CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

MDA572_NEURAL NETWORKS AND DEEP LEARNING

CIA – 2

(Develop a Video Processing Application using YOLO Model. Note: Try to create your own Dataset / Customize the existing dataset)

SUBMITTED BY : **ROHITH RAHUL**

REGISTER NO : **2282420**

DEPARTMENT : **DATA SCIENCE AND STATISTICS**

COURSE : **MSC DATA ANALYTICS – 5th TRIMESTER**

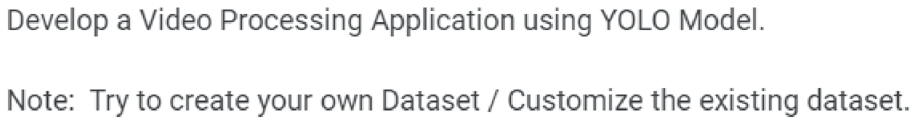
SUBMITTED TO : **Dr DALVIN VINOTH KUMAR A**
Associate Professor,
Department of Data Science and Statistics,
CHRIST (Deemed to be University).

DATE : **13-01-2024**

CONTENTS

OBJECT DETECTION USING YOLO v8 MODEL

OBJECTIVE



Develop a Video Processing Application using YOLO Model.

Note: Try to create your own Dataset / Customize the existing dataset.

OBJECT DETECTION

An essential component of computer vision is object detection, which is the process of locating and identifying things inside an image or video frame. Object detection techniques identify several items in an image and assign each one a bounding box that indicates its position inside the input image, in contrast to image classification tasks that provide a single label to a whole image.

Convolutional neural networks (CNNs), which are trained on enormous datasets to identify patterns and properties of various objects, are common deep learning models that power this activity. From self-driving automobiles, where it's critical to recognize other vehicles or pedestrians, to retail, where it may automatically track things on shelves, object detection algorithms have a wide range of uses.

With the development of various algorithms like R-CNN, SSD, and YOLO to maximize accuracy and inference speed over time, object identification has become a dynamic and quickly developing field in computer vision.

IMPORTANCE OF REAL-TIME OBJECT DETECTION

The ability to instantly recognize and locate things in a changing environment is what real-time object detection does to advance computer vision. "Real-time" refers to the ability to process images or video frames as they come in, usually at multiple frames per second, without any discernible lag.

This instantaneity is essential in applications where making snap decisions is crucial. For example, with autonomous cars, the capacity to recognize and respond in real time to pedestrians, traffic signals, and barriers might mean the difference between safe navigation and possible collisions.

Like this, real-time detection in security and surveillance can improve safety precautions by instantly alerting authorities to suspect activity. Furthermore, it is necessary for augmented reality (AR) apps to smoothly superimpose digital data on the real environment. Real-time object detection will become more and more necessary as technology develops, which emphasizes its importance in creating an interactive and responsive digital future.

YOLO ALGORITHM

The YOLO algorithm, an acronym for "You Only Look Once," transformed the field of object detection with its unique methodology. In contrast to conventional techniques that require several steps, like region suggestions and then classification, YOLO completes detection in a single network forward pass. The YOLO technique provides both speed and accuracy by partitioning an input image into a grid and concurrently predicting multiple bounding boxes and class probabilities for each grid cell.

Because of its unified architecture, it can process images very quickly, which makes it especially useful for real-time applications. The YOLO model has been released in multiple iterations since its creation, with each iteration improving its architecture and performance. The way that YOLO effectively strikes a balance between speed and accuracy highlights its importance in the field.

YOLO v8

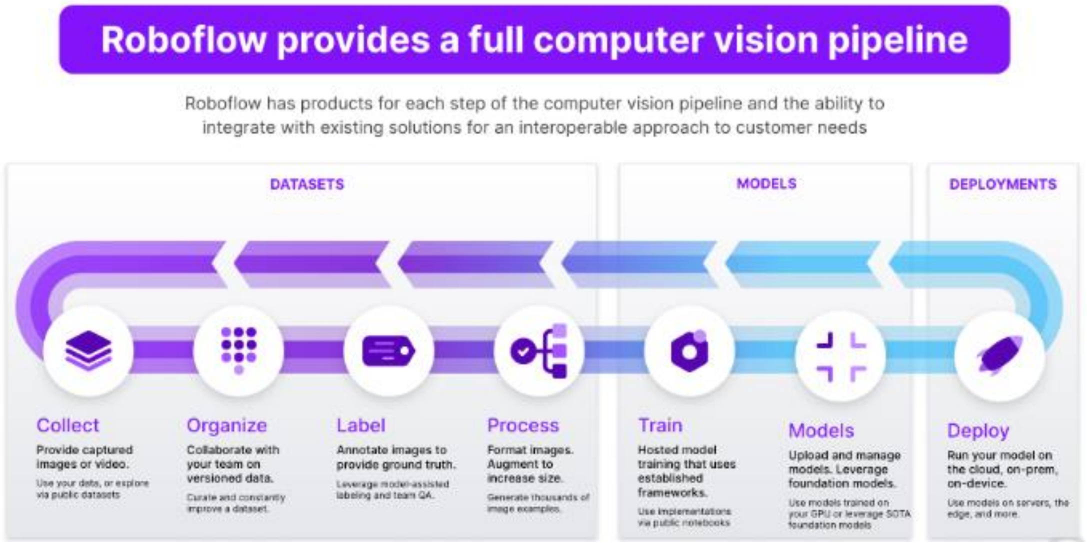
Compared to earlier iterations, YOLOv8 offers a plethora of additional features and enhancements. These include:

- An anchor-free design is used by YOLOv8, which facilitates model training across a variety of datasets.
- Self-attention mechanism: The self-attention mechanism in YOLOv8's network architecture aids in the improved learning of long-range dependencies between features.
- Adaptive training: To balance the loss function and maximize the learning rate during training, YOLOv8 employs adaptive training, which improves model performance.
- Advanced data augmentation: YOLOv8 uses several advanced data augmentation techniques, such as MixUp and Mosaic, to improve the robustness of the model to diverse variances in the data.

The need for computer vision models will continue to rise due to continued innovation, and YOLO will continue to have a distinct place in the market for several reasons. YOLO mainly relies on a unified detection technique that consolidates different object identification features into a single neural network to execute computer vision tasks successfully. Thanks to YOLO, the models can be trained using a single neural network into a full detection pipeline. It should come as no surprise that the method was used in a wide range of businesses and finally served as the focal point for object detection initiatives.

ROBOFLOW

Regardless of their level of expertise or experience, developers can create their own computer vision apps with Roboflow.



Roboflow provides a full computer vision pipeline

Roboflow has products for each step of the computer vision pipeline and the ability to integrate with existing solutions for an interoperable approach to customer needs

DATASETS

MODELS

DEPLOYMENTS

Collect

Provide captured images or video.

Use your data, or explore via public datasets

Organize

Collaborate with your team on versioned data.

Curate and constantly improve a dataset.

Label

Annotate images to provide ground truth.

Leverage model-assisted labeling and team QA.

Process

Format images. Augment to increase size.

Generate thousands of image examples.

Train

Hosted model training that uses established frameworks.

Use implementations via public notebooks

Models

Upload and manage models. Leverage foundation models.

Use models trained on your GPU or leverage SOTA foundation models

Deploy

Run your model on the cloud, on-prem, on-device.

Use models on servers, the edge, and more.

We offer all the resources required to transform an idea into a reliable computer vision model that is put into use.

To upload, arrange, work together, annotate, enhance, and handle your dataset, utilize Roboflow Datasets. After that, we can use Roboflow Train to streamline the training process and Roboflow Deploy to speed up deployment using a variety of deployment choices.

Roboflow is a deep learning tool that makes computer vision tasks easier. It gives developers the freedom to create computer vision apps regardless of their level of expertise or experience. Both object detection and classification models are supported. The following tasks can be completed using this tool:

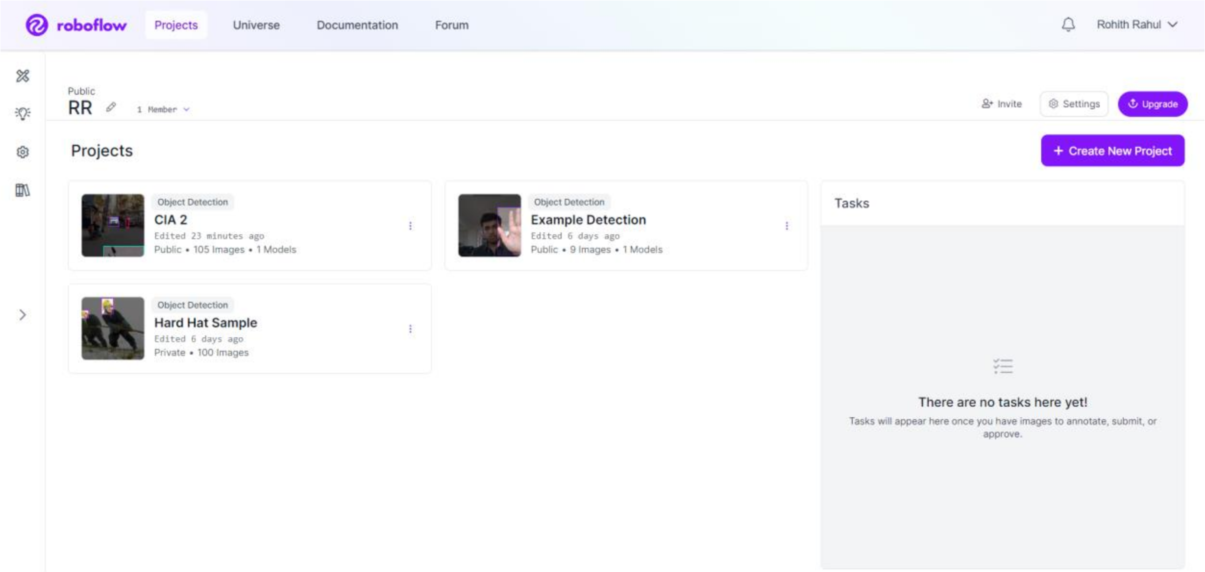
- Annotation of dataset
- Preprocessing of Datasets
- Project/Dataset Merge
- Check the health of the dataset
- Export Dataset
- Educate the model

MY PROJECT – OBJECT DETECTION USING YOLO v8 MODEL

➢ REGISTER AND SIGN IN:

Create an account by visiting the Roboflow website (https://roboflow.com/).

Access your account by logging in.



Public
RR
1 Member

Projects

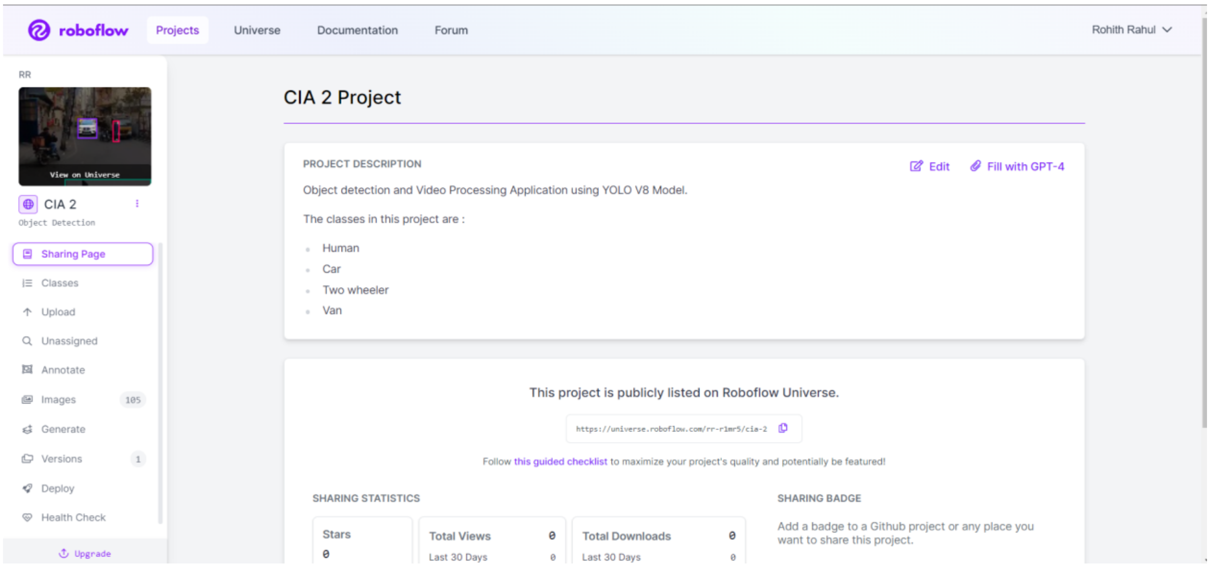
Object Detection
CIA 2
Edited 23 minutes ago
Public • 105 Images • 1 Models

Object Detection
Example Detection
Edited 6 days ago
Public • 9 Images • 1 Models

Object Detection
Hard Hat Sample
Edited 6 days ago
Private • 100 Images

Tasks

There are no tasks here yet!
Tasks will appear here once you have images to annotate, submit, or approve.



CIA 2
Object Detection

CIA 2 Project

PROJECT DESCRIPTION

Object detection and Video Processing Application using YOLO V8 Model.

The classes in this project are :

- Human
- Car
- Two wheeler
- Van

This project is publicly listed on Roboflow Universe.

https://universe.roboflow.com/rr-r1mr5/cia-2

Follow this guided checklist to maximize your project's quality and potentially be featured!

SHARING STATISTICS

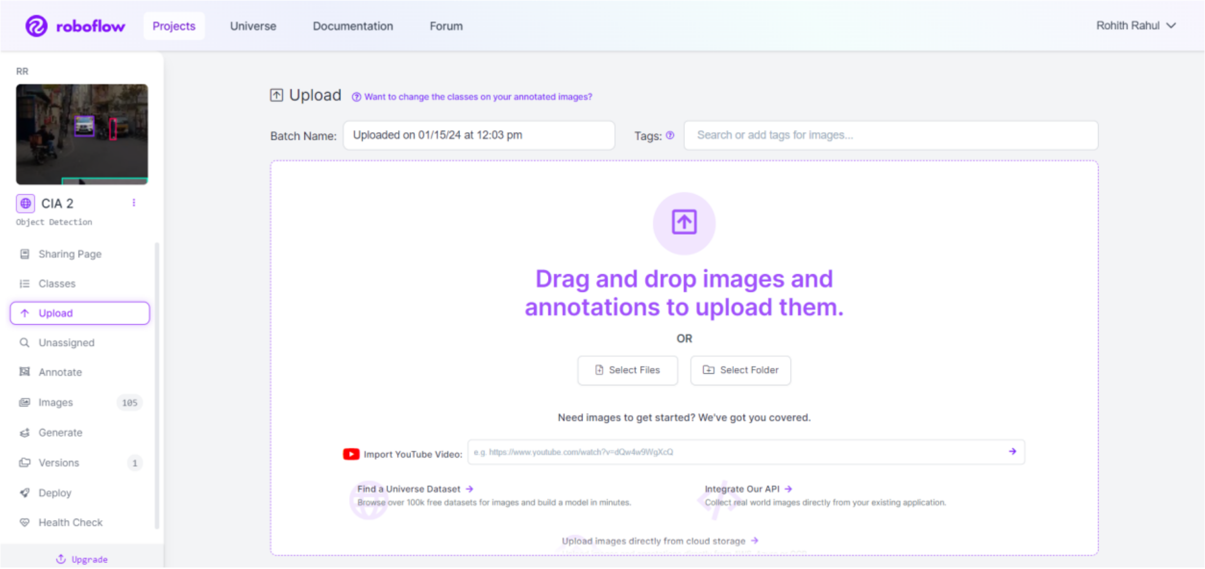
Stars	Total Views		Total Downloads	
0	Last 30 Days	0	Last 30 Days	0

SHARING BADGE

Add a badge to a Github project or any place you want to share this project.

➢ ASSEMBLE A DATASET:

Create a new dataset or upload an existing one after logging in. Roboflow can handle several picture formats.



Want to change the classes on your annotated images?

Batch Name: Uploaded on 01/15/24 at 12:03 pm

Tags: Search or add tags for images...

Drag and drop images and annotations to upload them.

OR

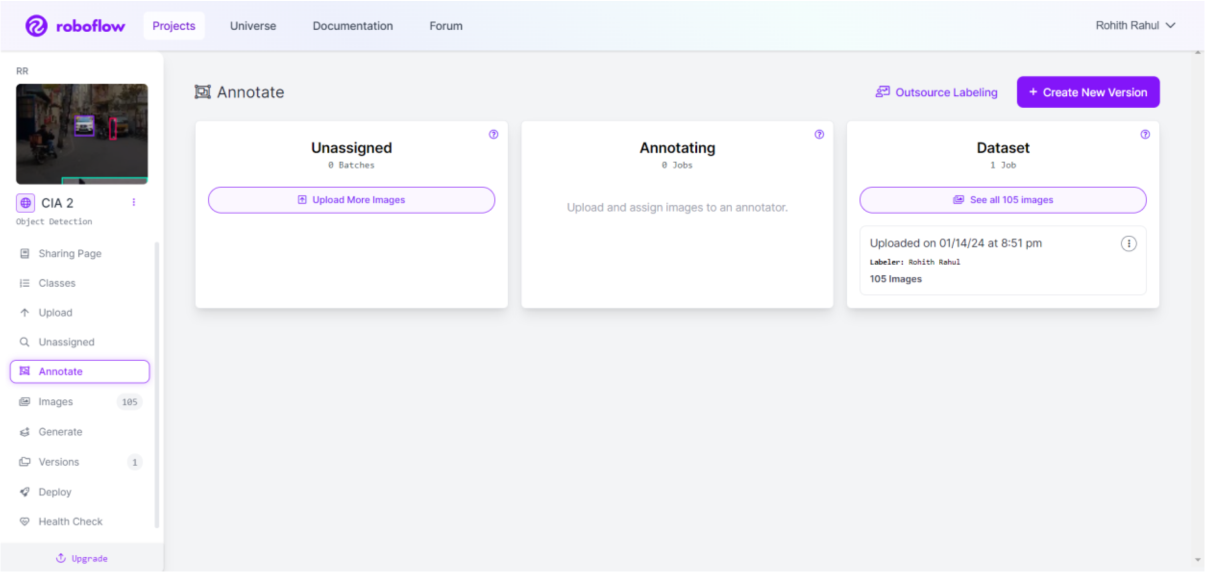
Need images to get started? We've got you covered.

Import YouTube Video: e.g. https://www.youtube.com/watch?v=dQw4w9WgXcQ

Find a Universe Dataset →
Browse over 100k free datasets for images and build a model in minutes.

Integrate Our API →
Collect real world images directly from your existing application.

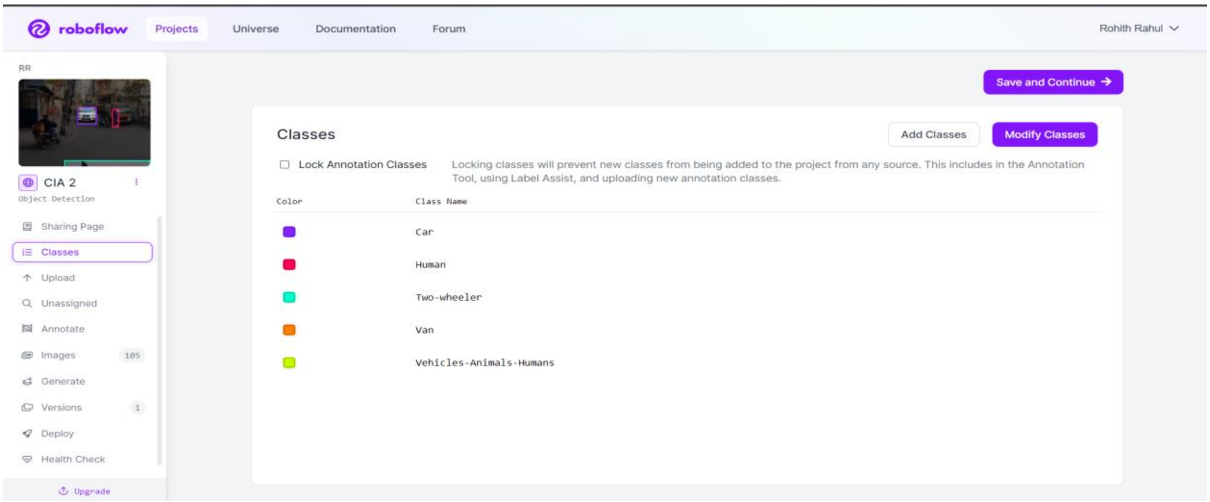
Upload images directly from cloud storage →



Unassigned
0 Batches

Annotating
0 Jobs
Upload and assign images to an annotator.

Dataset
1 Job
Uploaded on 01/14/24 at 8:51 pm
Labeler: Rohith Rahul
105 Images



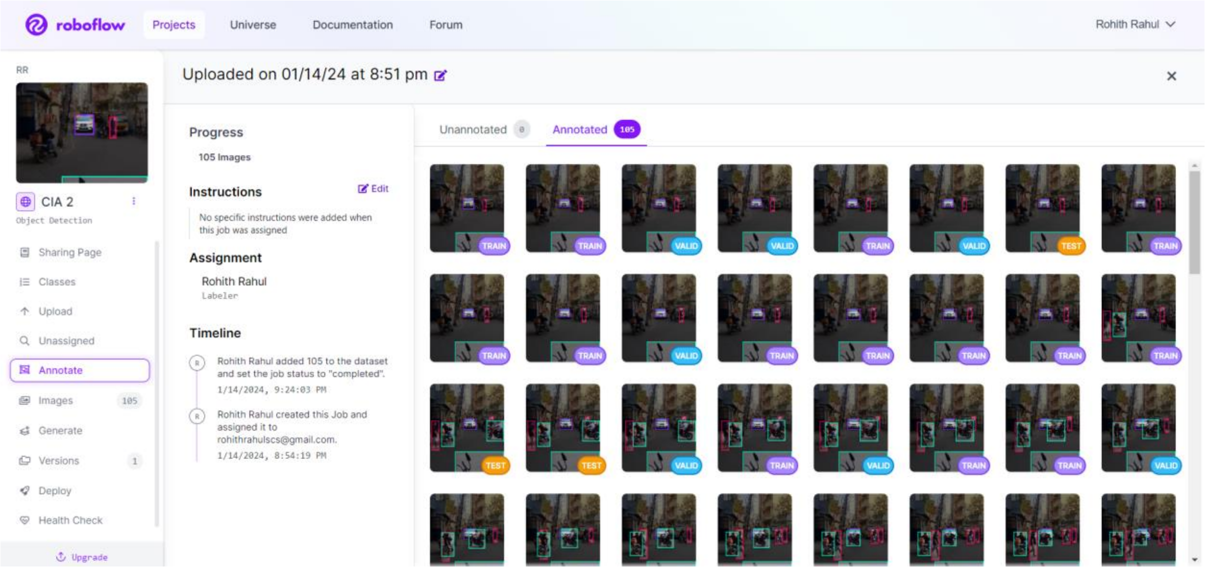
Classes

□ Lock Annotation Classes   Locking classes will prevent new classes from being added to the project from any source. This includes in the Annotation Tool, using Label Assist, and uploading new annotation classes.

Color	Class Name
🟪	Car
🟥	Human
🟦	Two-wheeler
🟧	Van
🟩	Vehicles-Animals-Humans

➢ PREPARING DATA:

Roboflow offers preparation techniques for data, such as image scaling, data augmentation, and normalization. Utilize these methods to improve the training procedure.



Uploaded on 01/14/24 at 8:51 pm

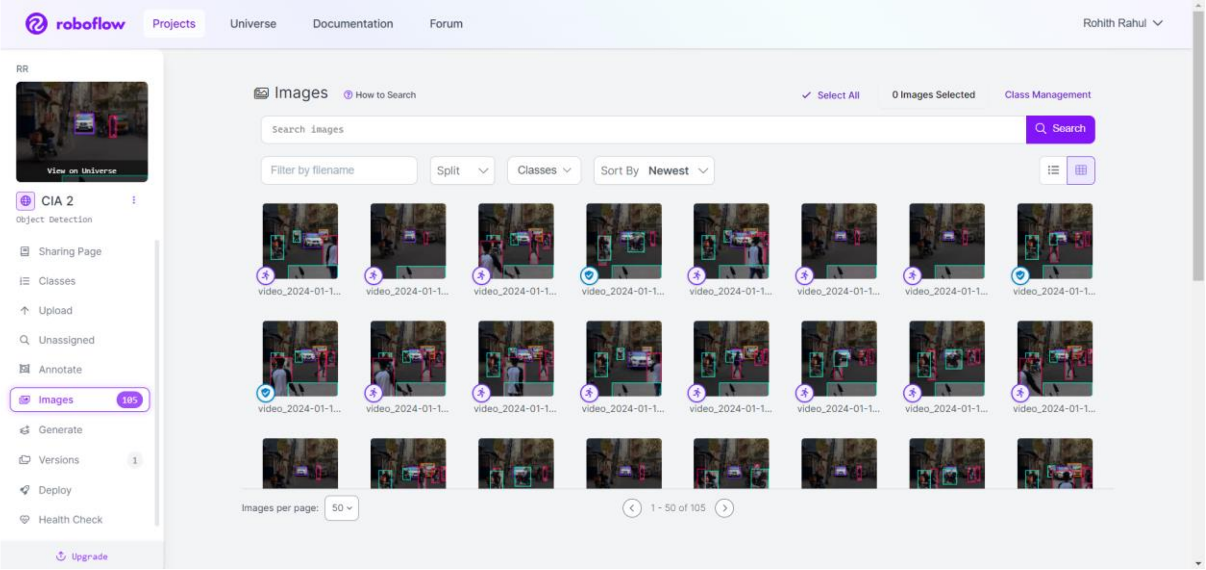
Progress
105 Images

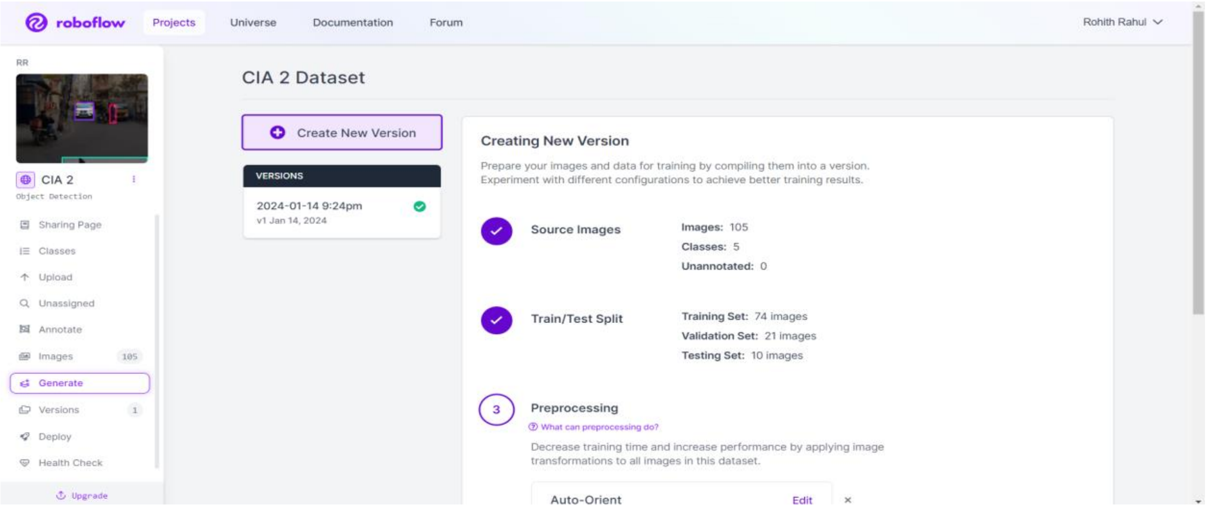
Instructions
No specific instructions were added when this job was assigned

Assignment
Rohith Rahul
Labeler

Timeline
Rohith Rahul added 105 to the dataset and set the job status to "completed".
1/14/2024, 9:24:03 PM

Rohith Rahul created this Job and assigned it to rohithrahulscs@gmail.com.
1/14/2024, 8:54:19 PM





CIA 2 Dataset

2024-01-14 9:24pm
v1 Jan 14, 2024

Creating New Version

Prepare your images and data for training by compiling them into a version. Experiment with different configurations to achieve better training results.

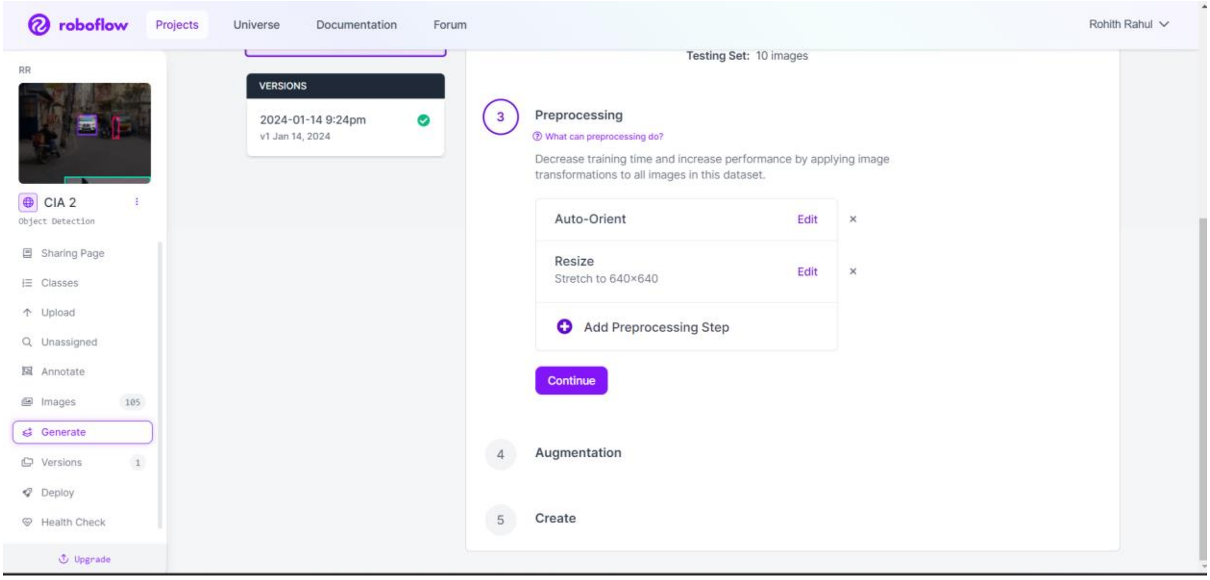
Source Images
Images: 105
Classes: 5
Unannotated: 0

Train/Test Split
Training Set: 74 images
Validation Set: 21 images
Testing Set: 10 images

3 Preprocessing

Decrease training time and increase performance by applying image transformations to all images in this dataset.

Auto-Orient



2024-01-14 9:24pm
v1 Jan 14, 2024

Testing Set: 10 images

3 Preprocessing

What can preprocessing do?

Decrease training time and increase performance by applying image transformations to all images in this dataset.

Auto-Orient Edit ×

Resize Stretch to 640×640 Edit ×

Add Preprocessing Step

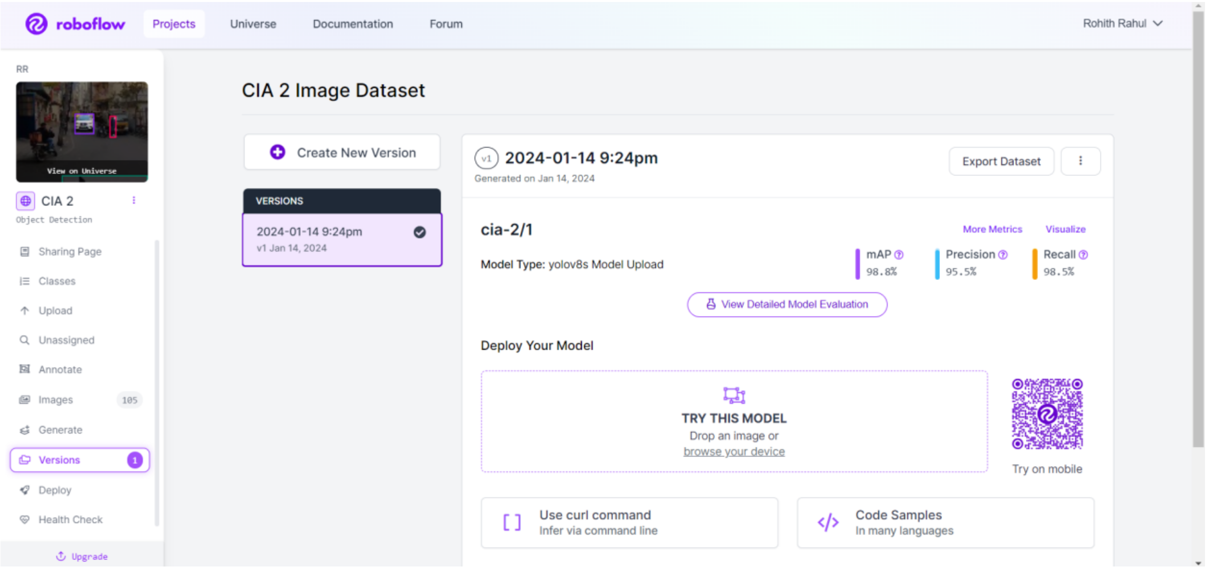
4 Augmentation

5 Create

➢ SET UP THE YOLO V8 MODEL:

In your dataset page, choose YOLO as the model architecture. Make sure you choose YOLO v8 from the model selections if it is available.

Set up the input size, anchor boxes, and other hyperparameters in the model.



CIA 2 Image Dataset

2024-01-14 9:24pm
v1 Jan 14, 2024

v1 2024-01-14 9:24pm
Generated on Jan 14, 2024

cia-2/1

Model Type: yolov8s Model Upload

mAP 98.8%

Precision 95.5%

Recall 98.5%

Deploy Your Model

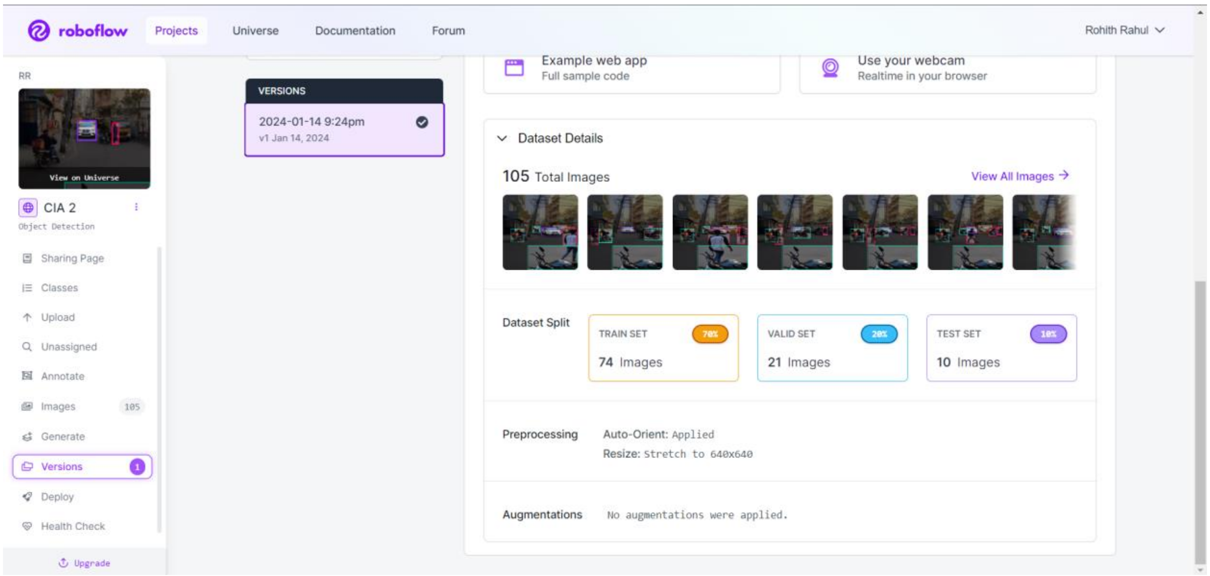
TRY THIS MODEL
Drop an image or
browse your device

Use curl command
Infer via command line

Code Samples
In many languages

➢ DATA DIVISION:

Divide the dataset into test, validation, and training sets. This aids in evaluating the model's effectiveness with unknown data.



2024-01-14 9:24pm
v1 Jan 14, 2024

Dataset Details

105 Total Images

Dataset Split

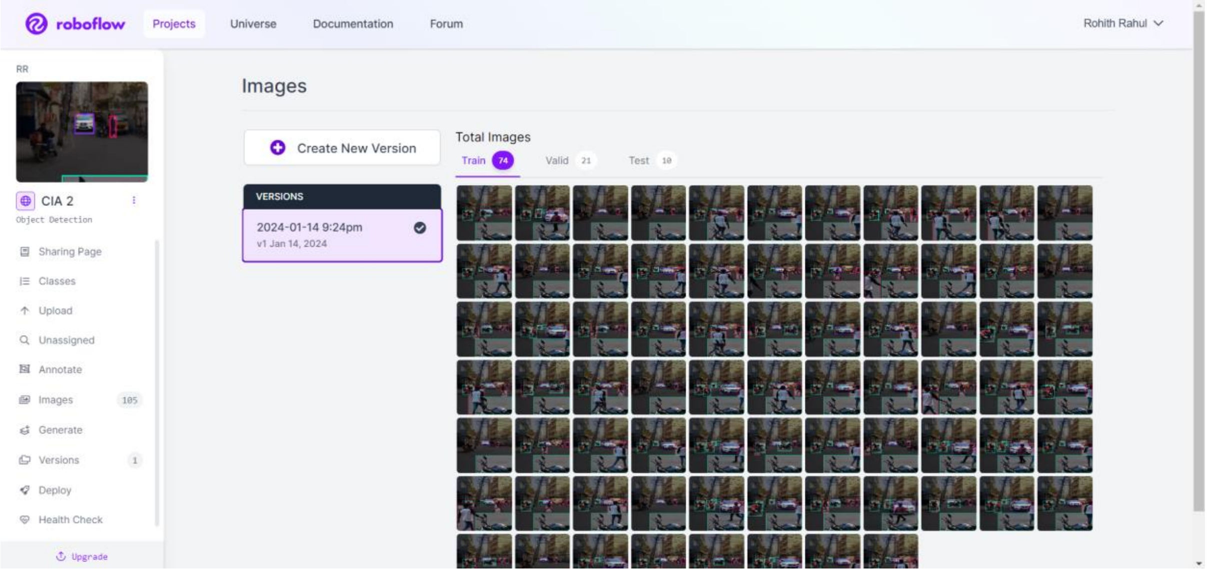
TRAIN SET 70% — 74 Images

VALID SET 20% — 21 Images

TEST SET 10% — 10 Images

Preprocessing: Auto-Orient: Applied
Resize: Stretch to 640x640

Augmentations: No augmentations were applied.



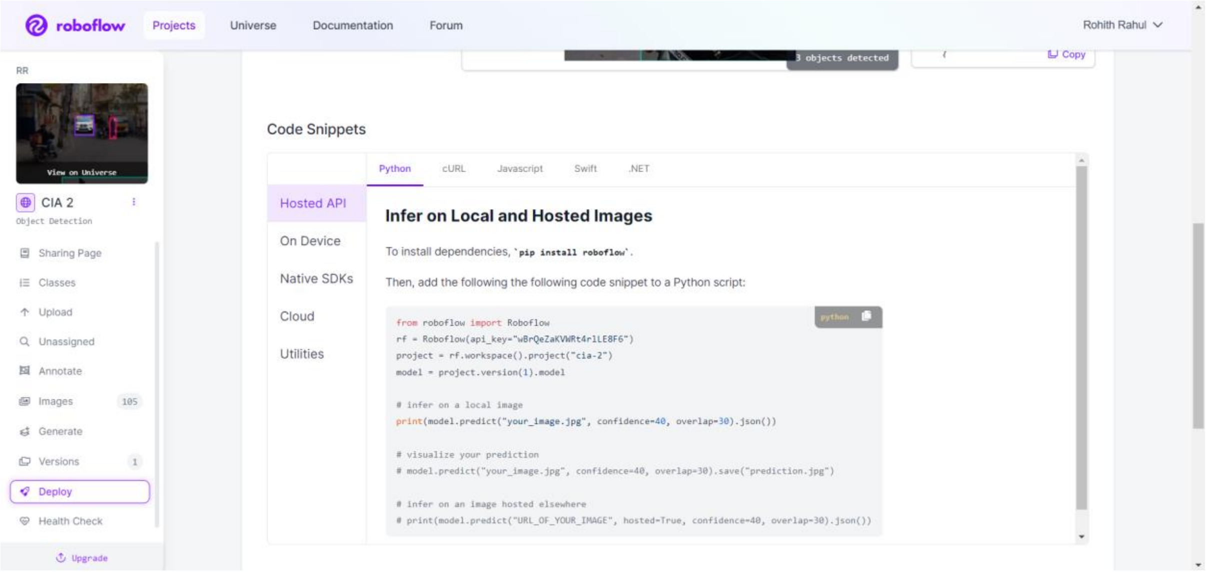
2024-01-14 9:24pm
v1 Jan 14, 2024

Total Images

Train 74 Valid 21 Test 10

➢ CREATE CODE AND GET IT HERE:

You can create code snippets for your favorite deep learning framework (e.g., PyTorch, TensorFlow) with Roboflow. After deciding on the suitable framework, download the code.



Code Snippets

Infer on Local and Hosted Images

To install dependencies, `pip install roboflow`.

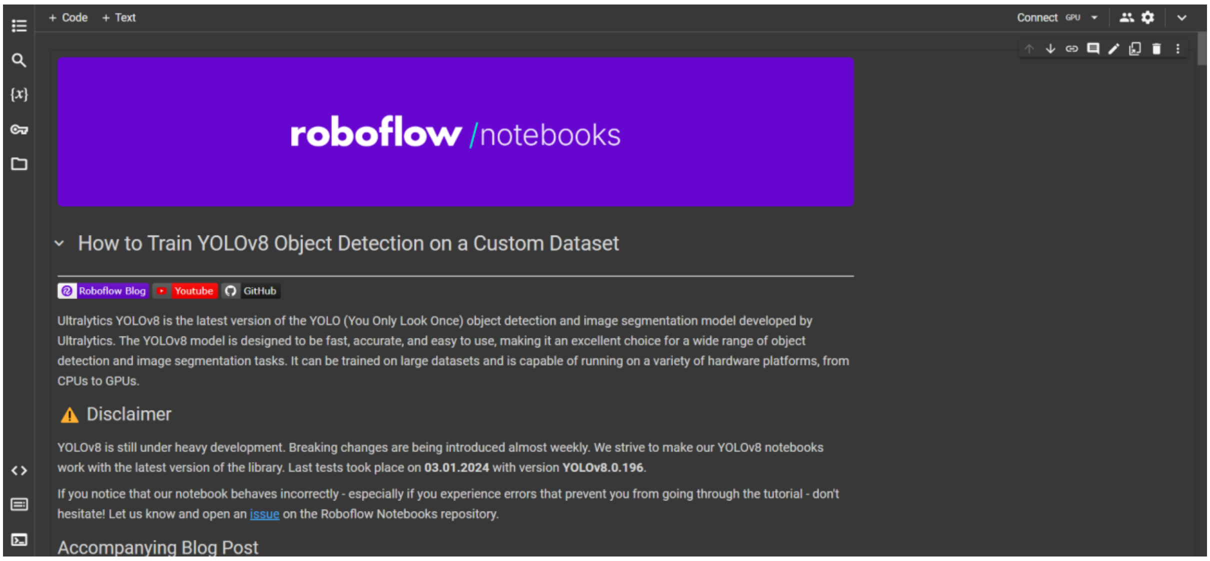
Then, add the following the following code snippet to a Python script:

```python
from roboflow import Roboflow
rf = Roboflow(api_key="wBrQeZaKVWRt4rlLE8F6")
project = rf.workspace().project("cia-2")
model = project.version(1).model

# infer on a local image
print(model.predict("your_image.jpg", confidence=40, overlap=30).json())

# visualize your prediction
# model.predict("your_image.jpg", confidence=40, overlap=30).save("prediction.jpg")

# infer on an image hosted elsewhere
# print(model.predict("URL_OF_YOUR_IMAGE", hosted=True, confidence=40, overlap=30).json())
```

➢ EDUCATE THE MODEL:

Your YOLO v8 model can be trained using the given code. Observe the guidelines given in the code sample.

Track training results and make any necessary hyperparameter adjustments.



How to Train YOLOv8 Object Detection on a Custom Dataset

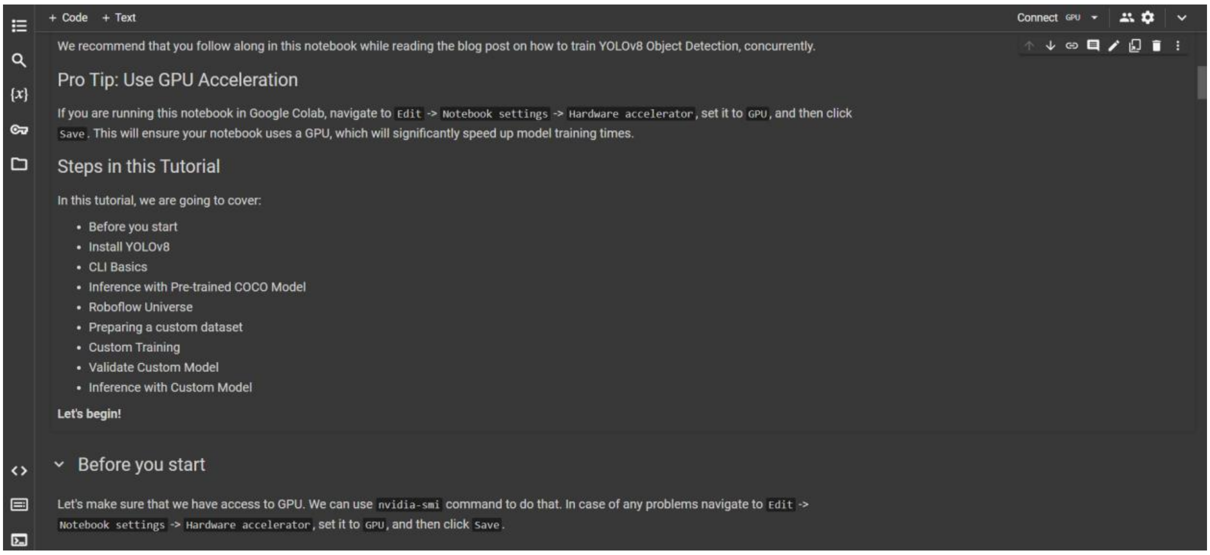
Ultralytics YOLOv8 is the latest version of the YOLO (You Only Look Once) object detection and image segmentation model developed by Ultralytics. The YOLOv8 model is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection and image segmentation tasks. It can be trained on large datasets and is capable of running on a variety of hardware platforms, from CPUs to GPUs.

⚠ Disclaimer

YOLOv8 is still under heavy development. Breaking changes are being introduced almost weekly. We strive to make our YOLOv8 notebooks work with the latest version of the library. Last tests took place on **03.01.2024** with version **YOLOv8.0.196**.

If you notice that our notebook behaves incorrectly - especially if you experience errors that prevent you from going through the tutorial - don't hesitate! Let us know and open an issue on the Roboflow Notebooks repository.

Accompanying Blog Post



We recommend that you follow along in this notebook while reading the blog post on how to train YOLOv8 Object Detection, concurrently.

Pro Tip: Use GPU Acceleration

If you are running this notebook in Google Colab, navigate to `Edit` -> `Notebook settings` -> `Hardware accelerator`, set it to `GPU`, and then click `Save`. This will ensure your notebook uses a GPU, which will significantly speed up model training times.

Steps in this Tutorial

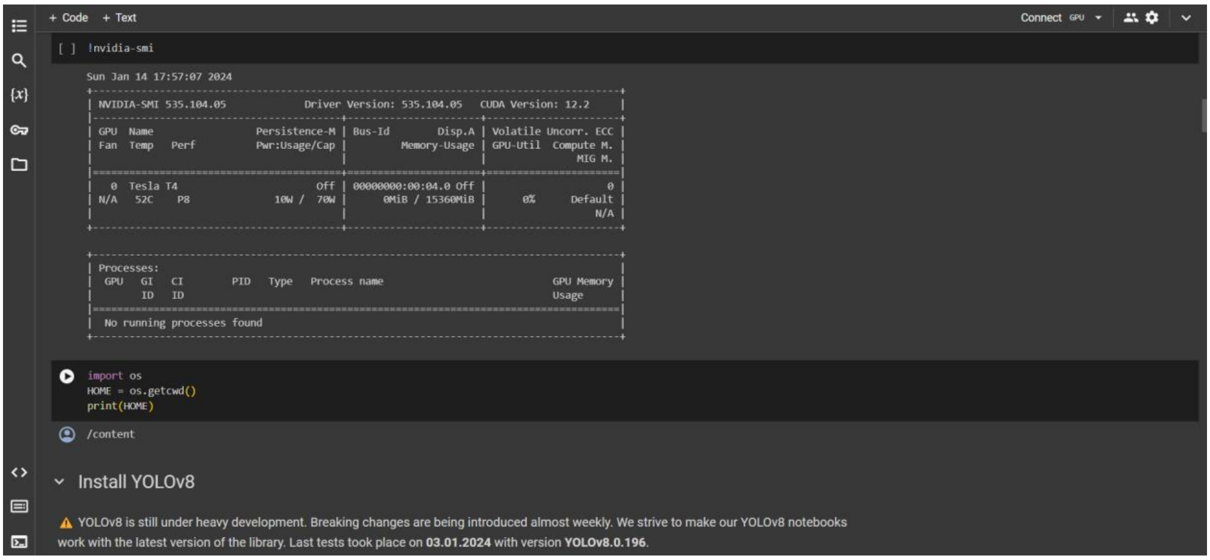
In this tutorial, we are going to cover:

- Before you start
- Install YOLOv8
- CLI Basics
- Inference with Pre-trained COCO Model
- Roboflow Universe
- Preparing a custom dataset
- Custom Training
- Validate Custom Model
- Inference with Custom Model

Let's begin!

Before you start

Let's make sure that we have access to GPU. We can use `nvidia-smi` command to do that. In case of any problems navigate to `Edit` -> `Notebook settings` -> `Hardware accelerator`, set it to `GPU`, and then click `Save`.



```
[ ] !nvidia-smi
```

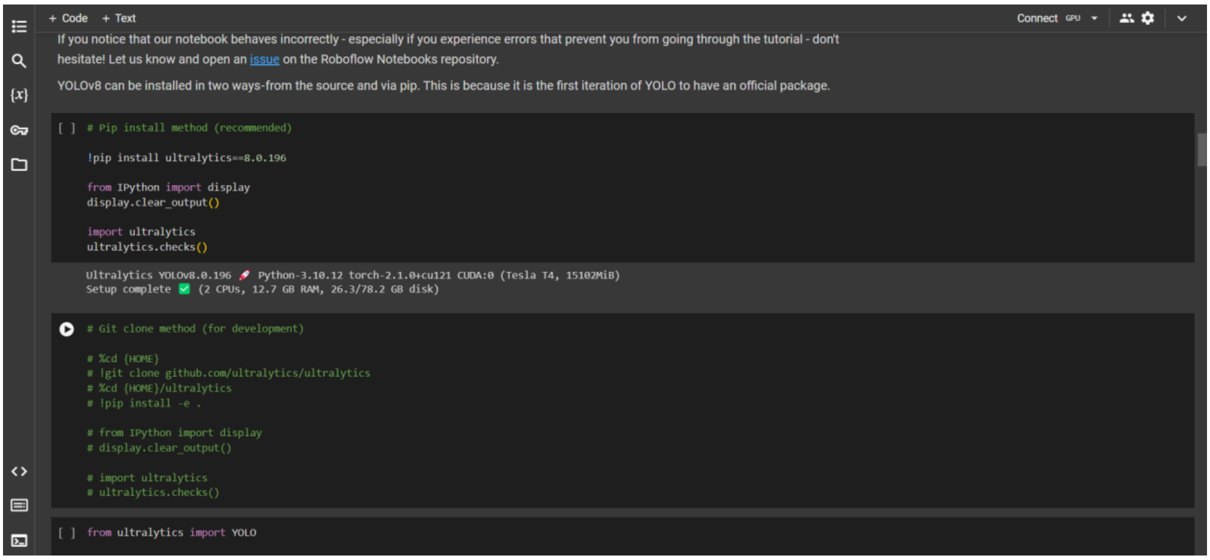
```
Sun Jan 14 17:57:07 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   52C    P8              10W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

```
import os
HOME = os.getcwd()
print(HOME)
```

```
/content
```

Install YOLOv8

⚠ YOLOv8 is still under heavy development. Breaking changes are being introduced almost weekly. We strive to make our YOLOv8 notebooks work with the latest version of the library. Last tests took place on **03.01.2024** with version **YOLOv8.0.196**.



If you notice that our notebook behaves incorrectly - especially if you experience errors that prevent you from going through the tutorial - don't hesitate! Let us know and open an issue on the Roboflow Notebooks repository.

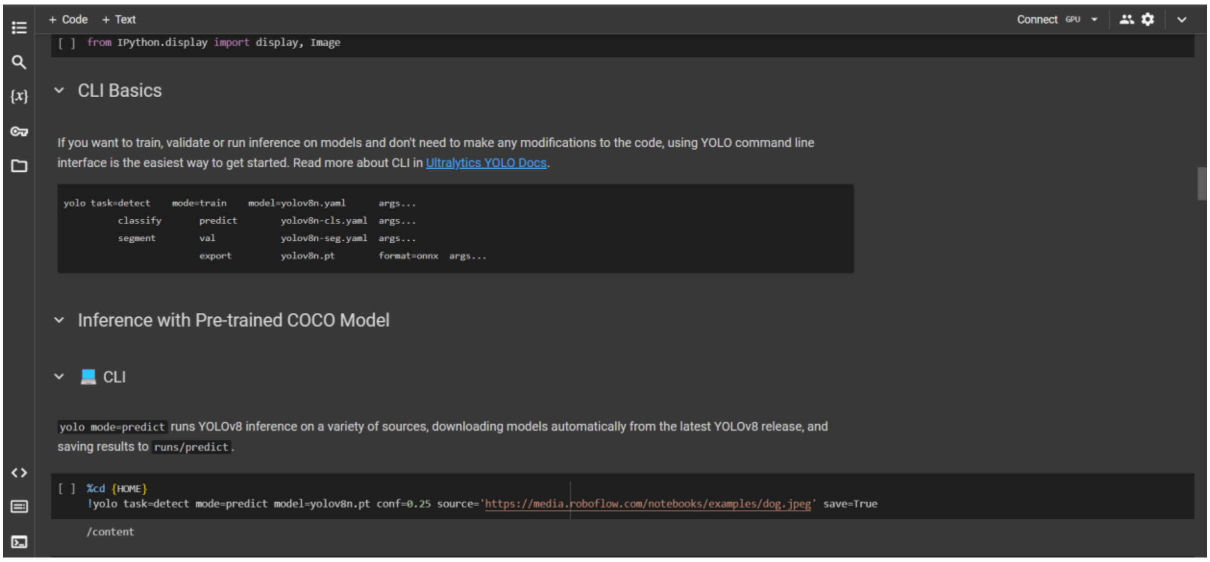
YOLOv8 can be installed in two ways-from the source and via pip. This is because it is the first iteration of YOLO to have an official package.

```
[ ] # Pip install method (recommended)

    !pip install ultralytics==8.0.196

    from IPython import display
    display.clear_output()

    import ultralytics
    ultralytics.checks()
```

```
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 26.3/78.2 GB disk)
```

```
# Git clone method (for development)

# %cd {HOME}
# !git clone github.com/ultralytics/ultralytics
# %cd {HOME}/ultralytics
# !pip install -e .

# from IPython import display
# display.clear_output()

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

```
[ ] from ultralytics import YOLO
```



```
[ ] from IPython.display import display, Image
```

CLI Basics

If you want to train, validate or run inference on models and don't need to make any modifications to the code, using YOLO command line interface is the easiest way to get started. Read more about CLI in Ultralytics YOLO Docs.

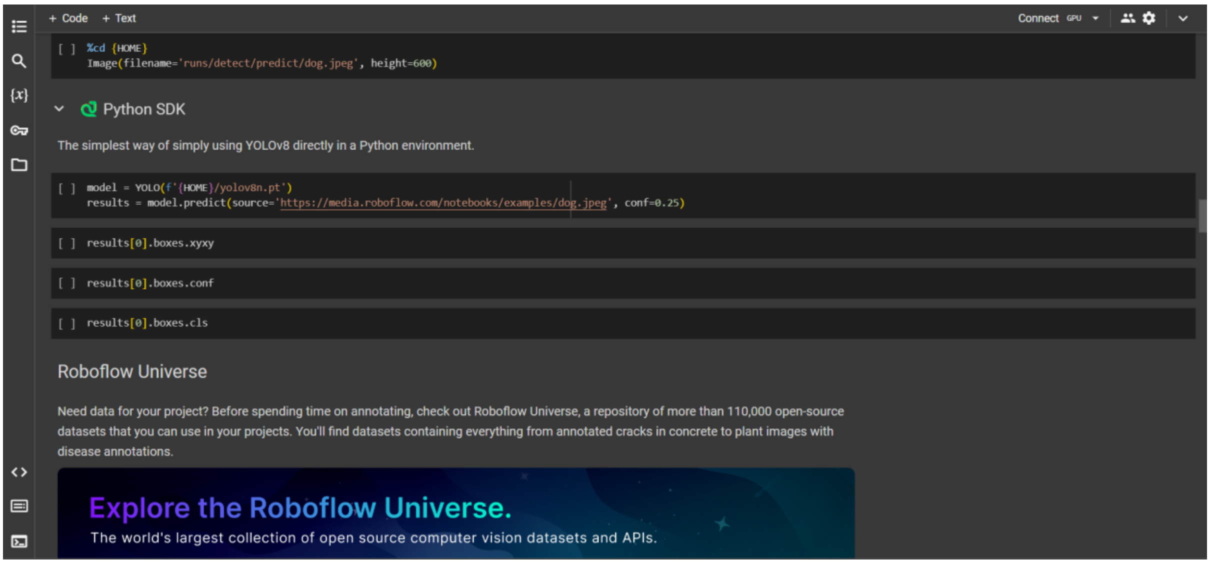
```
yolo task=detect    mode=train    model=yolov8n.yaml      args...
          classify       predict        yolov8n-cls.yaml  args...
          segment        val            yolov8n-seg.yaml  args...
                         export         yolov8n.pt        format=onnx  args...
```

Inference with Pre-trained COCO Model

💻 CLI

`yolo mode=predict` runs YOLOv8 inference on a variety of sources, downloading models automatically from the latest YOLOv8 release, and saving results to `runs/predict`.

```
[ ] %cd {HOME}
    !yolo task=detect mode=predict model=yolov8n.pt conf=0.25 source='https://media.roboflow.com/notebooks/examples/dog.jpeg' save=True

    /content
```



```
[ ] %cd {HOME}
    Image(filename='runs/detect/predict/dog.jpeg', height=600)
```

Python SDK

The simplest way of simply using YOLOv8 directly in a Python environment.

```
[ ] model = YOLO(f'{HOME}/yolov8n.pt')
    results = model.predict(source='https://media.roboflow.com/notebooks/examples/dog.jpeg', conf=0.25)
```

```
[ ] results[0].boxes.xyxy
```

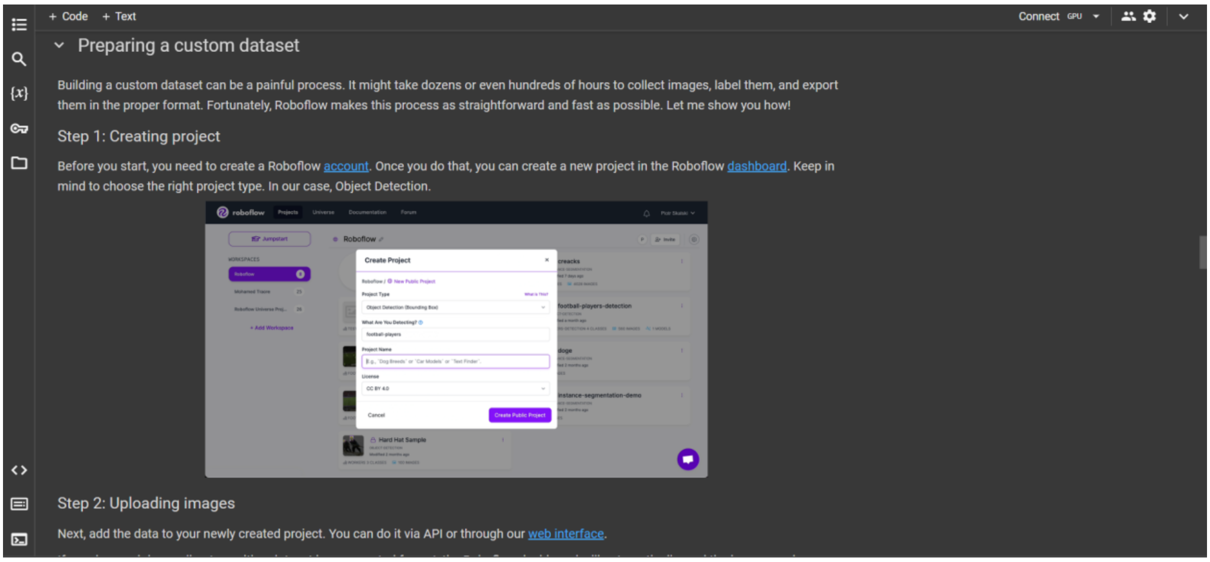
```
[ ] results[0].boxes.conf
```

```
[ ] results[0].boxes.cls
```

Roboflow Universe

Need data for your project? Before spending time on annotating, check out Roboflow Universe, a repository of more than 110,000 open-source datasets that you can use in your projects. You'll find datasets containing everything from annotated cracks in concrete to plant images with disease annotations.

Explore the Roboflow Universe.
The world's largest collection of open source computer vision datasets and APIs.


+ Code
+ Text
Connect GPU

Preparing a custom dataset

Building a custom dataset can be a painful process. It might take dozens or even hundreds of hours to collect images, label them, and export them in the proper format. Fortunately, Roboflow makes this process as straightforward and fast as possible. Let me show you how!

Step 1: Creating project

Before you start, you need to create a Roboflow account. Once you do that, you can create a new project in the Roboflow dashboard. Keep in mind to choose the right project type. In our case, Object Detection.

Project Type: Object Detection (Bounding Box)

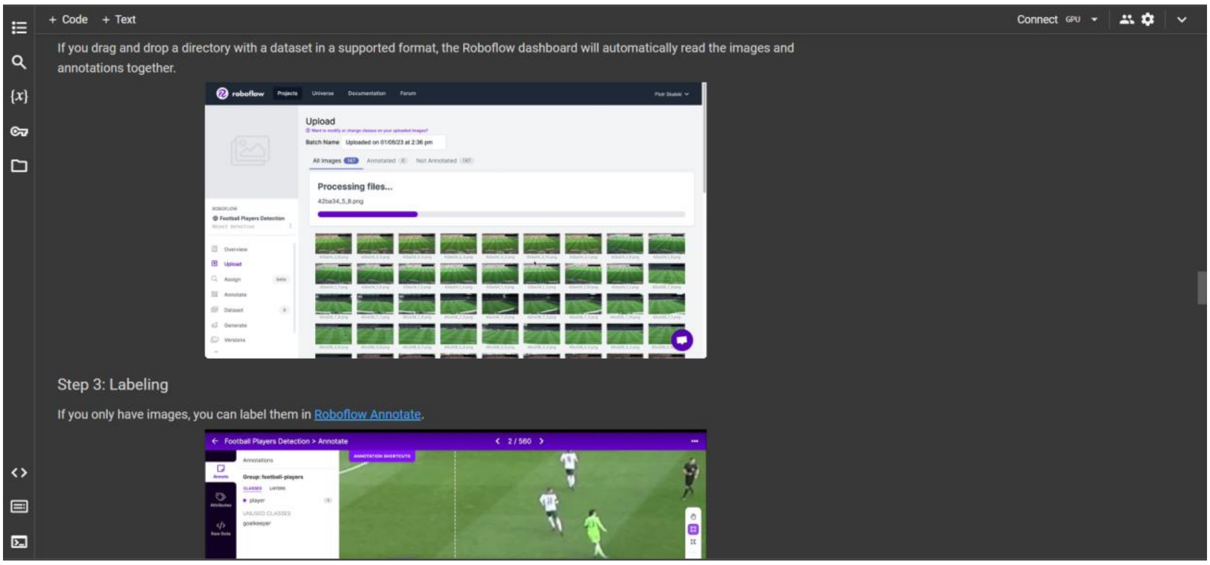
What Are You Detecting? football-players

Project Name: E.g., 'Dog Breeds' or 'Car Models' or 'Text Finder'.

License: CC BY 4.0

Step 2: Uploading images

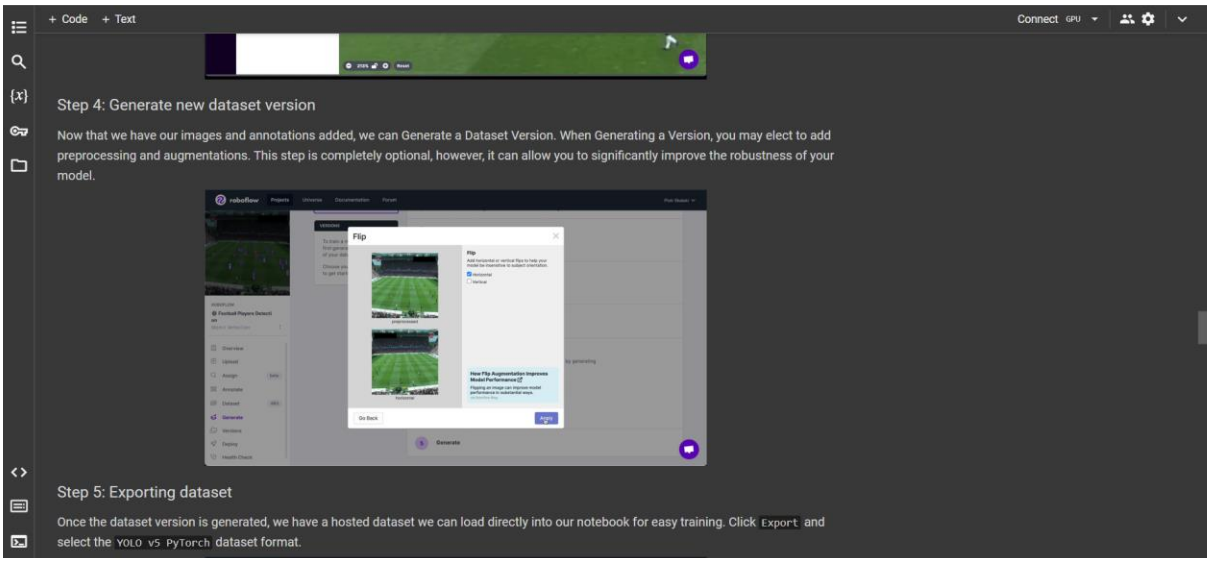
Next, add the data to your newly created project. You can do it via API or through our web interface.



If you drag and drop a directory with a dataset in a supported format, the Roboflow dashboard will automatically read the images and annotations together.

Step 3: Labeling

If you only have images, you can label them in Roboflow Annotate.

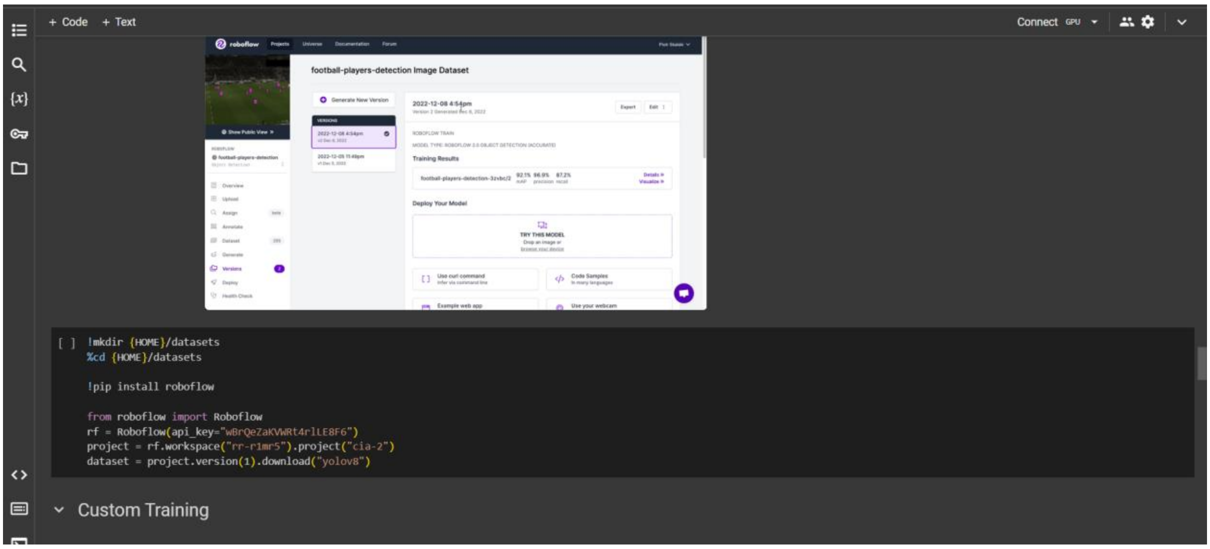


Step 4: Generate new dataset version

Now that we have our images and annotations added, we can Generate a Dataset Version. When Generating a Version, you may elect to add preprocessing and augmentations. This step is completely optional, however, it can allow you to significantly improve the robustness of your model.

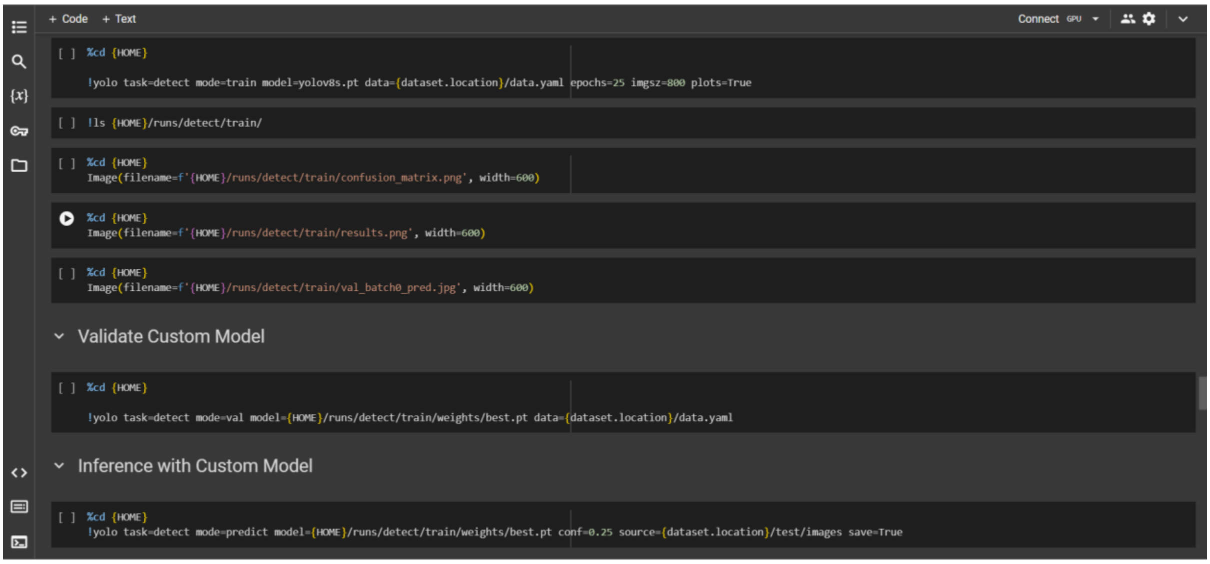
Step 5: Exporting dataset

Once the dataset version is generated, we have a hosted dataset we can load directly into our notebook for easy training. Click `Export` and select the `YOLO v5 PyTorch` dataset format.



```
[ ] !mkdir {HOME}/datasets
    %cd {HOME}/datasets

    !pip install roboflow

    from roboflow import Roboflow
    rf = Roboflow(api_key="wBrQeZaKVWRt4rlLE8F6")
    project = rf.workspace("rr-r1mr5").project("cia-2")
    dataset = project.version(1).download("yolov8")
```

Custom Training



```
%cd {HOME}

!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml epochs=25 imgsz=800 plots=True
```

```
!ls {HOME}/runs/detect/train/
```

```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/confusion_matrix.png', width=600)
```

```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/results.png', width=600)
```

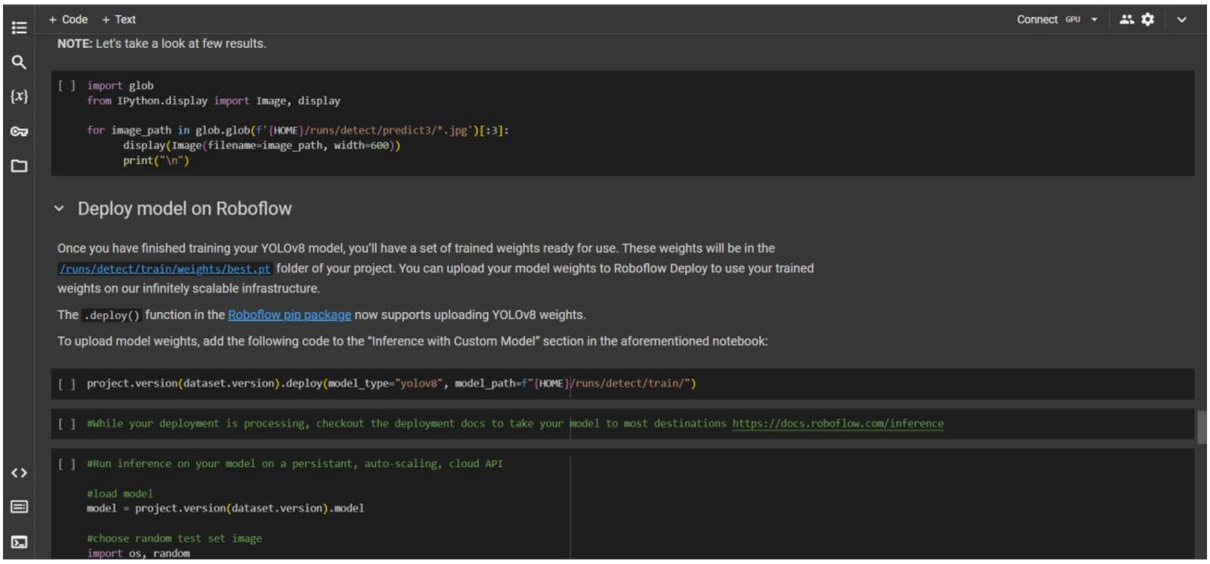
```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/val_batch0_pred.jpg', width=600)
```

Validate Custom Model

```
%cd {HOME}

!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml
```

Inference with Custom Model

```
%cd {HOME}
!yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt conf=0.25 source={dataset.location}/test/images save=True
```



NOTE: Let's take a look at few results.

```
import glob
from IPython.display import Image, display

for image_path in glob.glob(f'{HOME}/runs/detect/predict3/*.jpg')[:3]:
     display(Image(filename=image_path, width=600))
     print("\n")
```

Deploy model on Roboflow

Once you have finished training your YOLOv8 model, you'll have a set of trained weights ready for use. These weights will be in the `/runs/detect/train/weights/best.pt` folder of your project. You can upload your model weights to Roboflow Deploy to use your trained weights on our infinitely scalable infrastructure.

The `.deploy()` function in the Roboflow pip package now supports uploading YOLOv8 weights.

To upload model weights, add the following code to the "Inference with Custom Model" section in the aforementioned notebook:

```
project.version(dataset.version).deploy(model_type="yolov8", model_path=f"{HOME}/runs/detect/train/")
```

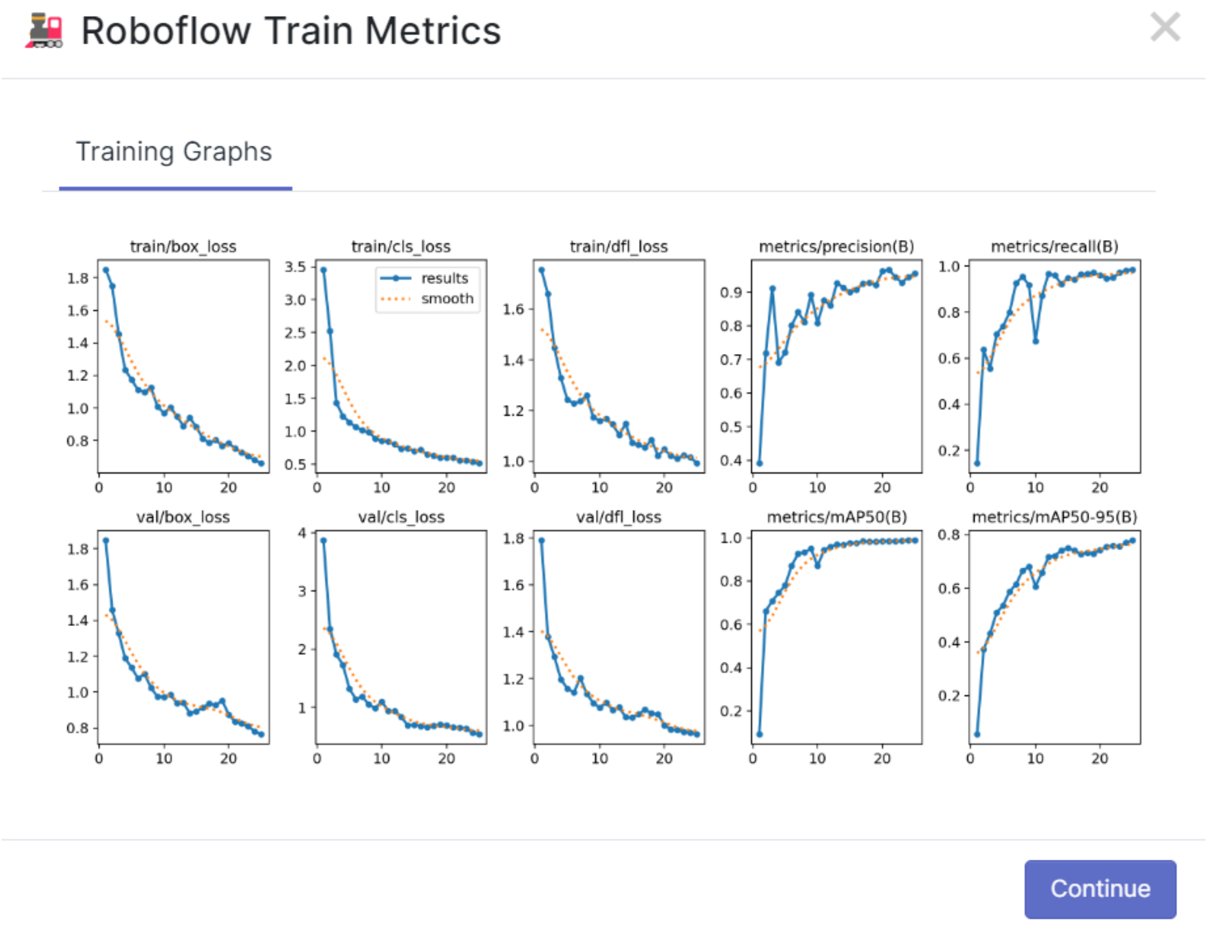
```
#While your deployment is processing, checkout the deployment docs to take your model to most destinations https://docs.roboflow.com/inference
```

```
#Run inference on your model on a persistant, auto-scaling, cloud API

#load model
model = project.version(dataset.version).model

#choose random test set image
import os, random
```

➢ ASSESS THE PERFORMANCE OF THE MODEL

To make sure the model generalizes well to new data, assess its performance on the validation set after training.

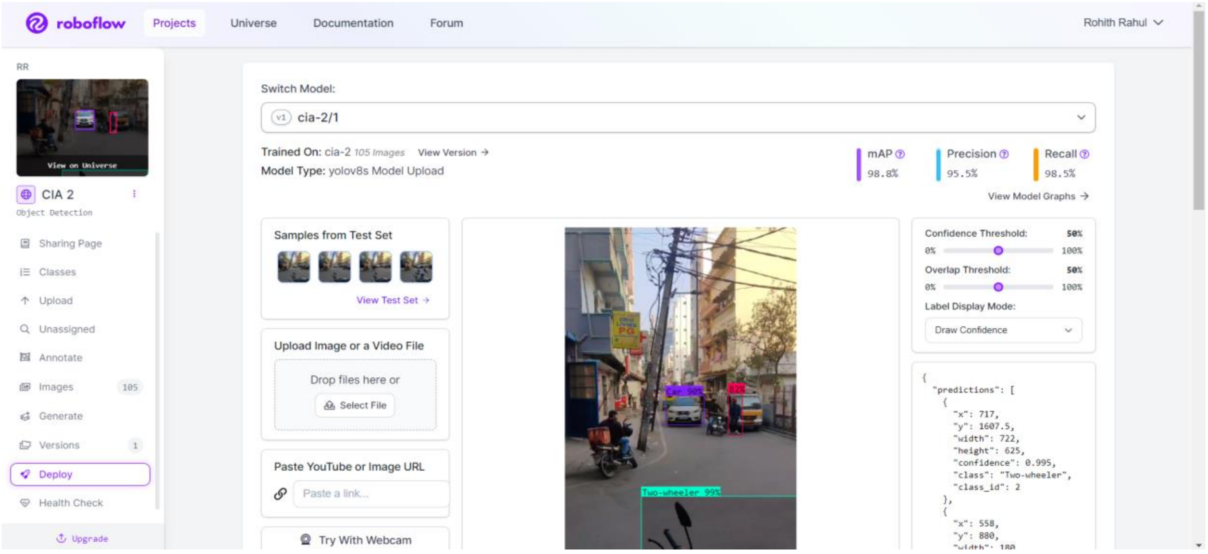
Roboflow facilitates the analysis of model performance by offering metrics and visualizations.



➢ CONCLUSION AND FORECAST:

Use the learned model to make inferences. Utilize the code that has been generated to forecast upcoming pictures or videos.

You can include the model into your application or use the deployment options Roboflow may offer.



Switch Model: cia-2/1

Trained On: cia-2 *105 Images*

Model Type: yolov8s Model Upload

mAP 98.8%

Precision 95.5%

Recall 98.5%

Confidence Threshold: 50% (0% – 100%)

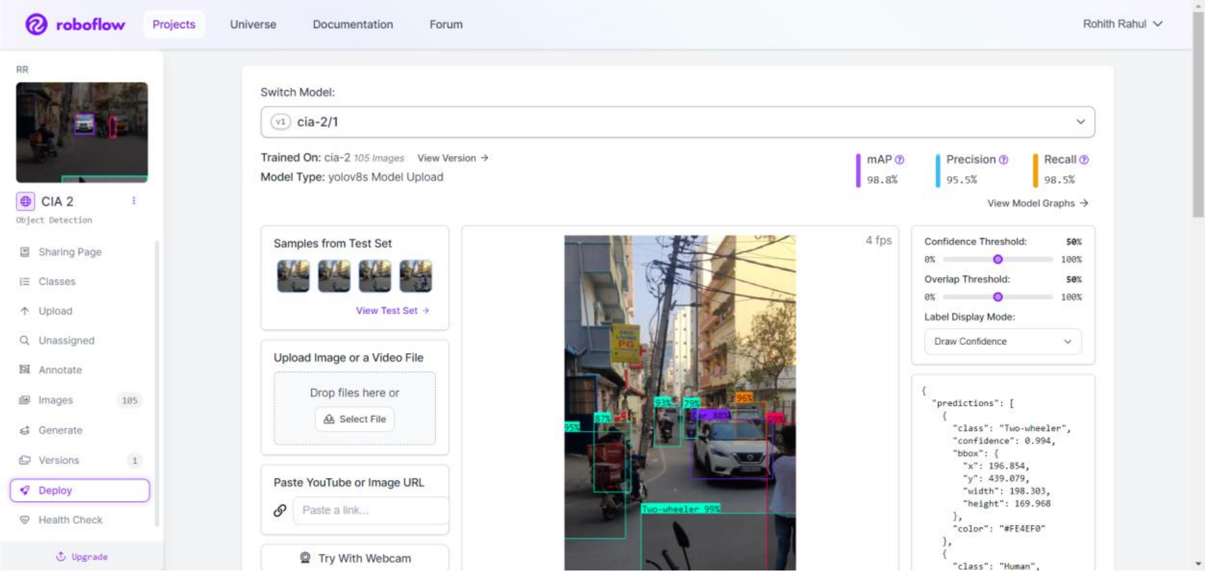
Overlap Threshold: 50% (0% – 100%)

Label Display Mode: Draw Confidence

```
{
  "predictions": [
    {
      "x": 717,
      "y": 1607.5,
      "width": 722,
      "height": 625,
      "confidence": 0.995,
      "class": "Two-wheeler",
      "class_id": 2
    },
    {
      "x": 558,
      "y": 880,
      "width": 180
```

➢ MODEL DEPLOYMENT

Utilize the offered deployment options or integrate the model into your own application to utilize it in real-world scenarios.



Switch Model: v1 cia-2/1

Trained On: cia-2 *105 Images*

Model Type: yolov8s Model Upload

mAP 98.8% | Precision 95.5% | Recall 98.5%

Samples from Test Set

Upload Image or a Video File

Drop files here or

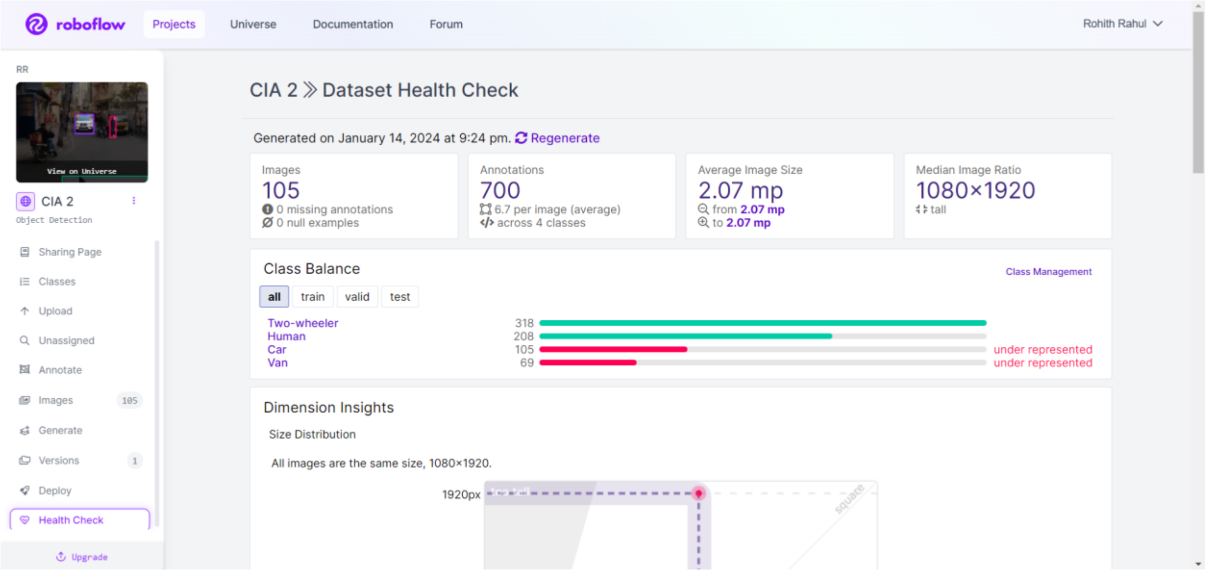
Paste YouTube or Image URL

Confidence Threshold: 50% (0% – 100%)

Overlap Threshold: 50% (0% – 100%)

Label Display Mode: Draw Confidence

```
{
  "predictions": [
    {
      "class": "Two-wheeler",
      "confidence": 0.994,
      "bbox": {
        "x": 196.854,
        "y": 439.079,
        "width": 198.303,
        "height": 169.968
      },
      "color": "#FE4EF0"
    },
    {
      "class": "Human",
```



CIA 2 » Dataset Health Check

Generated on January 14, 2024 at 9:24 pm.

Images	Annotations	Average Image Size	Median Image Ratio
105	700	2.07 mp	1080×1920
0 missing annotations	6.7 per image (average)	from 2.07 mp	tall
0 null examples	across 4 classes	to 2.07 mp	

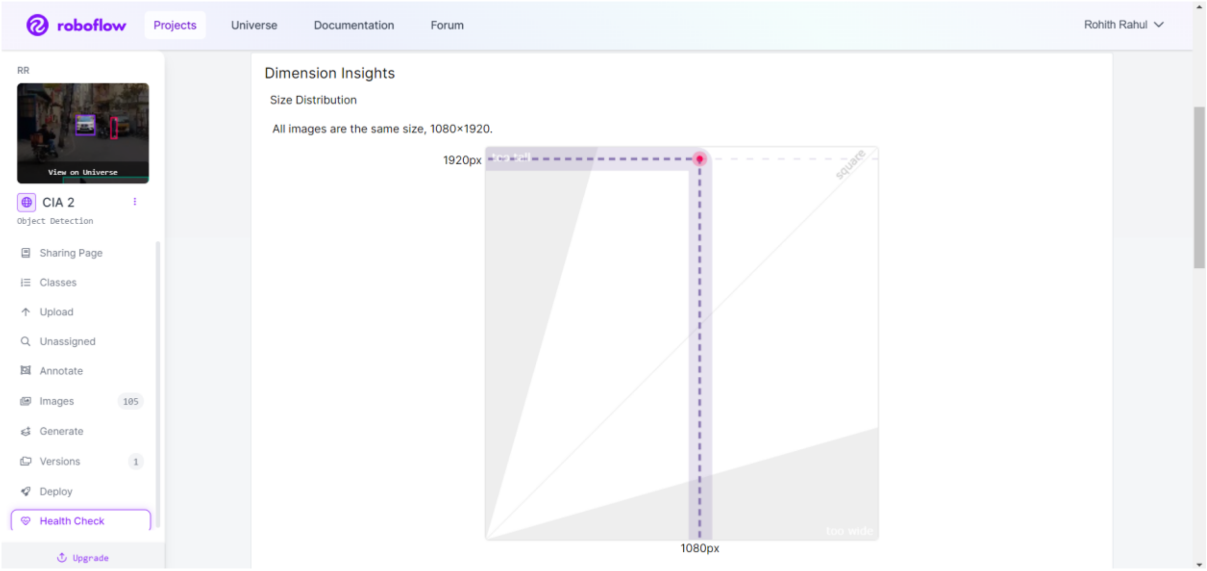
Class Balance

Two-wheeler	318	
Human	208	
Car	105	under represented
Van	69	under represented

Dimension Insights

Size Distribution

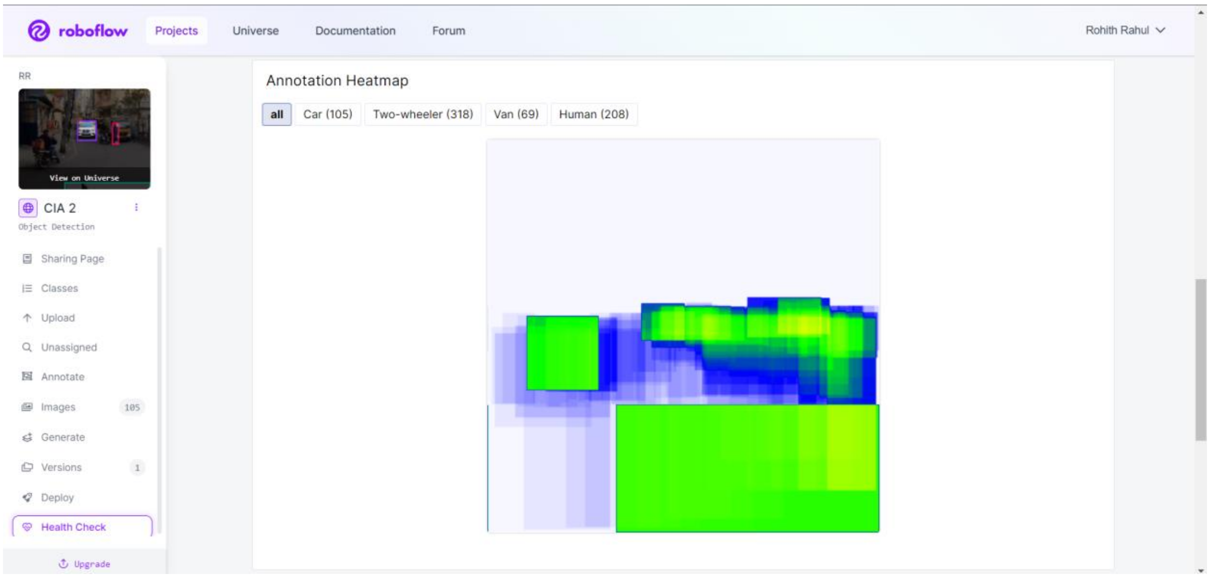
All images are the same size, 1080×1920.



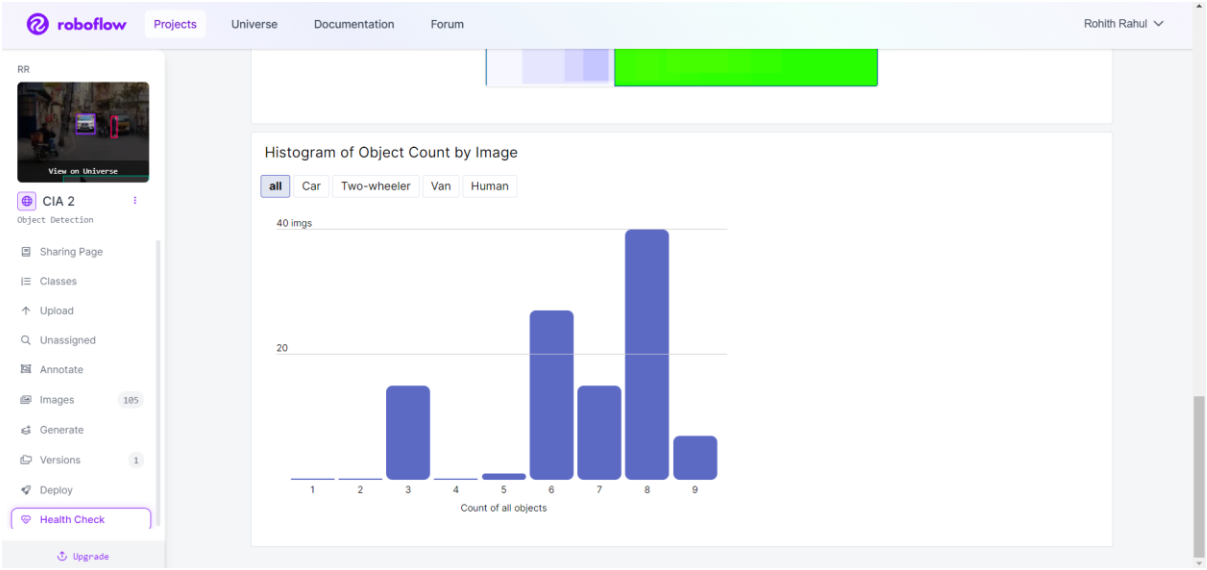
Dimension Insights

Size Distribution

All images are the same size, 1080×1920.

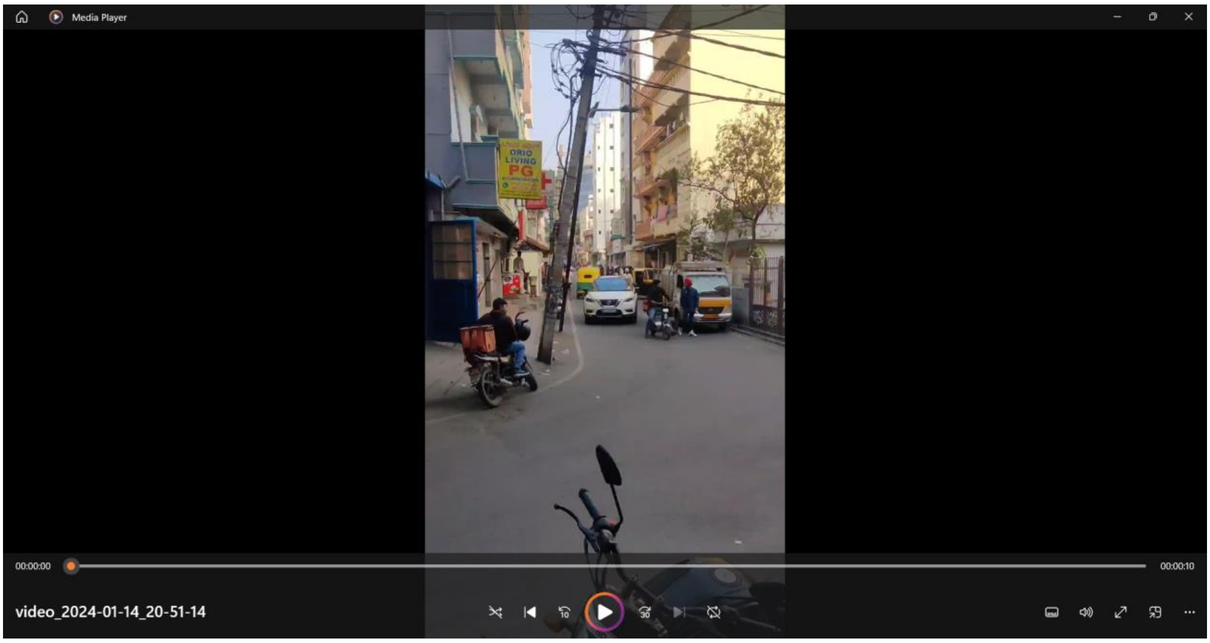


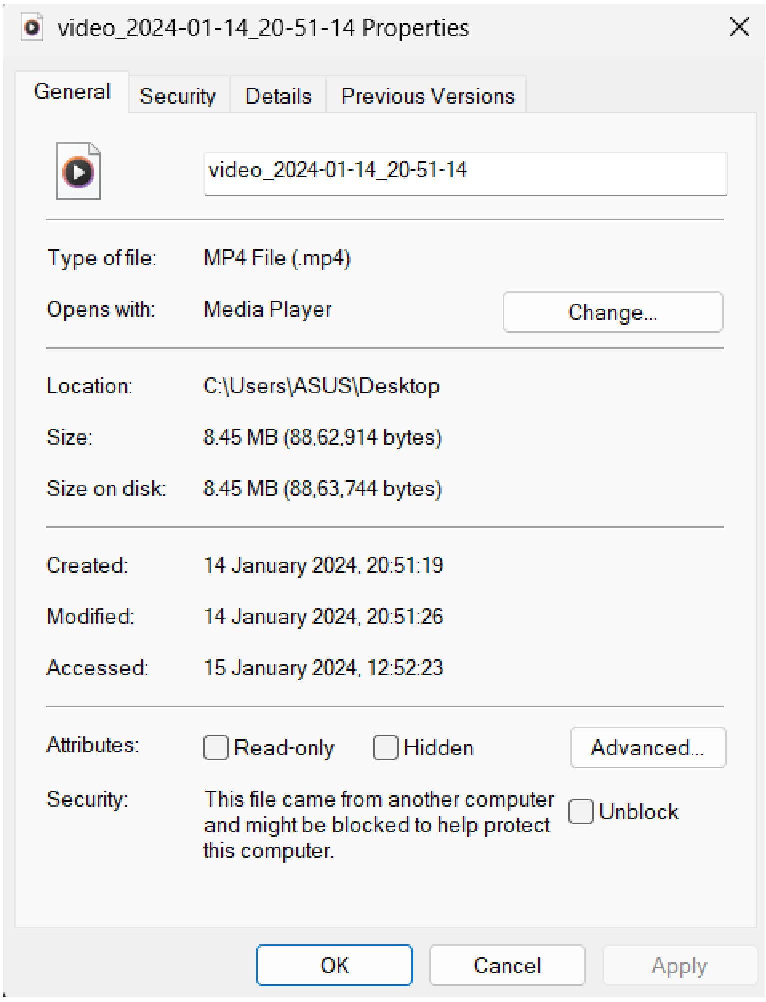
Annotation Heatmap



Histogram of Object Count by Image

INPUT VIDEO FILE




video_2024-01-14_20-51-14 Properties

video_2024-01-14_20-51-14

Type of file: MP4 File (.mp4)

Opens with: Media Player

Location: C:\Users\ASUS\Desktop

Size: 8.45 MB (88,62,914 bytes)

Size on disk: 8.45 MB (88,63,744 bytes)

Created: 14 January 2024, 20:51:19

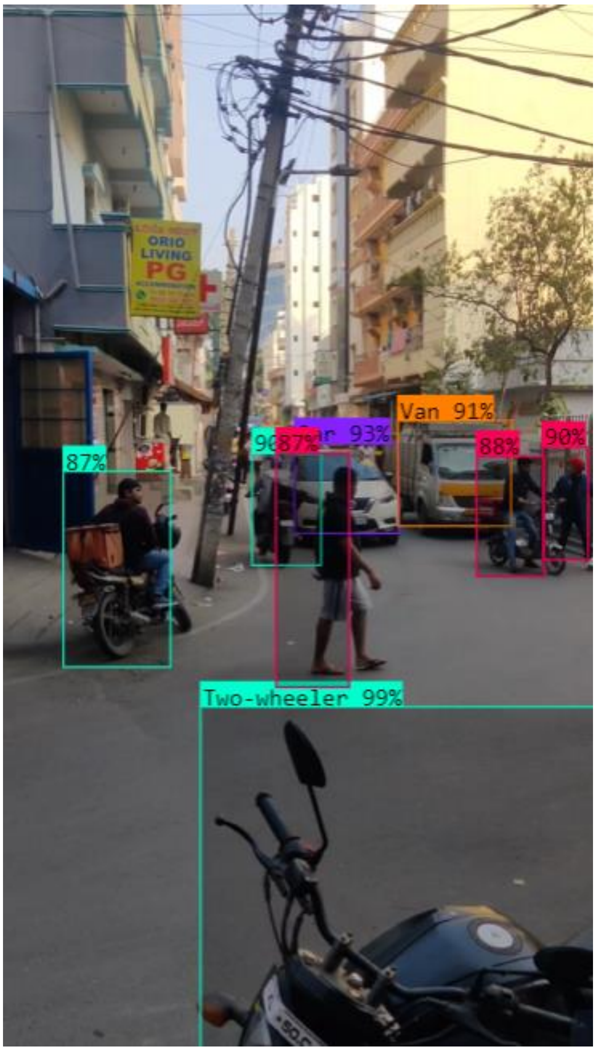
Modified: 14 January 2024, 20:51:26

Accessed: 15 January 2024, 12:52:23

Attributes: □ Read-only □ Hidden

Security: This file came from another computer and might be blocked to help protect this computer. □ Unblock

OUTPUT



REFERENCES

- Yolo and Object Detection: https://www.superannotate.com/blog/yolo-object-detection#:~:text=The%20YOLO%20algorithm%2C%20which%20stands,forward%20pass%20of%20the%20network.

- Roboflow: https://medium.com/red-buffer/roboflow-d4e8c4b52515

- Roboflow procedure: https://blog.roboflow.com/getting-started-with-roboflow/

QUICK LINKS

Roboflow project: https://universe.roboflow.com/rr-r1mr5/cia-2

Try on mobile

THANK YOU