**Documentation – Object detection in images using deep learning (POC)**

Before beginning, create a virtual environment.

1. **Scraping**

web-scraping.ipynb file contains the scraping code. Three approaches have been used to scrap images. In the first cell is the Bing image downloader. In the next selenium scraps images. In the last cell beautiful soup is used but it only downloads 80 images. So, it is advised to use bing and selenium. Scrapped images should be in **'bmp', 'jpg', 'jpeg', 'png', 'tif', 'tiff', 'dng'** format as YOLOv5 supports only these formats.

a) The first cell contains the code which uses the bing image downloader

* Uncomment the line 1, 2 as they need to be installed before executing the code
* Line 4: Import the module
* Line 5: Provide the keywords in the list for the images we want to download. Example- ‘gas pipe on site images’
* Line 6-8: Iterate through each item in the list and download images in line 7. Limit defines the number of images we want to download. ‘**output\_dir**’ is the folder in which the subfolders will be created of each item in the list and will download images in them respectively

b) Next cell contains selenium script.

* Line 1-4: imports
* Line 6-13: make directories. At line 9, provide keyword to the ‘**search**’ variable of which images we want to download, example- ‘gas meter images’
* Line 15-27: Automate actions in chrome driver. At line 20 ‘**url**’ variable contains the link of google images page from where images will be searched and downloaded
* Line 29-48: The links of the images are stored in ‘**src**’ variable. At line 39 and 43, provide the number of images to download
* Line 50-54: Images will be downloaded from the links in the ‘**src**’ list

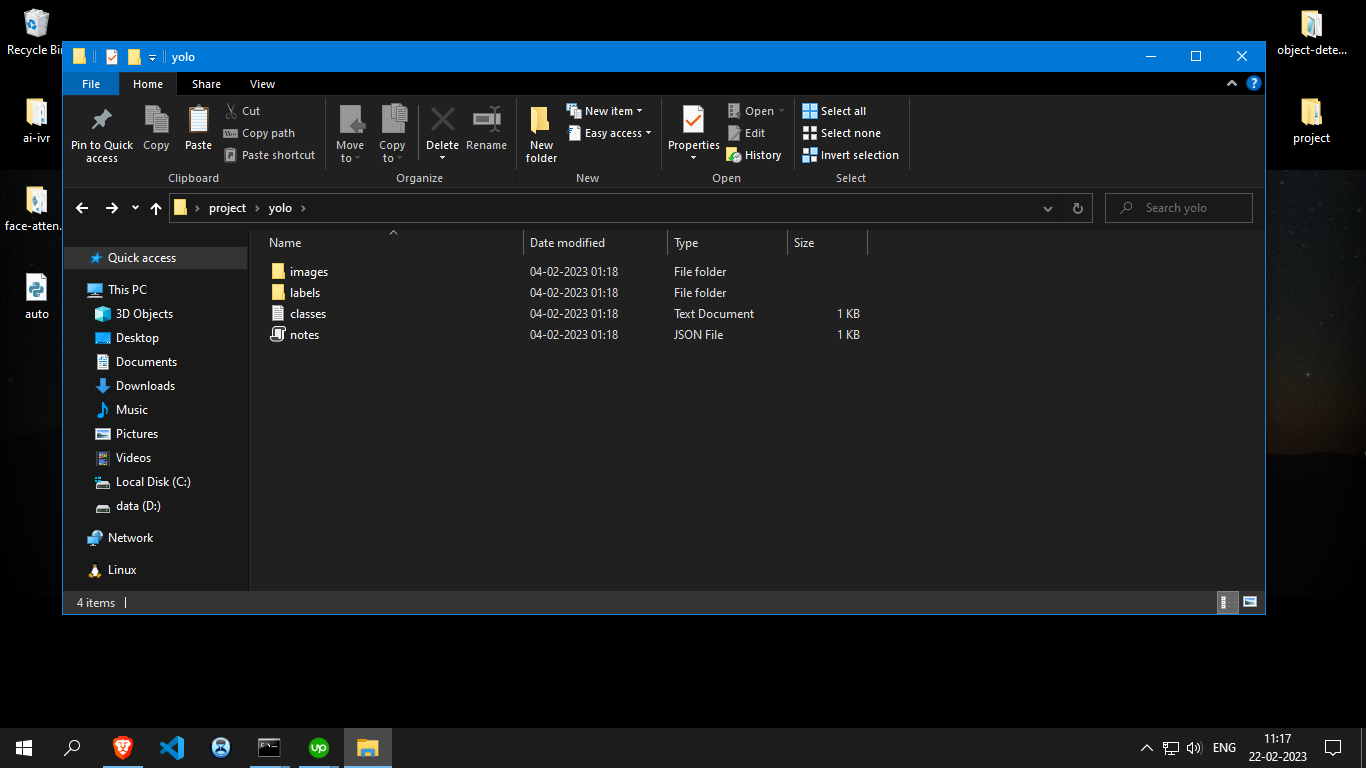
c) The last cell contains beautiful soup script

* At line 17, provide the keywords of the images to download in the ‘**list**’ variable
* At line 26, provide number of images to download to ‘**n\_images**’ variable
* The images will be downloaded in the subfolders of ‘**google-80**' directory

1. **Annotation**

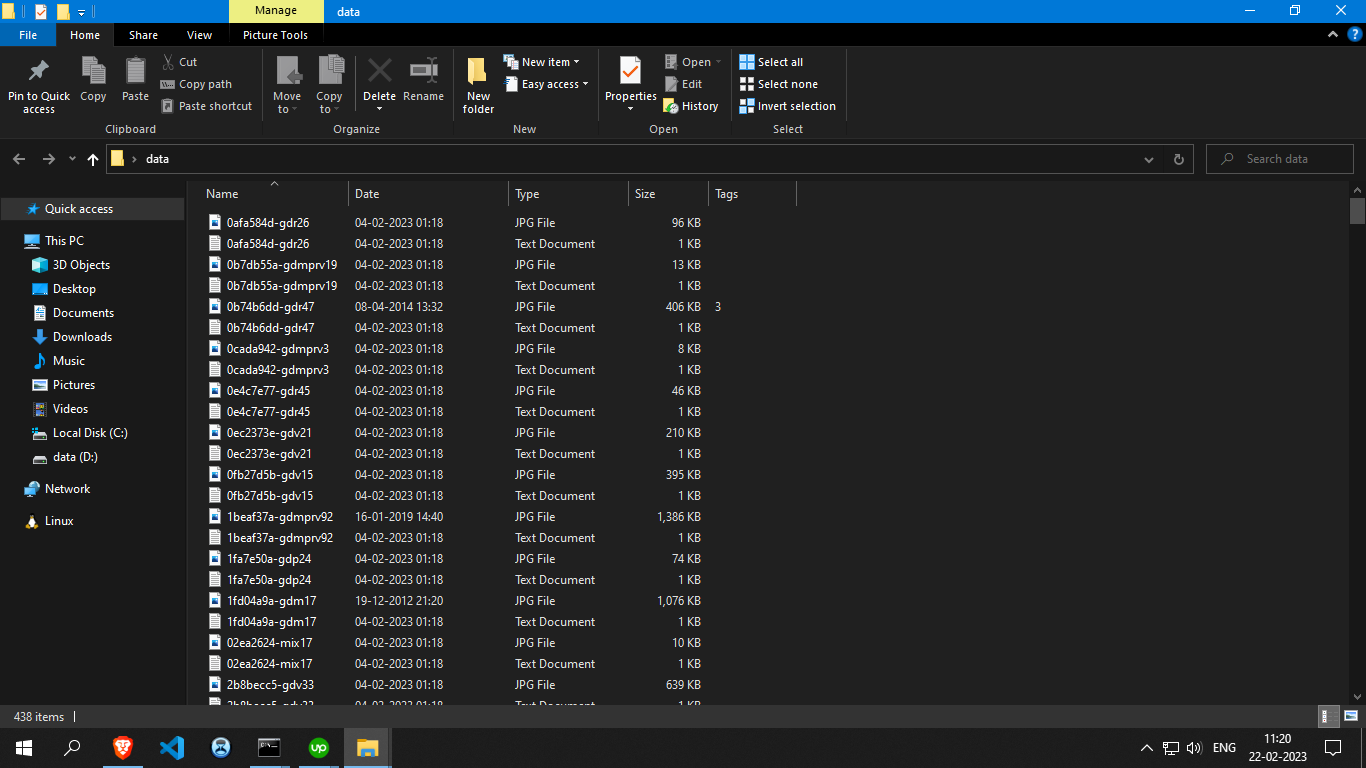
For annotation label studio tool was used.

* Install the label studio through command: **pip install label-studio**
* Launch label studio through command: **label-studio start**
* Click ‘**Create’** to make a new project. Assign it a name. Then import data through ‘**data import**’ option and upload your scraped images. In the labelling setup, under **computer vision** (on left) choose ‘**object detection with bounding boxes**’ option in our case.
* Under ‘**add labels names**’ write label names and click add. The color of the label can be changed by clicking the labels added
* One can configure settings accordingly if needed or leave as default and click ‘S**ave’**
* All images will be loaded and click the first image and annotate them by first clicking the label at the bottom and then draw bounding box around the object. Once done click ‘**Submit**’. To update bounding box after submitting one can make changes and click ‘**Update’**
* Once all images are annotated click the project name at the top. It’ll be looking like this: **Projects / {project name} / Labeling**
* Click ‘**Export**’ and choose in what format you want to export the annotations. We chose Yolo format.
* Once annotations are downloaded, extract the zip file and it will look like this:

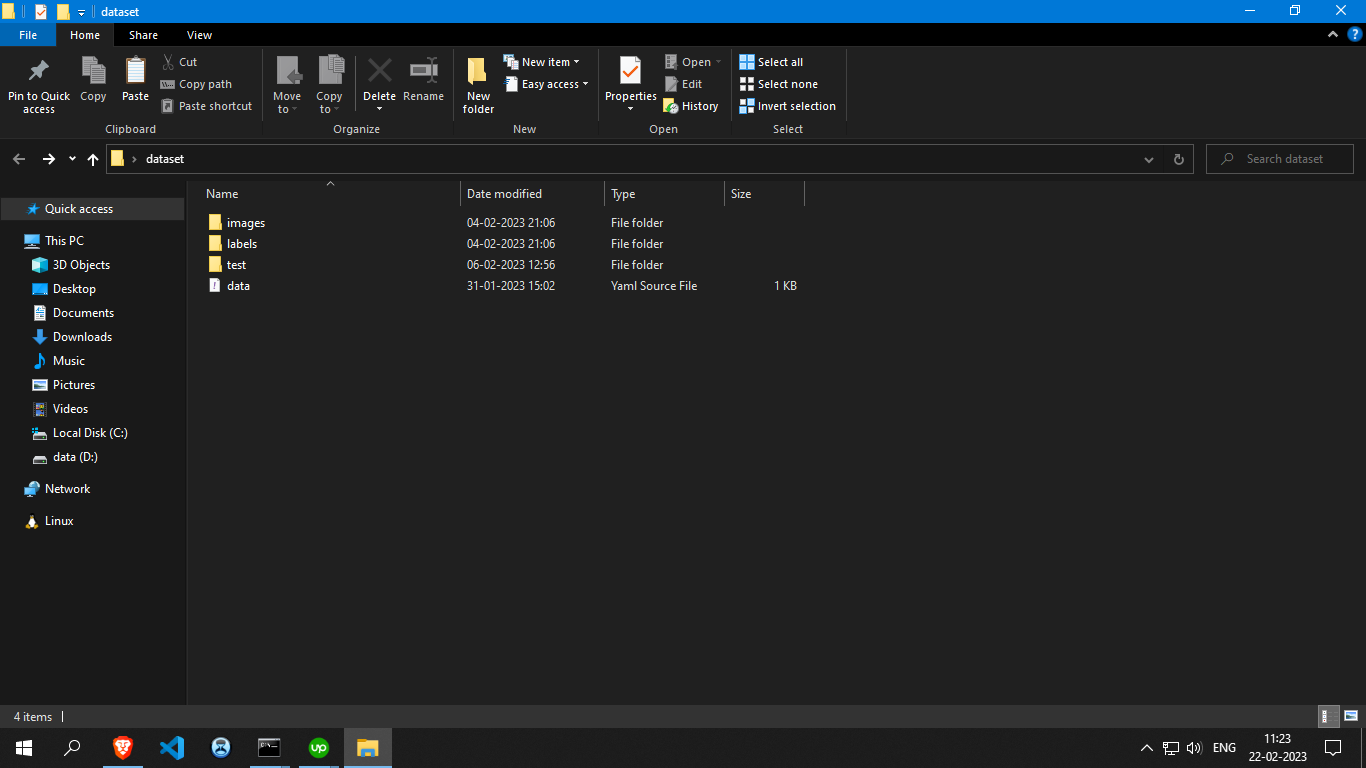


1. **Prepare the dataset**

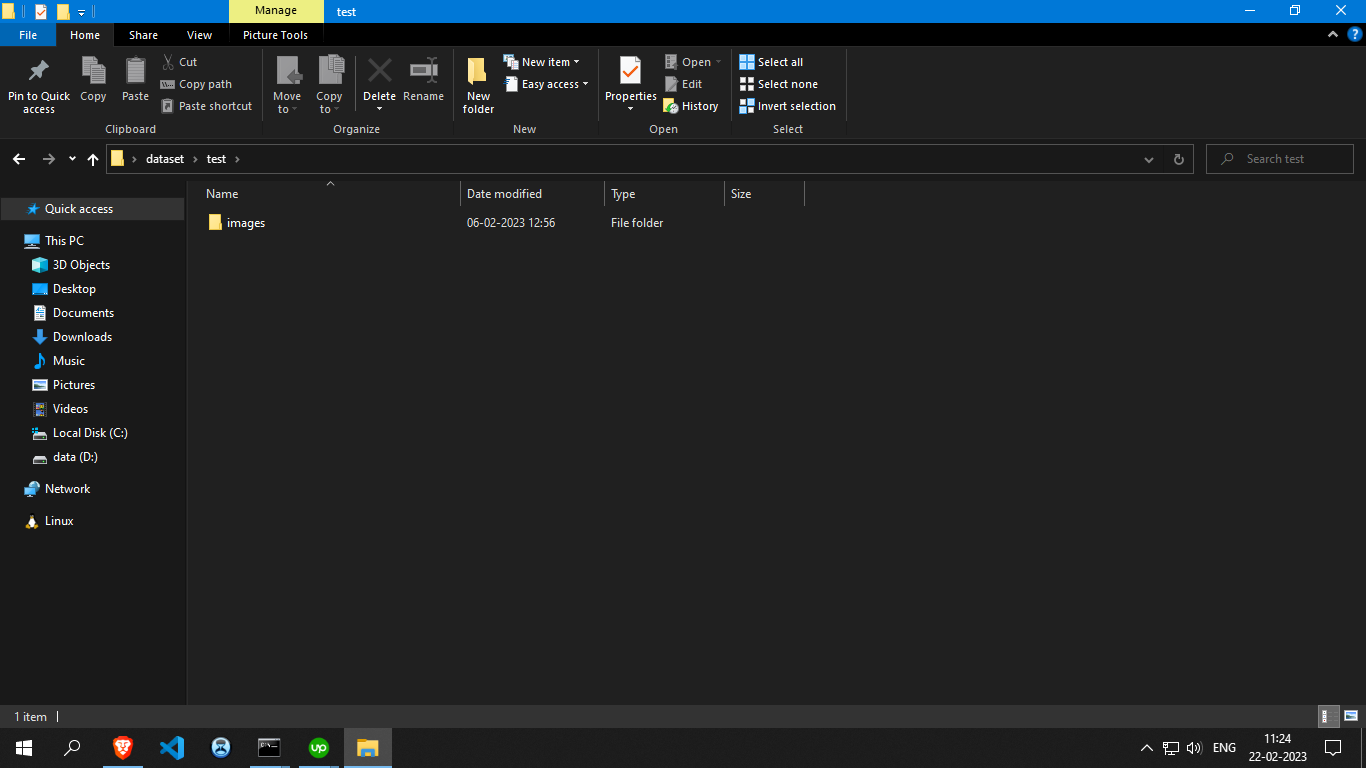
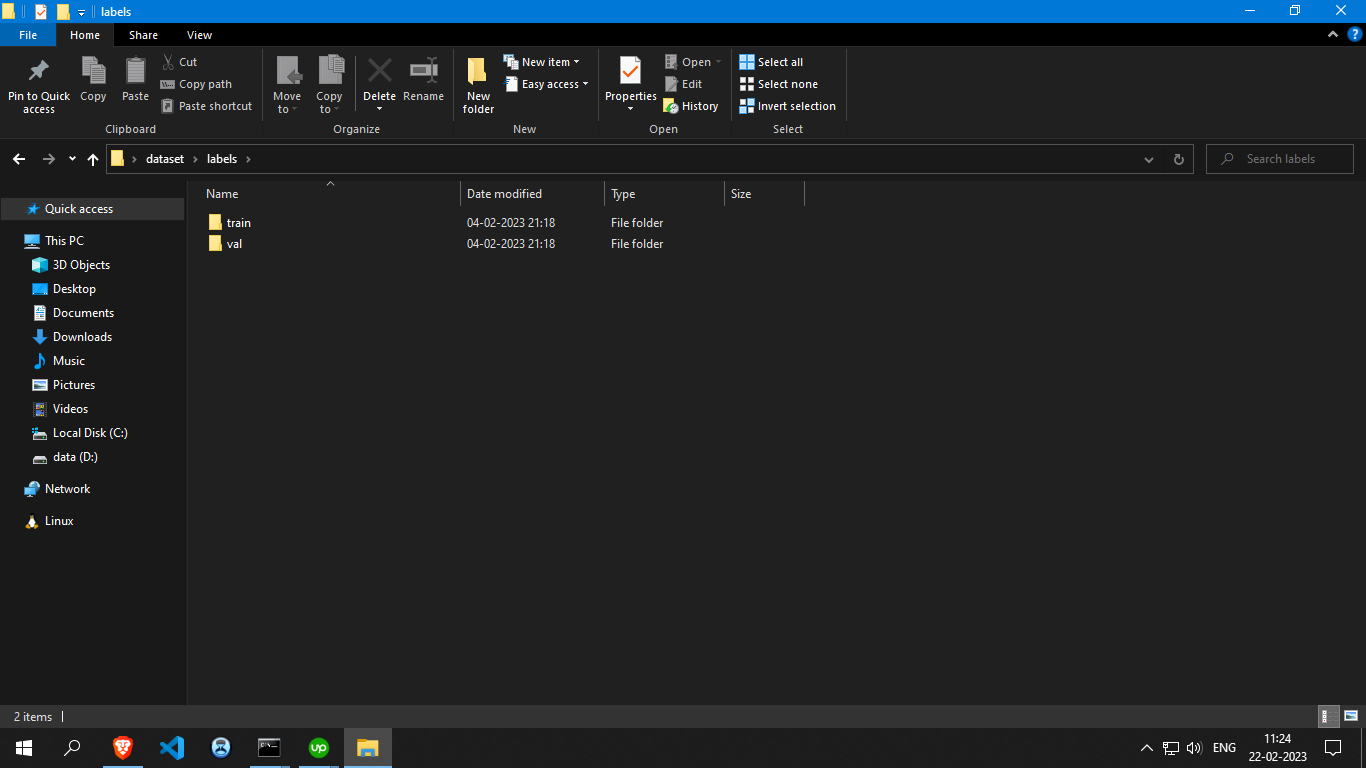
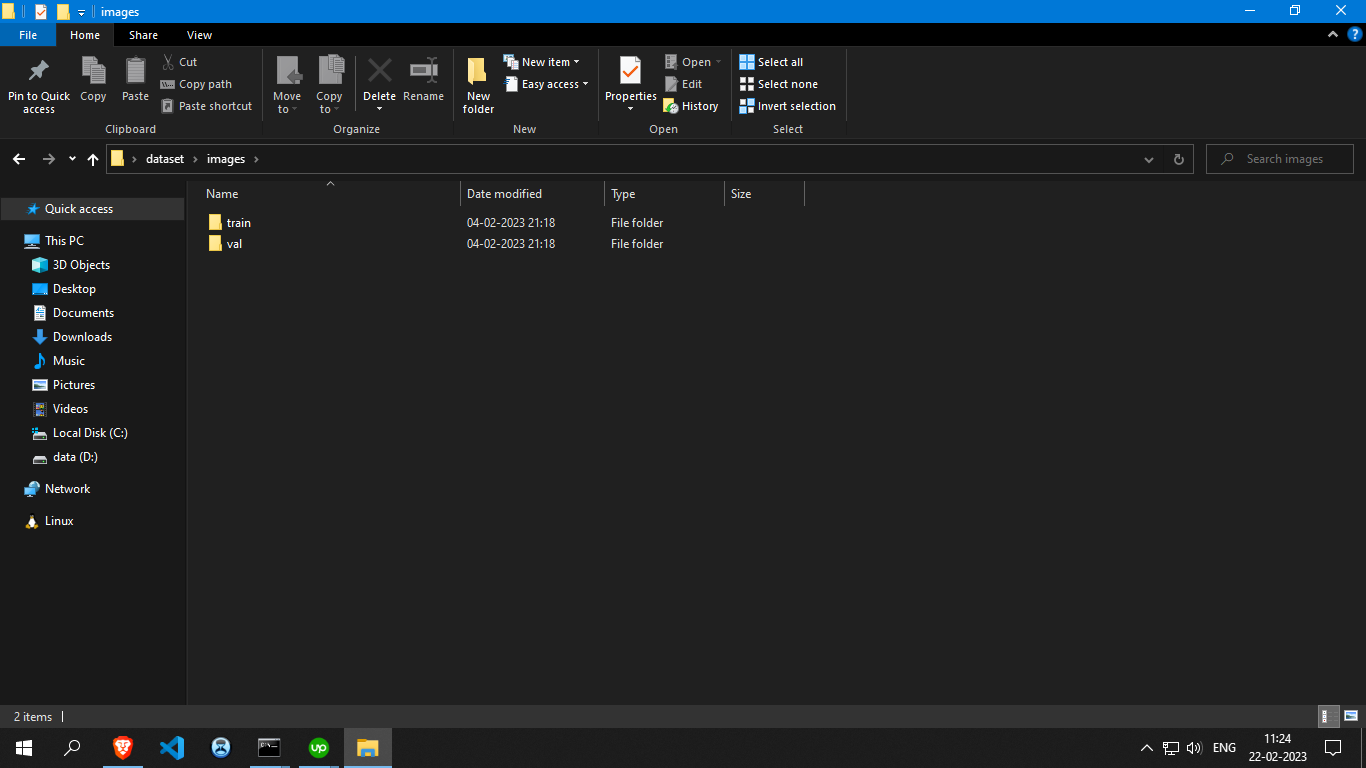
* Create a folder ‘data’
* Copy all the images and labels from the extracted ‘YOLO’ folder and paste it in the ‘data’ folder created. After pasting it will look like the following:



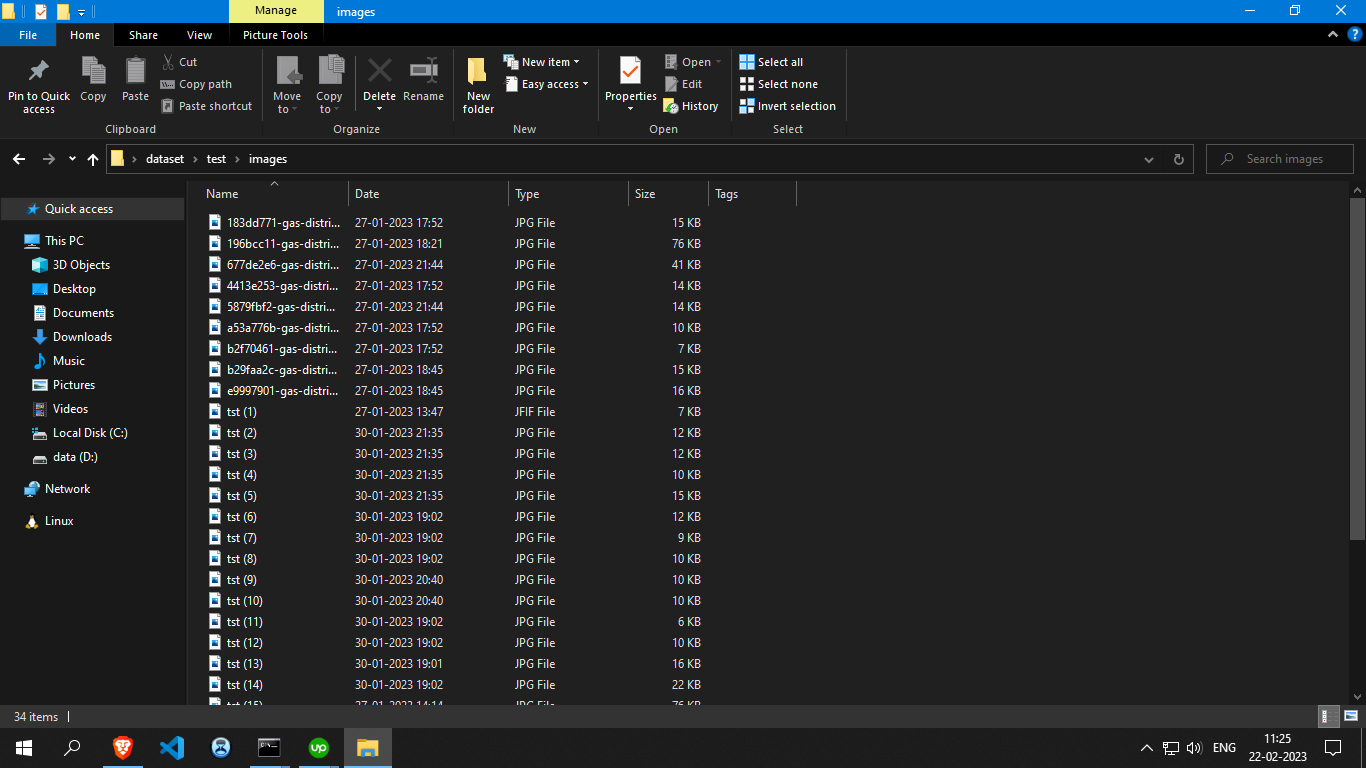
* Now create a folder ‘dataset’
* Create subfolder ‘images’, ‘labels’ and ‘test’



* Again, create subfolders ‘train’ and ‘val’ in both ‘images’ and ‘labels’ folders. Also create a subfolder ‘images’ in ‘test’ folder and place unseen images for testing purpose.



* Unseen test images:

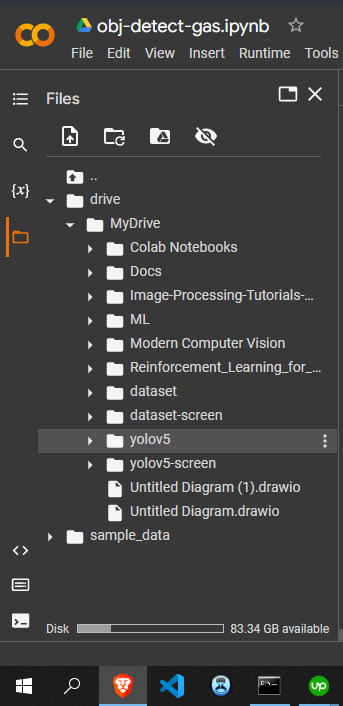


* Once all directories are set open the ‘train-val-split-files.ipynb’ file in jupyter notebook
* At line 10, 11, 12: pass the path of the data folder (containing images and their labels) in the ‘crsPath’ variable. From here the data will be split and copied in train and val folder in dataset folder
* Pass the path of the train and val folder in images folder in the dataset folder to ‘trainPath’ and ‘valPath’ variable
* Choose the train and validation ratio. Run the cell.
* In the next cell, pass the path of the train and val folders of both images and labels in ‘trainimagePath’, ‘trainlabelPath’, ‘valimagePath’, ‘vallabelPath’ variables respectively.
* Run the cell
* Our image and label data has been split into train and val folder
* We'll also require a data.yaml file which contains the path of our dataset and path of the train and val folders. It also contains the classes and their indexes
* Now our dataset is ready and upload it to the google drive.

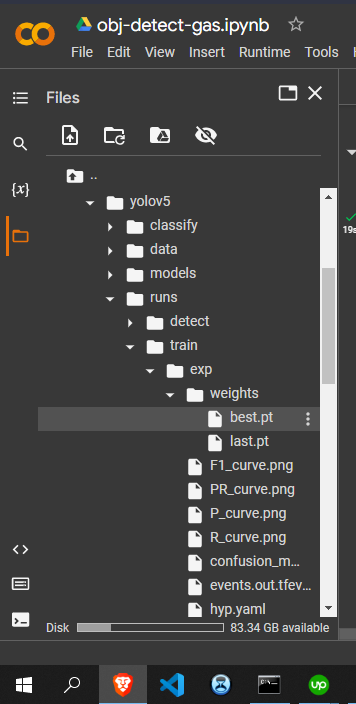
1. Custom training YOLOv5 model

For object detection we used YOLOv5. Open the object-detection-gas.ipynb in the google colab.

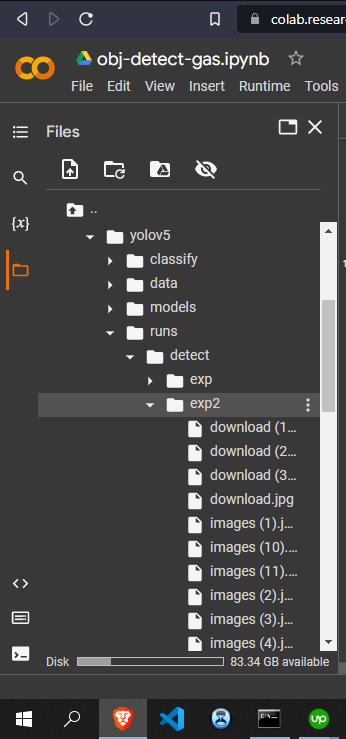
* Run the first cell. Here, connect with the google drive where our dataset is placed
* Once drive is mounted, change path to ‘**MyDrive**’ by running the next cell directory in which our model will be saved
* Run the next cell. Here clone the YOLOv5 repo and install all requirements. Once setup is done, we’ll train our model. The file structure should be like this:



* Once directories are set run the train.py cell to start training. We must pass the path of data.yaml file in the –data argument and in --weights pass the yolo pretrained model. We can also pass various arguments like batch size, epochs, img, etc.
* Once training is done our model will be saved in ‘runs\train\exp\weights\best.pt’



* To test our model, we’ll run detect.py cell. We'll pass the path of our custom trained model in --weights argument and in --source we’ll pass the path to our test images to test the detections of our model on unseen images. The detections will be saved in runs\detect\exp



* The Yolov5 model and its files will be saved in google drive and one can download them in the hour of need

1. Deploy the app on huggingface

Now our model has been trained next is to create a gradio app and deploy it to the huggingface.

* The app.py contains the gradio webapp
* Requirements.txt contains all requirements to deploy the app
* Download the best.pt model from google drive. It will be in yolov5/runs/train/exp/weights/best.pt
* Now we have all our files ready, login with your huggingface account. Click your profile at top right corner and select ‘New Space’. Choose a name for space, select space SDK (gradio), choose space hardware (free or paid), select visibility of the space (public or private) and click ‘Create Space’
* Click ‘Files and version’
* Choose ‘Add File’ and then ‘Upload files’
* Upload app.py, requirements.txt, best.pt
* Once all files are uploaded click ‘App’ at the top. The building process will be initiated and wait for a few minutes.
* Once build is finished the web app will be displayed and one can test the model by uploading the images and it will return number of objects and an image containing the bounding boxes around the objects with their confidence intervals