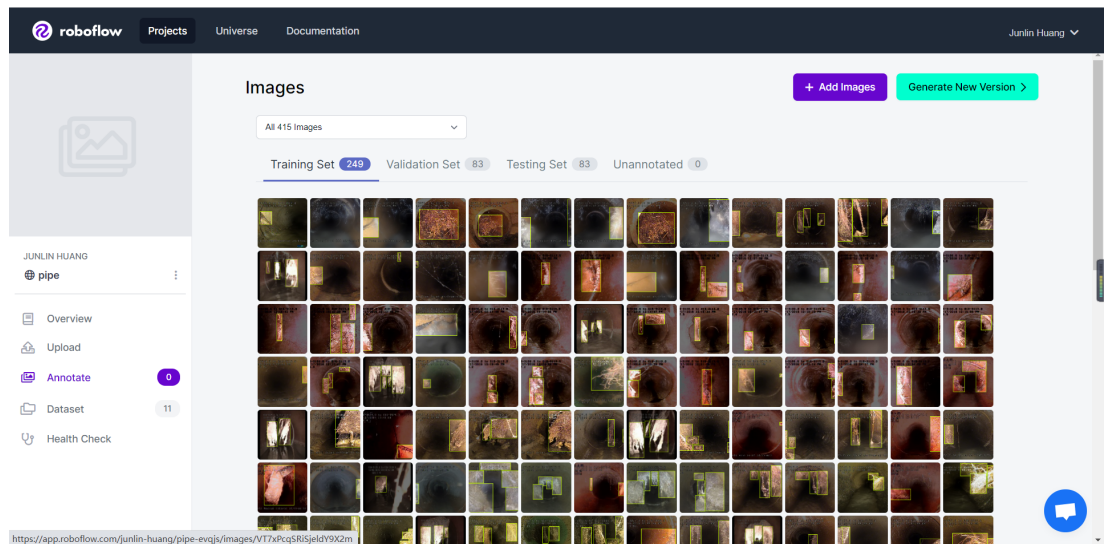


1. Data Labeling / Preprocessing

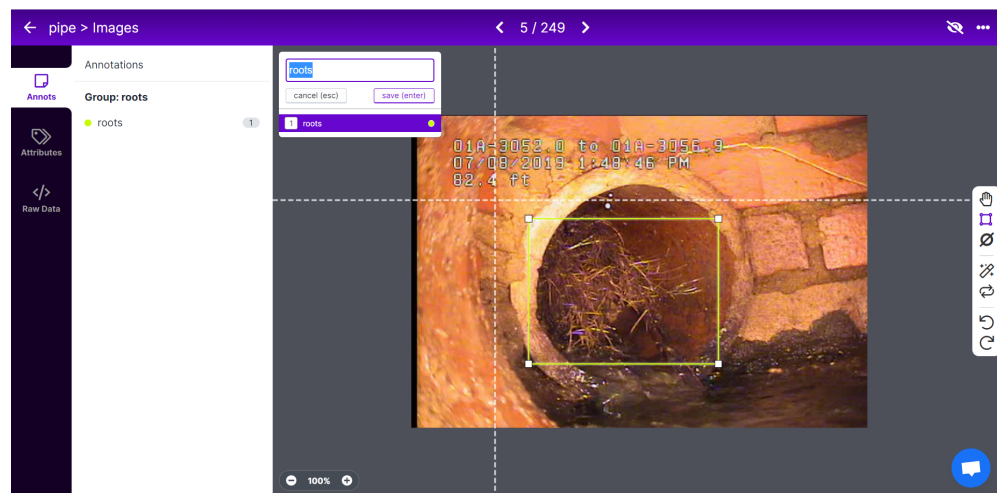
Tool: Roboflow.com

Steps:

1. Create project on roboflow, and upload unlabeled / labeled images
2. Use annotate tool on the project page to label all the images

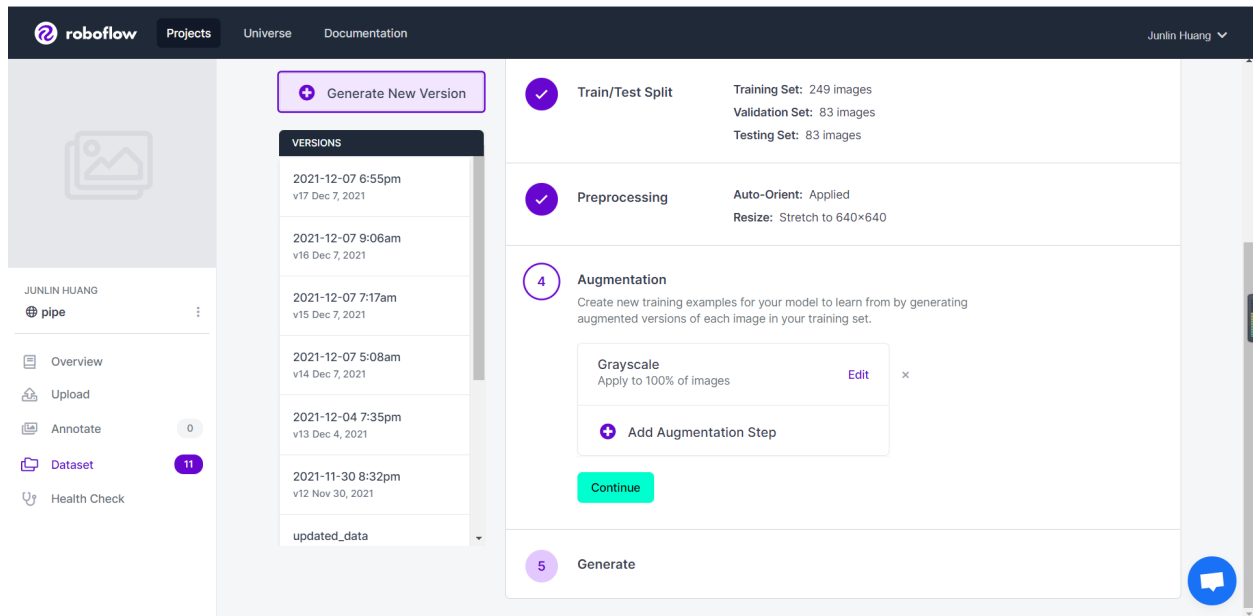


3. Open an image and use Create Tool in the bar on the right side to create a rectangular box to label a root.

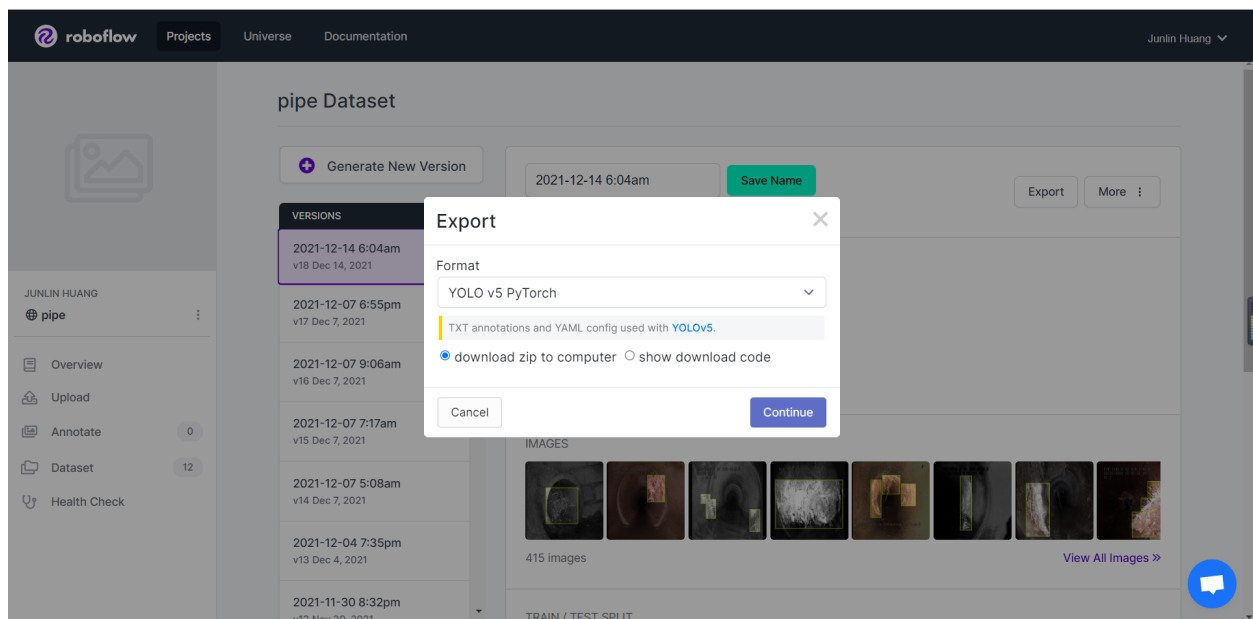


4. Label all images until there are no unannotated images.
5. Use the dataset tool to generate a version of the dataset.

- Recommend dataset settings: Image size 640 x 640, augmented 100% grayscale.



6. Export the generated dataset to YOLO v5 PyTorch format in a zip file.



2. Training Process

- Google colab is easier to set up
- Steps:
1. Open the YOLO jupyter notebook on google colab / anaconda
 2. Upload the dataset to google colab / anaconda

- First run the Setup portion in the notebook to clone YOLOv5 into the current directory.

▼ Setup

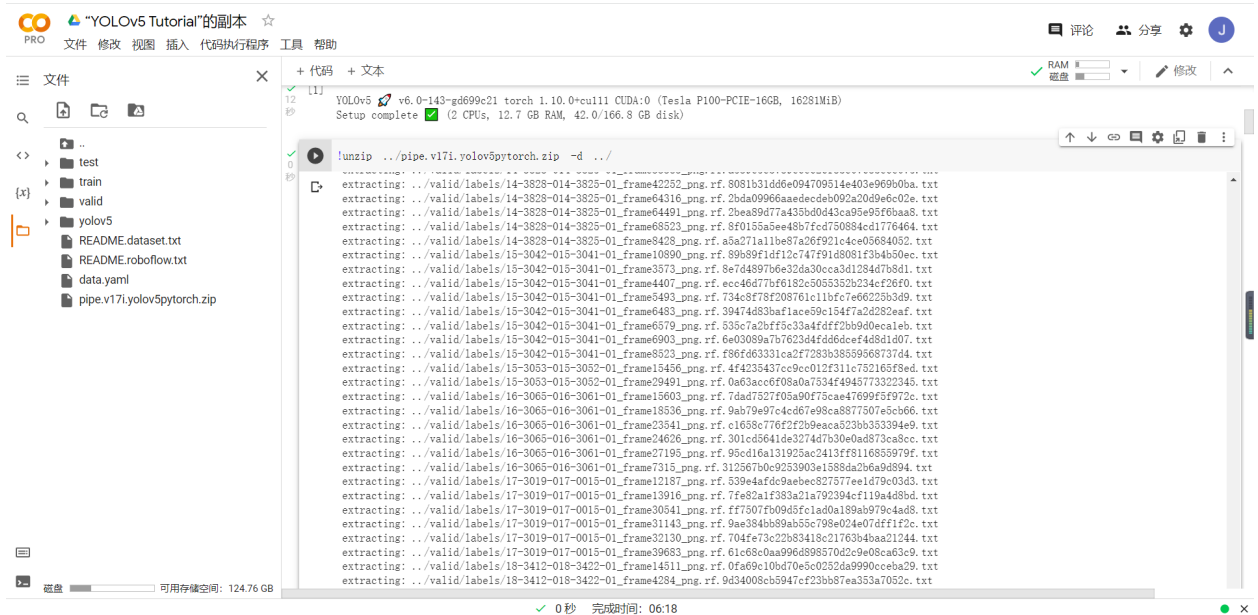
Clone repo, install dependencies and check PyTorch and GPU.

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

import torch
from yolov5 import utils
display = utils.notebook_init() # checks
```

YOLOv5 v6.0-143-gd699c21 torch 1.10.0+cu111 CUDA:0 (Tesla P100-PCIE-16GB, 16281MiB)
Setup complete

- Use unzip command to extract the dataset zip file to a chosen destination directory that stores the dataset

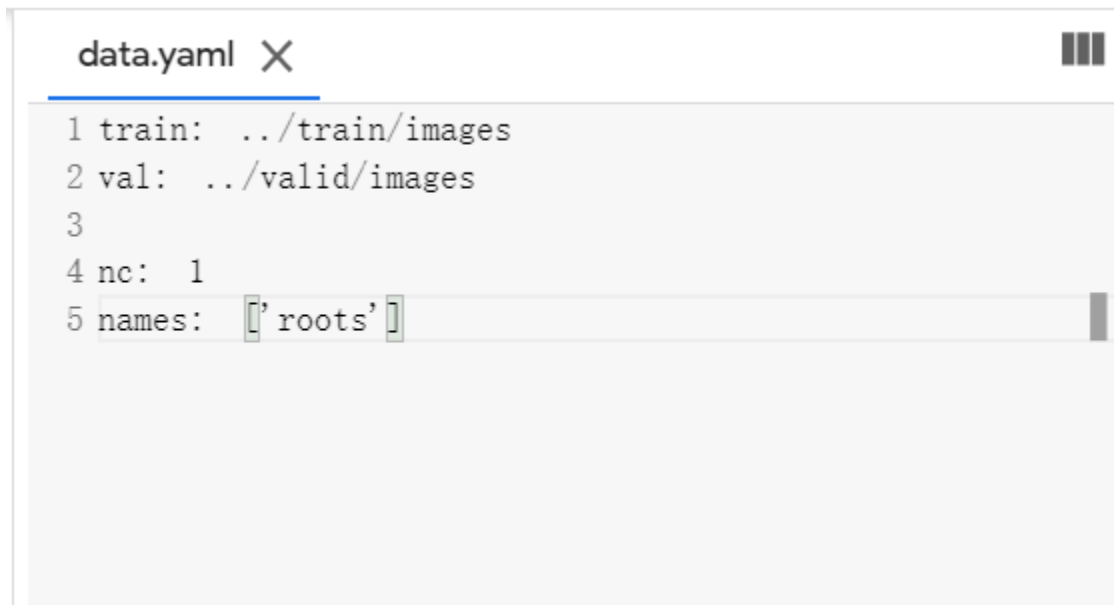


The screenshot shows a Jupyter Notebook titled "YOLOv5 Tutorial" with a file explorer on the left and a code cell on the right. The file explorer shows a directory structure with files like README.dataset.txt, README.robotflow.txt, data.yaml, and pipe.v171.yolov5pytorch.zip. The code cell contains the command `!unzip .../pipe.v171.yolov5pytorch.zip -d ../` and its output, which lists the contents of the extracted zip file, including various label files and image files. The status bar at the bottom indicates that the operation was completed successfully in 0 seconds.

```
!unzip .../pipe.v171.yolov5pytorch.zip -d ../
```

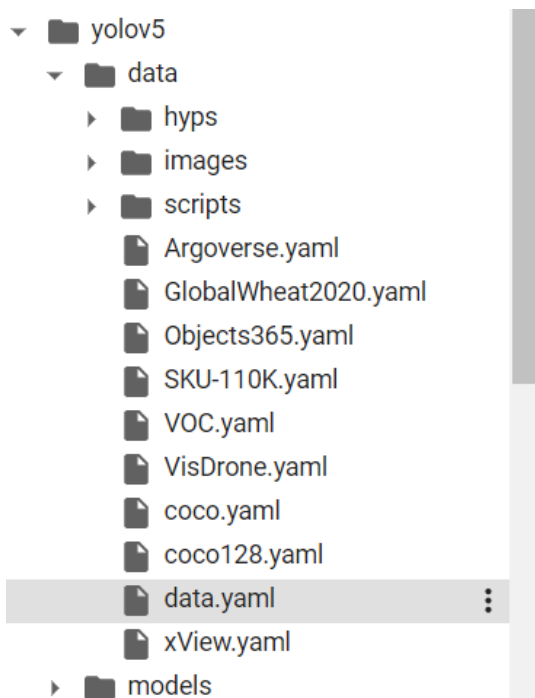
extracting: ../valid/labels/14-3828-014-3825-01_frame42252.png.rf.8081b31dd6e094709514e403e9690ba.txt
extracting: ../valid/labels/14-3828-014-3825-01_frame64316.png.rf.2bda09966aaedecde092a20d9e6c02e.txt
extracting: ../valid/labels/14-3828-014-3825-01_frame64491.png.rf.2bea89d77a433b0d443ca95e95f6baa8.txt
extracting: ../valid/labels/14-3828-014-3825-01_frame68523.png.rf.8f0155a5e48b7fcd750884cd1776464.txt
extracting: ../valid/labels/14-3828-014-3825-01_frame8428.png.rf.a8a271a1be87a26f921c4ce05684052.txt
extracting: ../valid/labels/15-3042-015-3041-01_frame10890.png.rf.89b89f1df12c747f91d8081f3b4b50ee.txt
extracting: ../valid/labels/15-3042-015-3041-01_frame3573.png.rf.8e7d4897b6e32da30cca3d1284d7b8d1.txt
extracting: ../valid/labels/15-3042-015-3041-01_frame4407.png.rf.ecc46d77b6f8182c5055352b234cf26f0.txt
extracting: ../valid/labels/15-3042-015-3041-01_frame6483.png.rf.734c8f78f208761c11bf7e6e22b3d9.txt
extracting: ../valid/labels/15-3042-015-3041-01_frame6579.png.rf.535c7a2bfff5c33a4fddf72b9d0decaeb.txt
extracting: ../valid/labels/15-3042-015-3041-01_frame6903.png.rf.6e03089a7b7623d4fdd6dce4d8d1d07.txt
extracting: ../valid/labels/15-3042-015-3041-01_frame8523.png.rf.f86fd63331ca2f7283b3859968737d4.txt
extracting: ../valid/labels/15-3053-015-3052-01_frame15456.png.rf.4f4235437ce9ce012f311c752165f9ed.txt
extracting: ../valid/labels/15-3053-015-3052-01_frame29491.png.rf.0a63acc6f08a0a7534f494577322345.txt
extracting: ../valid/labels/16-3065-016-3061-01_frame15603.png.rf.7dad7527f05a90f75cae47699f5f972c.txt
extracting: ../valid/labels/16-3065-016-3061-01_frame18536.png.rf.9ab79e97c4cd67e98ca8877507a5c66.txt
extracting: ../valid/labels/16-3065-016-3061-01_frame23541.png.rf.c1658c776f2f2b9eaca523b353394e9.txt
extracting: ../valid/labels/16-3065-016-3061-01_frame24626.png.rf.301cd5641de3274d7b30e0ad873ca8ec.txt
extracting: ../valid/labels/16-3065-016-3061-01_frame27195.png.rf.95cd16a131923ac2413ff811685979f.txt
extracting: ../valid/labels/16-3065-016-3061-01_frame7315.png.rf.312567b0c9253903e1588da2b6a9d894.txt
extracting: ../valid/labels/17-3019-017-0015-01_frame12187.png.rf.539e4afdc9aebec37377ee1d79c03d5.txt
extracting: ../valid/labels/17-3019-017-0015-01_frame13916.png.rf.7fe82a1f383a3e1792394ef119a4d8d.txt
extracting: ../valid/labels/17-3019-017-0015-01_frame30541.png.rf.f7f507f509d5f5c1ad0a189ab979c4a8.txt
extracting: ../valid/labels/17-3019-017-0015-01_frame31143.png.rf.9ae384bcb89ab58c798e024e07df1f2c.txt
extracting: ../valid/labels/17-3019-017-0015-01_frame32130.png.rf.704fe732c2b83418c21763b4baa21244.txt
extracting: ../valid/labels/17-3019-017-0015-01_frame39683.png.rf.61c68c0aa996d98570d2c9e08ca63c9.txt
extracting: ../valid/labels/18-3412-018-3422-01_frame14511.png.rf.0fa69c10bd70e5c0252da9990ceba29.txt
extracting: ../valid/labels/18-3412-018-3422-01_frame4284.png.rf.9d34008cb5947cf23bb87ea353a7052c.txt

5. Modify the data.yaml file extracted from the dataset, so that the “train” parameter is where the training portion of the dataset is stored, and the “val” parameter is where the validation portion of the dataset is stored. “nc” refers to the number of classes, in this case we have 1 class which is roots.

A screenshot of a code editor window titled "data.yaml" with a close button (X) and a hamburger menu icon. The editor contains the following YAML configuration:

```
1 train: ../train/images
2 val: ../valid/images
3
4 nc: 1
5 names: ['roots']
```

6. Copy the data.yaml file into the “data” folder in Yolo’s directory



7. To get the visualization reports on the training process, go to the “Train on Custom Data with Roboflow” part of the notebook, run the following part to use wandb tool to record the losses of the training process.

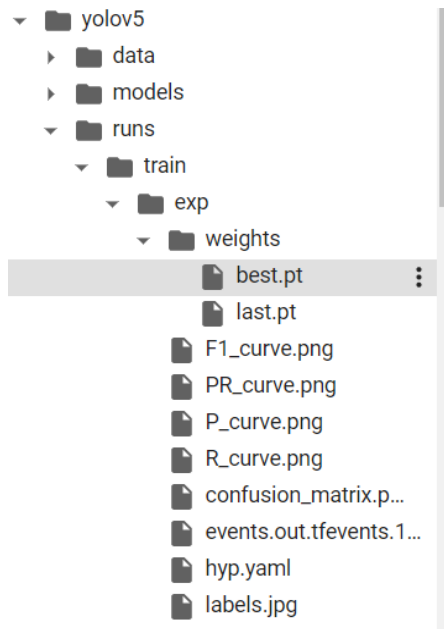
```
# Weights & Biases (optional)
!pip install -q wandb
import wandb
wandb.login()
```

8. To train the model, run the python train.py command with specific configurations. In this case, we set --img (img size) to 640 (since it was 640x640 when we generated the dataset), --batch (memory usage, 2/4/8/16/32/64, the larger the better, google colab pro can support 16 maximum), --epochs is set to 99 (refers to the number of iterations, depends on how many iterations the model can learn enough from the training dataset, this can be determined by after viewing the visualizations after the training process). --data (the configuration file for the dataset, which is the data.yaml we created before), --weights(the original pre-trained model we want to train on, yolov5x.pt is the best but takes time to train, other models can be found yolov5 github page <https://github.com/ultralytics/yolov5/releases>).

```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 16 --epochs 99 --data data.yaml --weights yolov5x.pt --cache
```

Epoch	gpu_mem	box	obj	cls	labels	img_size	
0/2	3.62G	0.04621	0.0711	0.02112	203	640: 100% 8/8	[00:04:00:00, 1.99it/s]
Class		Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 4/4 [00:00:00:00, 4.37it/s]
all		128	929	0.655	0.547	0.622	0.41
Epoch	gpu_mem	box	obj	cls	labels	img_size	
1/2	5.31G	0.04564	0.06898	0.02116	143	640: 100% 8/8	[00:01:00:00, 4.77it/s]
Class		Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 4/4 [00:00:00:00, 4.27it/s]
all		128	929	0.68	0.554	0.632	0.419
Epoch	gpu_mem	box	obj	cls	labels	img_size	
2/2	5.31G	0.04487	0.06883	0.01998	253	640: 100% 8/8	[00:01:00:00, 4.91it/s]
Class		Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 4/4 [00:00:00:00, 4.30it/s]
all		128	929	0.71	0.544	0.629	0.423

9. After the training process is done, (99 iterations takes around one and a half hour on google colab) the best iteration is saved to the “runs/train/exp/weights/best.pt” file in the yolov5 directory”

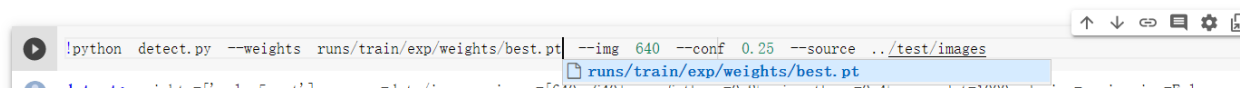


10. To detect roots for a folder of images / videos, use the python detect.py command in the “Inference” Part of the notebook. With configurations --weights (Here we specify the runs/train/exp/weights/best.pt file that we trained on pipe roots data) , --img (img size again 640x640), --source (the location of the images/ videos expected to be detected)

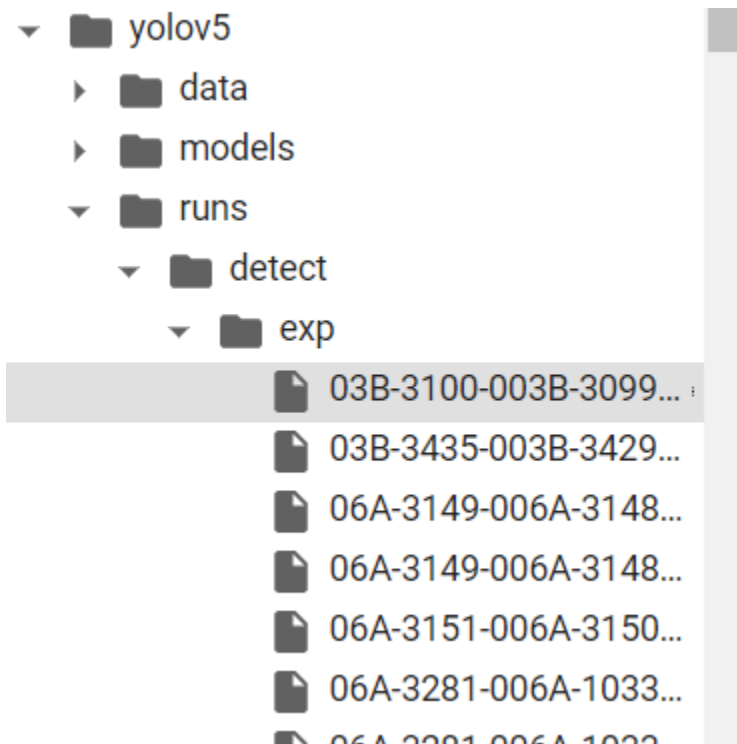
1. Inference

detect.py runs YOLOv5 inference on a variety of sources, downloading models automatically from the [latest YOLOv5 release](#), and saving results to runs/detect. Example inference sources are:

```
python detect.py --source 0 # webcam
img.jpg # image
vid.mp4 # video
path/ # directory
path/*.jpg # glob
'https://youtu.be/Zgi9g1ksQh0' # YouTube
'rtsp://example.com/media.mp4' # RTSP, RTMP, HTTP stream
```



11. The detected images / video will be saved to the “runs/detect/exp” folder.



12. Copy the output images to where you want to save the results. Below is an example result.

