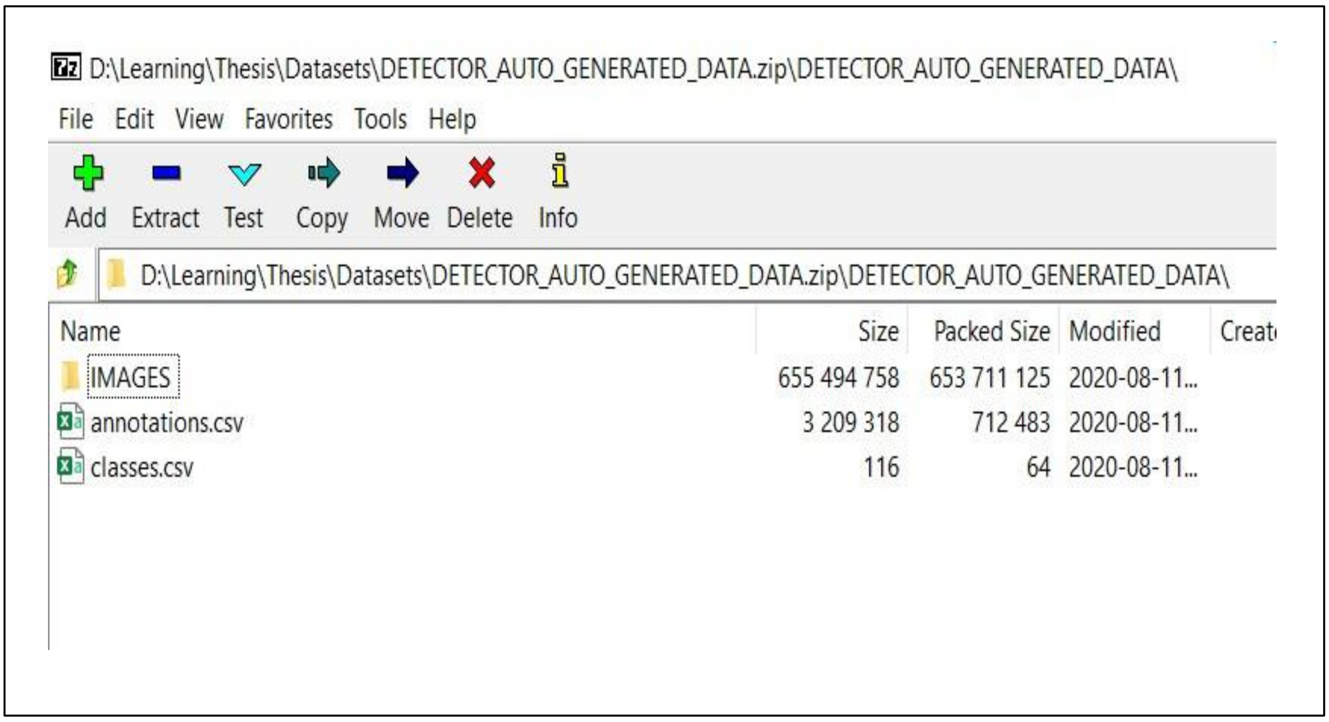
*The CUDA components listed here are based on the GPU specification of Nvidia GeForce GTX 1650 devices. It may defer based on the architecture of the device it would be replicated in future. It would be advised to kindly go through NVIDIA CUDA documentation to meet the requirements of the CUDA for that system.

3 Datasets

There are two datasets at use here namely ***NudeNet_classifier_dataset_v1*** (Bedapudi.P, (2019)) the one used for training the model while the other dataset ***LSPD dataset*** (Duy, et al (2022)) was used for testing purposes. The dataset used for training can be accessed from archive.org. While the latter is only available based on request.

3.1. Dataset Configuration

The dataset made available by NudeNet_classifier_dataset_v1 contains an IMAGE FOLDER and two associated csv files which provides the class names for the annotations. Refer Fig.3.

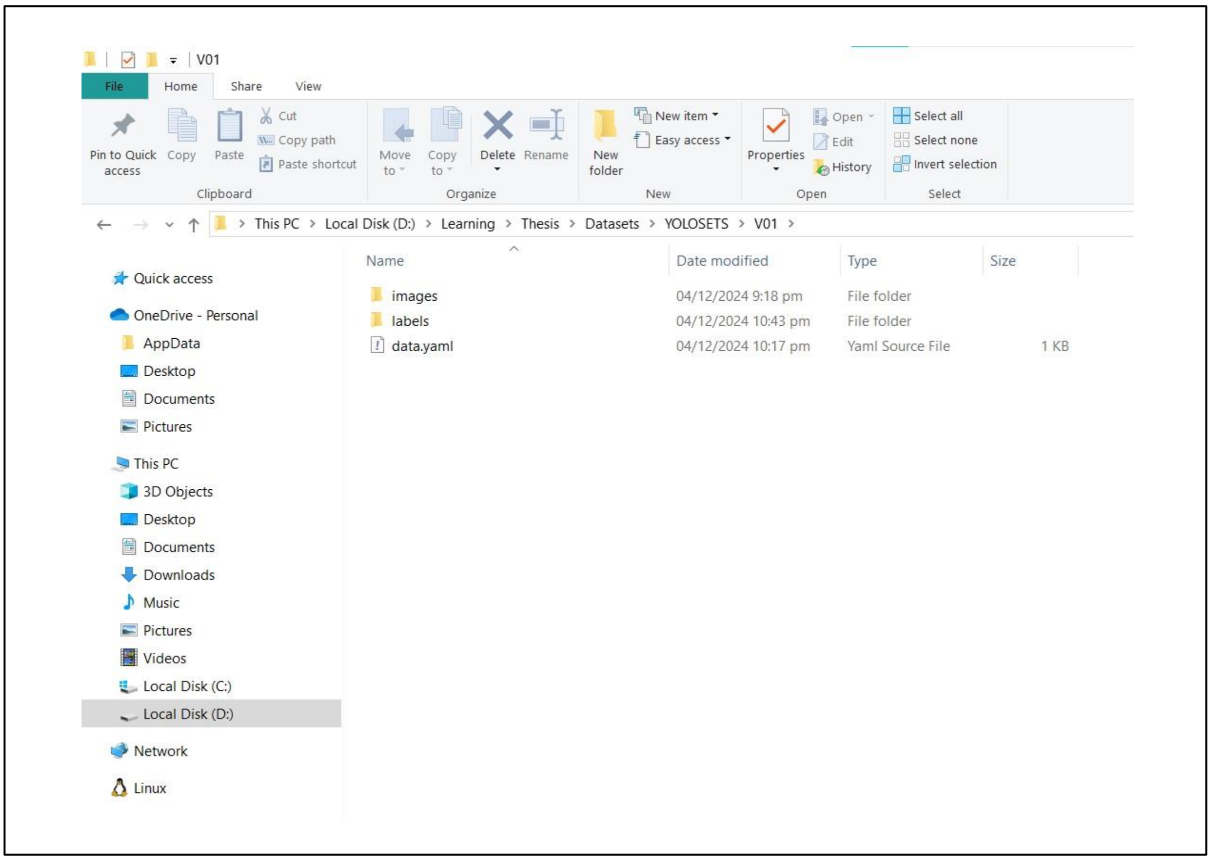

D:\Learning\Thesis\Datasets\DETECTOR_AUTO_GENERATED_DATA.zip\DETECTOR_AUTO_GENERATED_DATA\
File Edit View Favorites Tools Help
Add Extract Test Copy Move Delete Info

D:\Learning\Thesis\Datasets\DETECTOR_AUTO_GENERATED_DATA.zip\DETECTOR_AUTO_GENERATED_DATA\

Name	Size	Packed Size	Modified	Creat
IMAGES	655 494 758	653 711 125	2020-08-11...	
annotations.csv	3 209 318	712 483	2020-08-11...	
classes.csv	116	64	2020-08-11...	

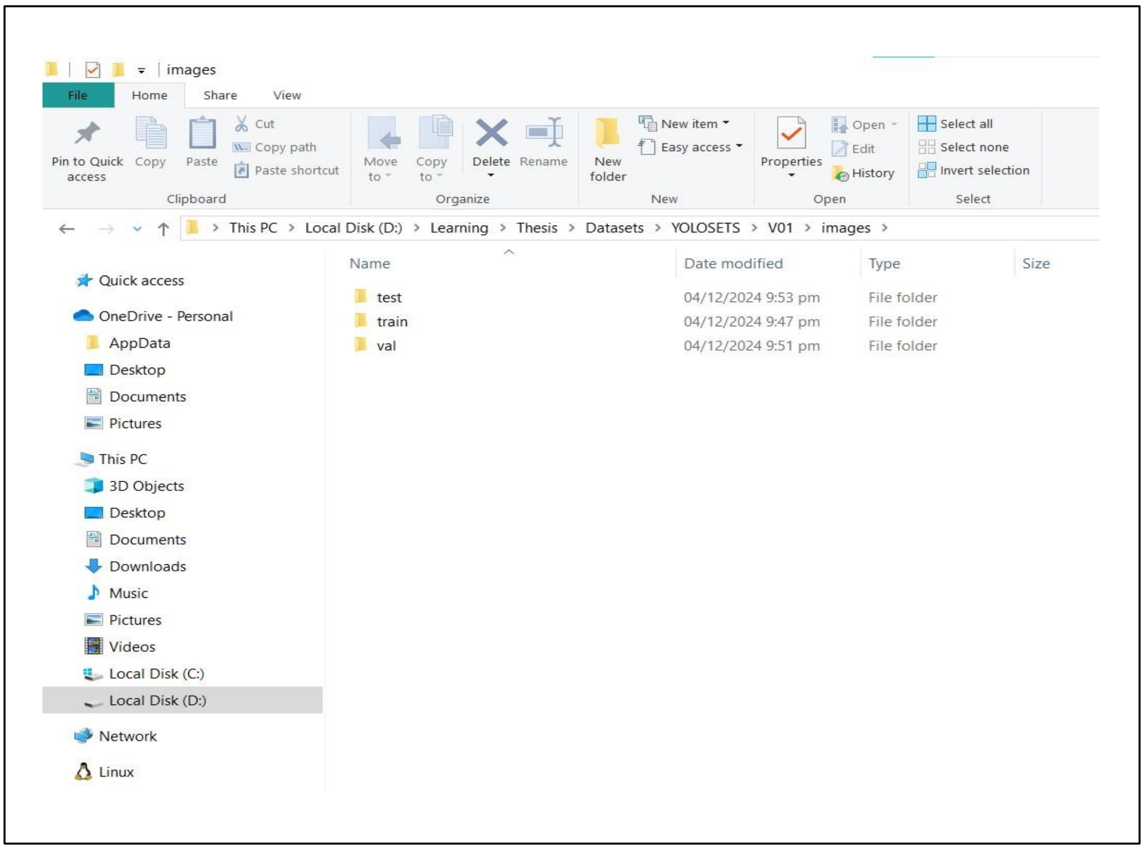
Fig.3: Dataset Archive

It is to be ensured that it follows the YOLO Dataset structure for the model to train and evaluate the data successfully. Refer Fig.4 for how dataset should be structured, Refer Fig. 5: for how train, test and validation data should be stored. Fig. 6: For how train, test and validation annotations are stored. The cache files visible in Fig.6 would only be generated during the first run in the ultralytics framework when loaded alongside a yolo model. The python scripts regarding the sanitization, label validation and label checking would be provided alongside the code.zip. It would help generate a folder based on yolo requirements for the project.



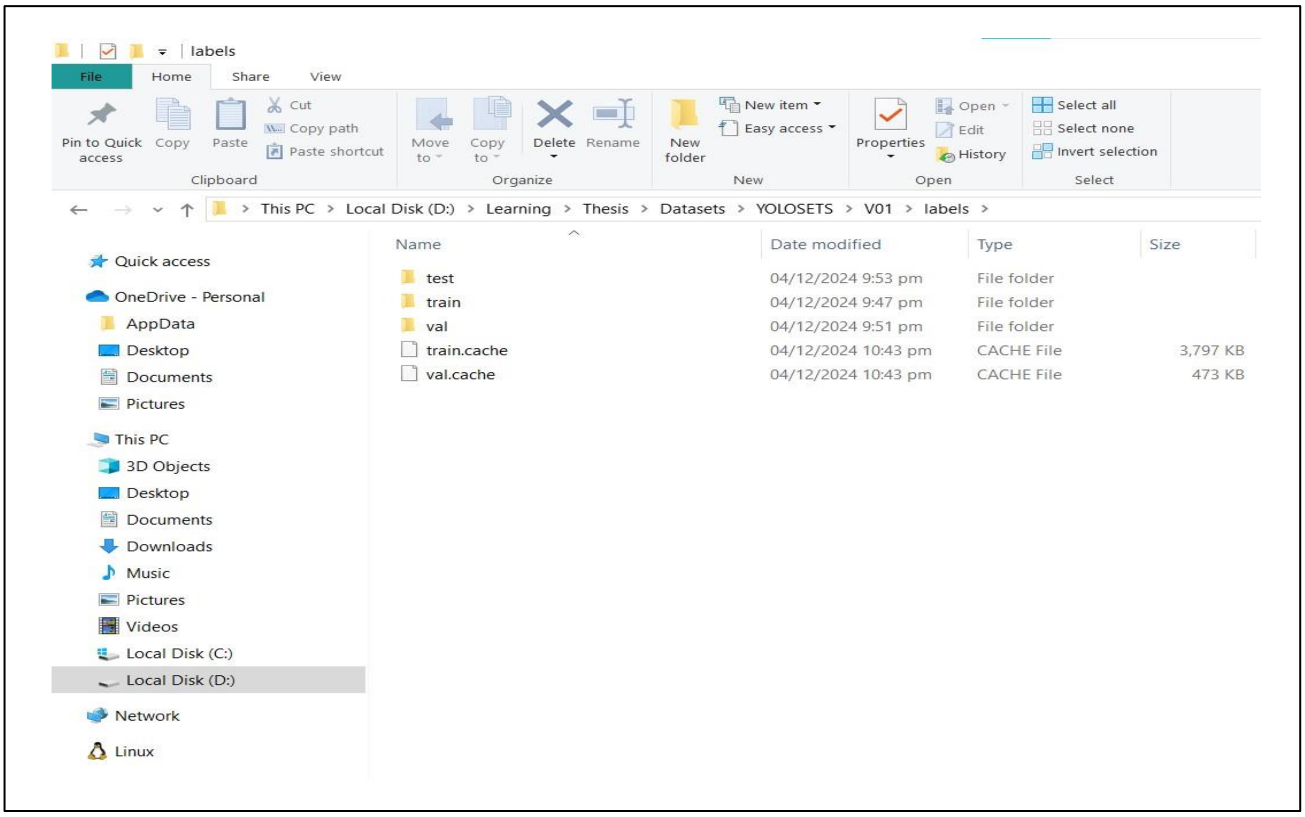
Name	Date modified	Type	Size
images	04/12/2024 9:18 pm	File folder	
labels	04/12/2024 10:43 pm	File folder	
data.yaml	04/12/2024 10:17 pm	Yaml Source File	1 KB

Fig.4: Dataset should be divided into images and labels alongside a yaml file.



Name	Date modified	Type	Size
test	04/12/2024 9:53 pm	File folder	
train	04/12/2024 9:47 pm	File folder	
val	04/12/2024 9:51 pm	File folder	

Fig 5: The images dataset should be further divided as test, train, and validation files.



This PC > Local Disk (D:) > Learning > Thesis > Datasets > YOLOSETS > V01 > labels >

Name	Date modified	Type	Size
test	04/12/2024 9:53 pm	File folder	
train	04/12/2024 9:47 pm	File folder	
val	04/12/2024 9:51 pm	File folder	
train.cache	04/12/2024 10:43 pm	CACHE File	3,797 KB
val.cache	04/12/2024 10:43 pm	CACHE File	473 KB

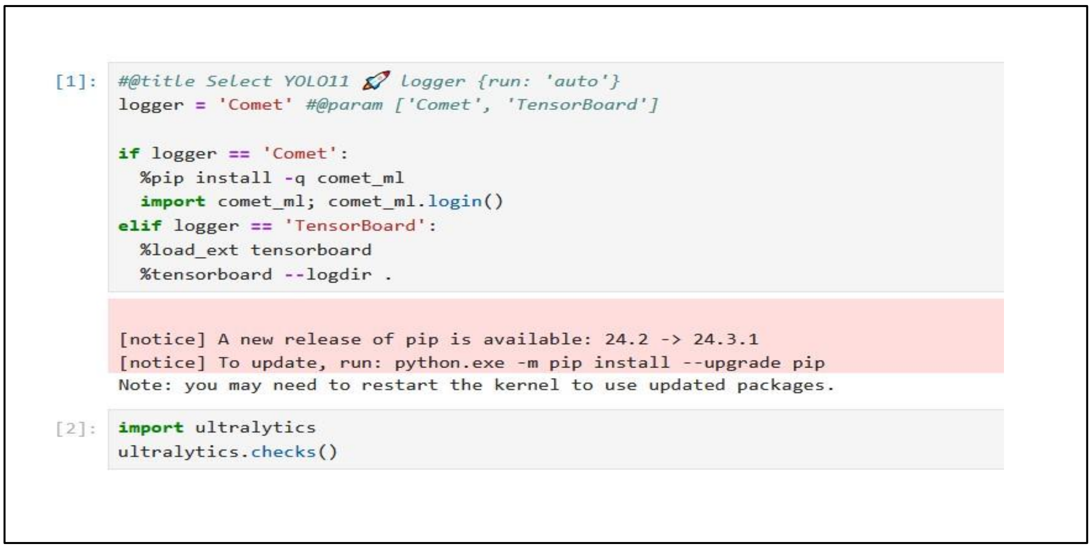
Fig.6: Train, Test and validation annotations are stored.

4 Analysis API

Comet.ml API (Comet.ml Documentation (2024)) was integrated to provide real-time metrics on the training model which goes through the tracking system of comet.ml. To ensure the proper working of the API, an account creation would be required as it helps save the model and return to it without having to run the training model again by the link. And an API key might be required to access every run. But the environment this setup was run in, the comet.ml account information was already logged in. So, API key was not required to connect to the account. Ensure that the logging is activated before the loading of libraries to ensure proper updates on records. Fig.7 displays the order in which it is followed in jupyter. And after the training has completed a link would be provided alongside the result table. Clicking on the link and accessing the page would open the page with all the statistics. But an account login is required as previously mentioned this also ensures separate data statistics for every different run and not overwrite the previous information.

5 Methodology

To train the model it is required to ensure that the folder follows the YOLO format structure. Scripts are provided inside the zip file to ensure the same. Separate code blocks are provided inside the scripts to ensure the necessary tasks, some extra code bases which are part of it are only intended for testing purposes, reiterating that code blocks may break the training process. The initial process was to check the image integrity and ensure the training data is split into training, test, and validation. Once the scripts are run to ensure the same. Ensure the path is right during this process. As the current path is based on the system the experimental setup is run on. The amp=False in the code is ensured during training process as there is a conflict for the code in certain GPU's specifically 1650 series. As training without that parameter would not produce any boxes. Finally, access the link after training to access the training statistics.



```
[1]: #@title Select YOLO11 🚀 logger {run: 'auto'}
     logger = 'Comet' #@param ['Comet', 'TensorBoard']

     if logger == 'Comet':
       %pip install -q comet_ml
       import comet_ml; comet_ml.login()
     elif logger == 'TensorBoard':
       %load_ext tensorboard
       %tensorboard --logdir .

     [notice] A new release of pip is available: 24.2 -> 24.3.1
     [notice] To update, run: python.exe -m pip install --upgrade pip
     Note: you may need to restart the kernel to use updated packages.

[2]: import ultralytics
     ultralytics.checks()
```

Fig.7 Load order of the script

References

Comet.ml Documentation (2024). Experiment Management for Machine Learning.Available at: comet.ml

FFmpeg Documentation (2024). FFmpeg Multimedia Framework. Available at: ffmpeg.org.

Jocher, G. and Qiu, J. (2024) *Ultralytics YOLO11*. Available at: https://github.com/ultralytics/ultralytics.

Nvidia cuDNN Documentation (2024). NVIDIA CUDA® Deep Neural Network (cuDNN). Available at: docs.nvidia.com

OpenCV Documentation (2024). Open Source Computer Vision Library. Available at: opencv.org.