Abstract

Bike rider helmet detection is a crucial computer vision task aimed at improving road safety by identifying bike rider who are not wearing helmets. This system leverages deep learning models, such as YOLOv8, to accurately detect helmets in real-time or from images or video streams. It can be integrated into traffic monitoring systems to automatically flag violations and enhance law enforcement efficiency. The implementation involves preprocessing images, training the model, and saving the model. I have also created a streamlit and django web application and deployed the web application in streamlit cloud. The proposed system helps reduce head injuries and fatalities in bike rider accidents by promoting helmet use.

Index

Chapter 1

1. Introduction

1.1) Computer Vison: Introduction

Computer Vision is an interdisciplinary field of study that enables computers to interpret and understand visual information from the world, much like the human visual system. It encompasses the development of algorithms, models, and systems that can process, analyze, and extract meaningful insights from visual data, typically in the form of images and videos. Computer Vision has wide-ranging applications across various industries, including healthcare, automotive, entertainment, surveillance, robotics, and more. This overview provides a comprehensive understanding of Computer Vision, its key components, applications, challenges, and future prospects.

1.2) Bike Rider Helmet Detection: Introduction

The importance of road safety has been increasingly recognized, leading to the implementation of various measures to protect vulnerable road users such as cyclists and motorcyclists or bike riders. One critical safety measure is the use of helmets, which significantly reduces the risk of head injuries in the event of an accident. Despite regulations mandating helmet use, compliance remains a challenge. To address this, the Bike Rider Helmet Detection project aims to leverage advanced computer vision techniques to automatically detect helmet usage among bike riders and passangers.

The Bike Rider Helmet Detection project utilizes one of the state-of-the-art deep learning models to accurately identify whether a bike rider is wearing a helmet. The core of the project is built on the YOLOv8 (You Only Look Once version 8) model, a highly efficient and powerful object detection algorithm developed and maintained by Ultralytics. YOLOv8 is renowned for its speed and accuracy, making it an ideal choice for real-time applications like helmet detection.

There are a significant number of the accidents on the roads today. The accident can have multiple scenarios whether due to pot holes or any car accidentally bumbs bike driver while driving etc. By wearing a helmet not only we are saving our head from any fatal accidents but also saving any passanger life too.

This explains why it is important to do more work in this field with an aim to reduce the occurrence of accidents related to any driving related injury and motivate overselves the importance of a helmet while riding a 2 wheeler vehicle.

Chapter 2

2. Objective

2.1) Problem:

Bike riders drivers who do not wear helmet which may result in fatal accidents and death in some cases.

2.2) Goal:

Create a deep learning model that an detect if a person is wearing helmet or not.

The main objective of this project is to develop a system that would detect if the bike rider along with the passengers are wearing a safety helmet or not. The dataset is collected through kaggle dataset. As manually annotation of the labels or semi label annotation would take significant amount of time. Only images are needed and and no explicit annotations are needed for this project as I am using ultralytics foundation models for auto labelling the images and video images to target model of YOLOv8. After training the model I have made a streamlit and django web application for this project along with documentation.

Chapter 3

3. Tools and Technology

As this is a deep learning project a significant amount of computation and memory allocation does matter a lot for this project. For inference of model on images and videos too required a significant amount of computation power.

3.1) Hardware Requirements :-

1) Desktop / Laptop / Server
2) 8 GB RAM at least
3) 150 GB Disk space or higher
4) Any processor Intel i5 / AMD
5) Google GPU - Tesla T4

3.2) Sofware Requirements :-

1) Windows / Ubuntu os (64 or 32 bit)
2) Google colab / Kaggle jupyter notebook
3) Python 3.10.12 or higher
4) Visual studio code editor, jupyter notebook
5) Sqlite version 0.5.6 or higher

For more in depth requirements of the major, minor packages and respective project dependencies. Please take a look at the respective github repository.

Chapter 4

4. Literature Review

Key Components of Computer Vision:

4.1) Image Acquisition:

Computer Vision begins with the acquisition of visual data, which can come from various sources, including cameras, sensors, or image databases.

4.2) Image Preprocessing:

Raw visual data often requires preprocessing to enhance quality, remove noise, and prepare it for analysis. This includes tasks like image resizing, filtering, and color correction.

4.3) Feature Extraction:

Feature extraction involves identifying and isolating relevant visual patterns or features within an image, such as edges, corners, or texture.

4.4) Object Detection and Recognition:

This component focuses on identifying and classifying objects within images or videos. Object detection and recognition algorithms enable computers to recognize and label objects, faces, or specific patterns.

4.5) Image Segmentation:

Image segmentation divides an image into meaningful regions or segments. It's crucial for tasks like medical image analysis, where different parts of an image may represent different anatomical structures.

4.6) Motion Analysis:

Motion analysis techniques track moving objects and can be used in applications like surveillance, sports analysis, and robotics.

4.7) Scene Understanding:

This involves higher-level interpretation of images or videos to understand the context and relationships between objects within a scene.

Chapter 5

5. Methodology

5.1) Data Collection:

Collect publicly available images / videos of bike rider, helmet and no helmet for the model to train upon. The images are collected from kaggle dataset

5.2) Data pre-processing:

Pre-processing and auto labeling images and videos are done by ultralytics framework by using a Grounding Segment Anything Model based on Ontology. For videos every 3 frame is considered.

5.3) Model Training:

Train YOLOv8 Model on custom dataset with 3 classes, Socastic Gradient Decent (SGD) optimizer, learning rate 0.01, batch size 16, epochs 50, image size 640, momentum 0.937, validation split around 20 %.

5.4) Model Evaluation:

Evaluate trained YOLOv8 Model on validation dataset with the metrics such as box loss, class loss, precision and recall. Confusion matrix.

5.5) Model Testing:

Run Inference on images and videos. To check how well the model perform.

5.6) Deployment:

The final model is deployed as a streamlit web application and django for local deployment along with application programming interface (API).

Chapter 6

6. Implementation details

6.1) Dataset Used:

The dataset that is used for this project can be find below:-

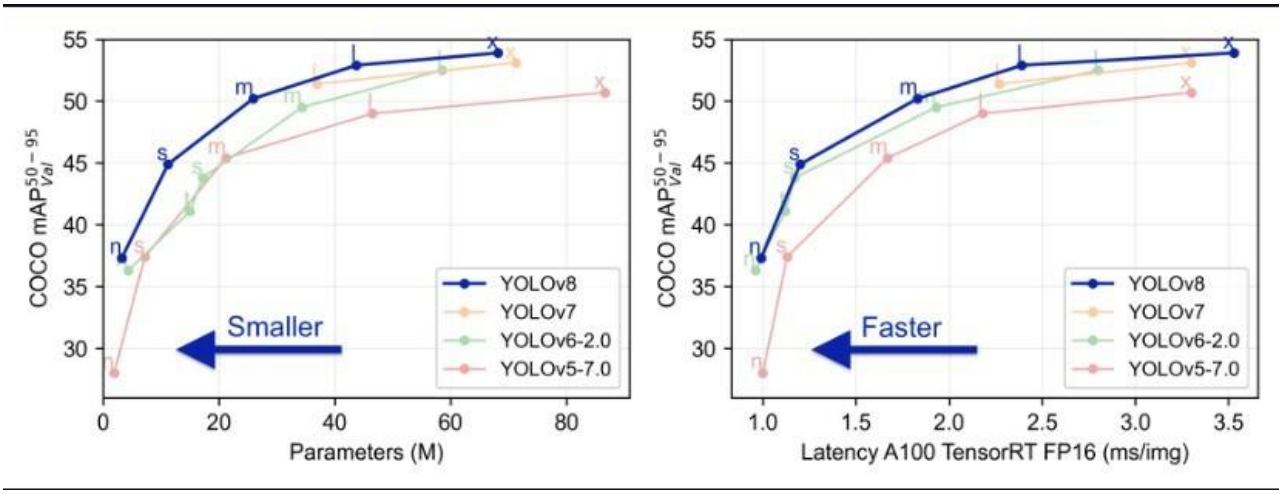
https://www.kaggle.com/datasets/andrewmvd/helmet-detection

This dataset contains 764 images of 2 distinct classes for the objective of helmet detection.
Bounding box annotations are provided in the PASCAL VOC format.

The classes are:

- With helmet
- Without helmet

6.2) YOLOv8 Metrics



6.3) What is YOLOv8?

YOLOv8 is from the YOLO family of models and was released on January 10, 2023. YOLO stands for You Only Look Once, and this series of models are thus named because of their ability to predict every object present in an image with one forward pass.

6.4) Why YOLOv8?

YOLOv8 by ultalytics is a state-of-the-art deep learning model designed for real-time object detection in computer vision applications. With its advanced architecture and cutting-edge algorithms, YOLOv8 has revolutionized the field of object detection, enabling accurate and efficient detection of objects in real-time scenarios.

YOLOv8 is quite stable as compare to latest YOLOv9 and recent YOLOv10.

6.5) What is Ultralytics?

Ultralytics is a company that specializes in developing advanced computer vision technologies, particularly the YOLO (You Only Look Once) series of models. Their framework, Ultralytics YOLO, provides state-of-the-art object detection, image segmentation, and classification capabilities. The base model, often a pre-trained YOLOv8 model or it can be any model like YOLOv5, can be fine-tuned or used directly to create custom target models for various applications for any custom dataset. This framework is widely used for real-time object detection tasks due to its speed and accuracy.

6.6) What are base models?

Base Model - A Base Model is a large foundation model that knows a lot about a lot. Base models are often multimodal and can perform many tasks. They're large, slow, and expensive. Examples of Base Models are GroundedSAM and GPT-4's upcoming multimodal variant. We use a Base Model (along with unlabeled input data and an Ontology) to create a Dataset.

6.7) What are target models?

Target Model - a Target Model is a supervised model that consumes a dataset and outputs a distilled model that is ready for deployment. Target Models are usually small, fast, and fine-tuned to perform a specific task very well (but they don't generalize well beyond the information described in their Dataset). Examples of Target Models are YOLOv8 and YOLOv5.

6.8) How pre-processing of the dataset is done?

Only images and videos are needed for this project and no annotations are needed from the data as annotation are generated by ultralytics framework called as **Autodistill** which uses ***Grounding SAM which is combination of Grounding DiNO (Zero short object detection model) and SAM (Segment Anything Model) (zero short object detection with prompting) from Meta for autolabeling dataset and preprocess and train on YOLOv8 large model***.

Autodistill uses big, slower foundation models to train small, faster supervised models. Using autodistill, you can go from unlabeled images to inference on a custom model running at the edge with no human intervention in between.

As foundation models get better and better they will increasingly be able to augment or replace humans in the labeling process.

6.9) How images are pre-processed?

The dataset image consist of 764 images of various Bike rider, Rider wearing helmet, Rider not wearing helmet. Out of 764 images only 611 images are used. I have removed some images because my google colab kernel is crashing significantly.

6.10) How videos are pre-processed?

Video is processed is such a way in which every 3 frame (can be changed through code) are considered as image data for the model to train upon.

For autolabelling the dataset of images and video frame autodistill uses something which is called as ontology

6.11) What is Ontology?

Ontology - an Ontology defines how your Base Model like Grounding SAM is prompted, what your Dataset will describe, and what your Target Model like YOLOv8 will predict. A simple Ontology is the CaptionOntology which prompts a Base Model with text captions and maps them to class names. Other Ontologies may, for instance, use a CLIP vector or example images instead of a text caption.

```python
from autodistill.detection import CaptionOntology

    # "<description of label>": "<label_name>"
    # "bike rider": "Bike_Rider", --> label 0
    # "bike rider and passanger with helmet": "Helmet", --> label 1
    # "bike rider and passanger with no helmet": "No_Helmet" --> label 2

ontology=CaptionOntology({
    "bike rider": "Bike_Rider",
    "helmet": "Helmet",
    "no helmet": "No_Helmet"
})
```

The preprocessed dataset is then divided into training and validation dataset to check the performance of model.

This YOLOv8 model is fine-tuned for custom dataset target values of **bike rider**, **helmet** and **no helme**t. Based on which best.pt and last.pt pytorch training weights are generated.

6.12) Sample Code

```
import supervision as sv

image_paths = sv.list_files_with_extensions(
    directory=IMAGE_DIR_PATH,
    extensions=["png", "jpg", "jpg"])

print('image count:', len(image_paths))
```

```
image count: 764
```

```
from autodistill_grounded_sam import GroundedSAM

base_model = GroundedSAM(ontology=ontology)
dataset = base_model.label(
    input_folder=IMAGE_DIR_PATH,
    extension=".png",
    output_folder=DATASET_DIR_PATH)
```

```
trying to load grounding dino directly
downloading dino model weights
```

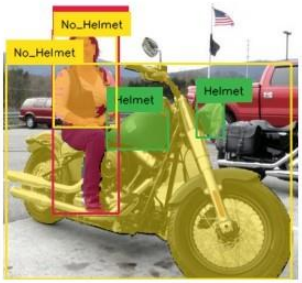
```
import supervision as sv

dataset = sv.DetectionDataset.from_yolo(
    images_directory_path=IMAGES_DIRECTORY_PATH,
    annotations_directory_path=ANNOTATIONS_DIRECTORY_PATH,
    data_yaml_path=DATA_YAML_PATH)

len(dataset)
```

```
611
```

```python
import supervision as sv

image_names = list(dataset.images.keys())[:SAMPLE_SIZE]

mask_annotator = sv.MaskAnnotator()
box_annotator = sv.BoxAnnotator()

images = []
for image_name in image_names:
    image = dataset.images[image_name]
    annotations = dataset.annotations[image_name]
    labels = [
        dataset.classes[class_id]
        for class_id
        in annotations.class_id]
    annotates_image = mask_annotator.annotate(
        scene=image.copy(),
        detections=annotations)
    annotates_image = box_annotator.annotate(
        scene=annotates_image,
        detections=annotations,
        labels=labels)
    images.append(annotates_image)

sv.plot_images_grid(
    images=images,
    titles=image_names,
    grid_size=SAMPLE_GRID_SIZE,
    size=SAMPLE_PLOT_SIZE)
```

BikesHelmets0.jpg

BikesHelmets412.jpg



BikesHelmets721.jpg

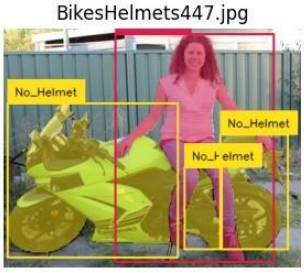
BikesHelmets511.jpg

BikesHelmets115.jpg

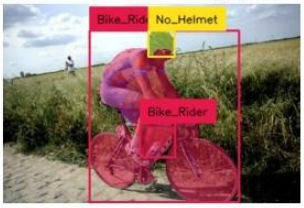
BikesHelmets611.jpg

BikesHelmets719.jpg

BikesHelmets447.jpg



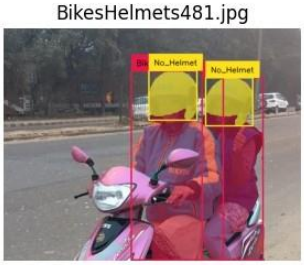
BikesHelmets290.jpg



BikesHelmets522.jpg

BikesHelmets394.jpg

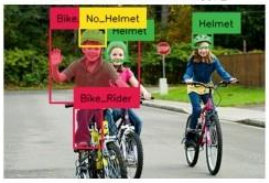
BikesHelmets481.jpg



BikesHelmets172.jpg

BikesHelmets665.jpg

BikesHelmets433.jpg



BikesHelmets405.jpg

```
%cd {HOME}

from autodistill_yolov8 import YOLOv8

target_model = YOLOv8("yolov8l.pt")
target_model.train(DATA_YAML_PATH, epochs=50) #100
```

```
/content
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8l.pt to yolov8l.pt...
100%|██████████| 83.7M/83.7M [00:00<00:00, 196MB/s]
New https://pypi.org/project/ultralytics/8.0.196 available 😃 Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.0.81 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
yolo/engine/trainer: task=detect, mode=train, model=yolov8l.pt, data=/content/dataset/data.yaml, epochs=50, patience=50, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=None, exist_ok=False, pretrained=False, optimizer=SGD, verbose=True, seed=0, deterministic=True, single_cls=False, image_weights=False, rect=False, cos_lr=False, close_mosaic=0, resume=False, amp=True, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_hybrid=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, show=False, save_txt=False, save_conf=False, save_crop=False, show_labels=True, show_conf=True, vid_stride=1, line_thickness=3, visualize=False, augment=False, agnostic_nms=False, classes=None, retina_masks=False, boxes=True, format=torchscript, keras=False, optimize=False, int8=False, dynamic=False, simplify=False, opset=None, workspace=4, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0, cfg=None, v5loader=False, tracker=botsort.yaml, save_dir=runs/detect/train
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100%|██████████| 755k/755k [00:00<00:00, 15.5MB/s]
Overriding model.yaml nc=80 with nc=3
```

```
                   from  n    params  module                                       arguments
  0                  -1  1      1856  ultralytics.nn.modules.Conv                  [3, 64, 3, 2]
  1                  -1  1     73984  ultralytics.nn.modules.Conv                  [64, 128, 3, 2]
  2                  -1  3    279808  ultralytics.nn.modules.C2f                   [128, 128, 3, True]
  3                  -1  1    295424  ultralytics.nn.modules.Conv                  [128, 256, 3, 2]
  4                  -1  6   2101248  ultralytics.nn.modules.C2f                   [256, 256, 6, True]
  5                  -1  1   1180672  ultralytics.nn.modules.Conv                  [256, 512, 3, 2]
  6                  -1  6   8396800  ultralytics.nn.modules.C2f                   [512, 512, 6, True]
  7                  -1  1   2360320  ultralytics.nn.modules.Conv                  [512, 512, 3, 2]
  8                  -1  3   4461568  ultralytics.nn.modules.C2f                   [512, 512, 3, True]
  9                  -1  1    656896  ultralytics.nn.modules.SPPF                  [512, 512, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.Concat                [1]
 12                  -1  3   4723712  ultralytics.nn.modules.C2f                   [1024, 512, 3]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14            [-1, 4]  1         0  ultralytics.nn.modules.Concat                [1]
 15                  -1  3   1247744  ultralytics.nn.modules.C2f                   [768, 256, 3]
 16                  -1  1    590336  ultralytics.nn.modules.Conv                  [256, 256, 3, 2]
 17           [-1, 12]  1         0  ultralytics.nn.modules.Concat                [1]
 18                  -1  3   4592640  ultralytics.nn.modules.C2f                   [768, 512, 3]
 19                  -1  1   2360320  ultralytics.nn.modules.Conv                  [512, 512, 3, 2]
 20            [-1, 9]  1         0  ultralytics.nn.modules.Concat                [1]
 21                  -1  3   4723712  ultralytics.nn.modules.C2f                   [1024, 512, 3]
 22       [15, 18, 21]  1   5585113  ultralytics.nn.modules.Detect                [3, [256, 512, 512]]
Model summary: 365 layers, 43632153 parameters, 43632137 gradients, 165.4 GFLOPs
```

```
Transferred 589/595 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/train', view at http://localhost:6006/
AMP: running Automatic Mixed Precision (AMP) checks with YOLOv8n...
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8n.pt to yolov8n.pt...
100%|██████████| 6.23M/6.23M [00:00<00:00, 76.4MB/s]
AMP: checks passed ✅
optimizer: SGD(lr=0.01) with parameter groups 97 weight(decay=0.0), 104 weight(decay=0.0005), 103 bias
train: Scanning /content/dataset/train/labels... 611 images, 1 backgrounds, 0 corrupt: 100%|██████████| 611/611
[00:00<00:00, 985.48it/s]
train: WARNING ⚠ /content/dataset/train/images/BikesHelmets247.jpg: 2 duplicate labels removed
train: WARNING ⚠ /content/dataset/train/images/BikesHelmets351.jpg: 1 duplicate labels removed
train: WARNING ⚠ /content/dataset/train/images/BikesHelmets454.jpg: 1 duplicate labels removed
train: WARNING ⚠ /content/dataset/train/images/BikesHelmets564.jpg: 1 duplicate labels removed
train: WARNING ⚠ /content/dataset/train/images/BikesHelmets574.jpg: 1 duplicate labels removed
train: WARNING ⚠ /content/dataset/train/images/BikesHelmets684.jpg: 1 duplicate labels removed
train: WARNING ⚠ /content/dataset/train/images/BikesHelmets687.jpg: 1 duplicate labels removed
train: New cache created: /content/dataset/train/labels.cache
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=
0.01, clip_limit=(1, 4.0), tile_grid_size=(8, 8))
val: Scanning /content/dataset/valid/labels... 153 images, 2 backgrounds, 0 corrupt: 100%|██████████| 153/153 [0
0:00<00:00, 525.87it/s]
val: WARNING ⚠ /content/dataset/valid/images/BikesHelmets164.jpg: 1 duplicate labels removed
val: New cache created: /content/dataset/valid/labels.cache
Plotting labels to runs/detect/train/labels.jpg...
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/detect/train
Starting training for 50 epochs...
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/50      13.2G       1.91      3.525      1.858         41        640: 100%|██████████| 39/39 [00:41<00:
00,  1.06s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 5/5
[00:05<00:00,  1.06s/it]
                   all        153        776      0.334      0.302      0.226      0.114

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/50      14.2G      1.175      1.963      1.318         17        640: 100%|██████████| 39/39 [00:33<00:
00,  1.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 5/5
[00:03<00:00,  1.37it/s]
                   all        153        776      0.428      0.503       0.41      0.255

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/50      14.1G     0.9655      1.571      1.167         34        640: 100%|██████████| 39/39 [00:33<00:
00,  1.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 5/5
[00:05<00:00,  1.04s/it]
                   all        153        776      0.451      0.397      0.383      0.233

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/50      14.2G     0.9835      1.434      1.156         31        640: 100%|██████████| 39/39 [00:33<00:
00,  1.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 5/5
[00:04<00:00,  1.04it/s]
                   all        153        776      0.469      0.516      0.482      0.305
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      49/50      14.2G     0.4481     0.3831     0.8859         37        640: 100%|██████████| 39/39 [00:33<00:
00,  1.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 5/5
[00:04<00:00,  1.13it/s]
                   all        153        776      0.813      0.675      0.752      0.608

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      50/50      14.1G      0.429     0.3665     0.8799         25        640: 100%|██████████| 39/39 [00:32<00:
00,  1.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 5/5
[00:08<00:00,  1.73s/it]
                   all        153        776      0.784      0.695      0.758      0.612

50 epochs completed in 0.590 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 87.6MB
Optimizer stripped from runs/detect/train/weights/best.pt, 87.6MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.81 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 268 layers, 43608921 parameters, 0 gradients, 164.8 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 5/5
[00:07<00:00,  1.55s/it]
                   all        153        776      0.783      0.693      0.758      0.612
            Bike_Rider        153        265       0.75      0.668      0.732      0.622
                Helmet        153        336      0.836      0.776      0.816      0.592
             No_Helmet        153        175      0.761      0.636      0.726      0.621
Speed: 1.4ms preprocess, 14.6ms inference, 0.0ms loss, 1.6ms postprocess per image
Results saved to runs/detect/train
```

6.13) Data and YOLOv8 configuration file

Open

data.yaml
~/Downloads/YOLOv8_Helmet_V0/dataset

```yaml
names:
- Bike_Rider
- Helmet
- No_Helmet
nc: 3
train: /content/dataset/train/images
val: /content/dataset/valid/images
```

Open

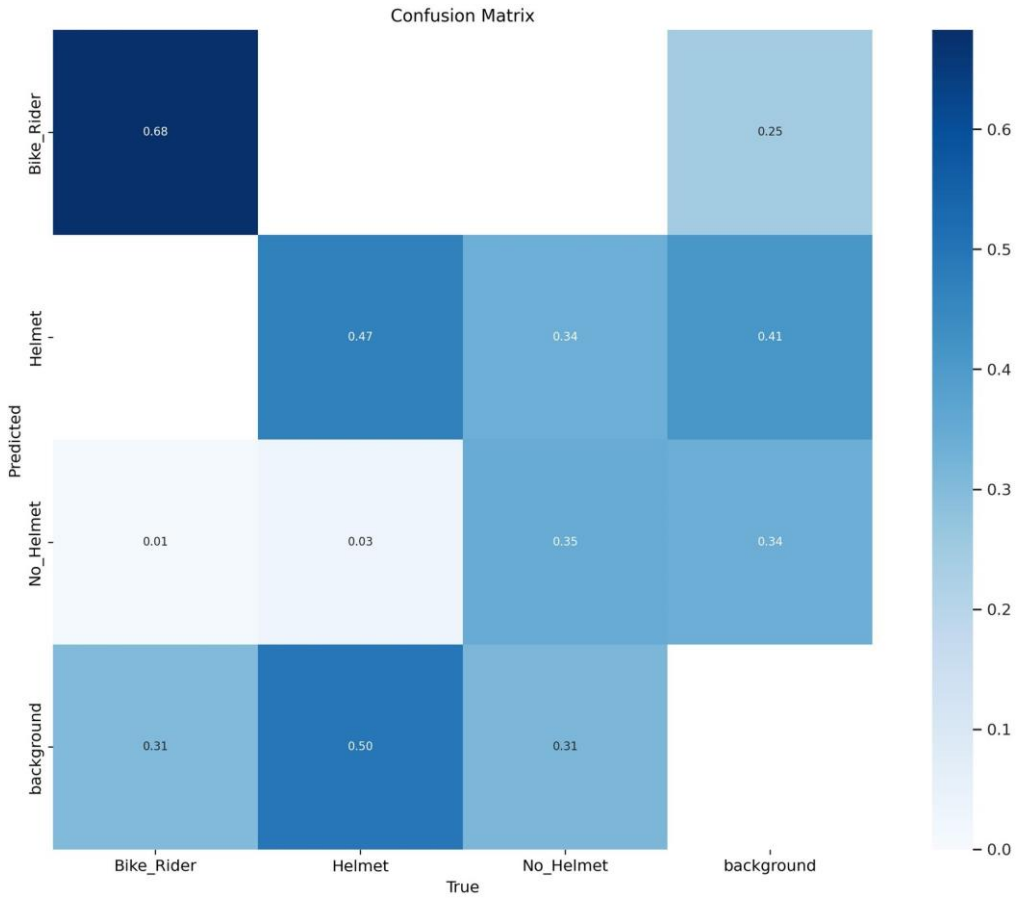
args.yaml
~/Downloads/YOLOv8_Helmet_V0/runs/detect/train

```yaml
task: detect
mode: train
model: yolov8l.pt
data: /content/dataset/data.yaml
epochs: 50
patience: 50
batch: 16
imgsz: 640
save: true
save_period: -1
cache: false
device: null
workers: 8
project: null
name: null
exist_ok: false
pretrained: false
optimizer: SGD
verbose: true
seed: 0
deterministic: true
single_cls: false
image_weights: false
rect: false
cos_lr: false
close_mosaic: 0
resume: false
amp: true
overlap_mask: true
mask_ratio: 4
dropout: 0.0
val: true
split: val
save_json: false
save_hybrid: false
conf: null
iou: 0.7
```

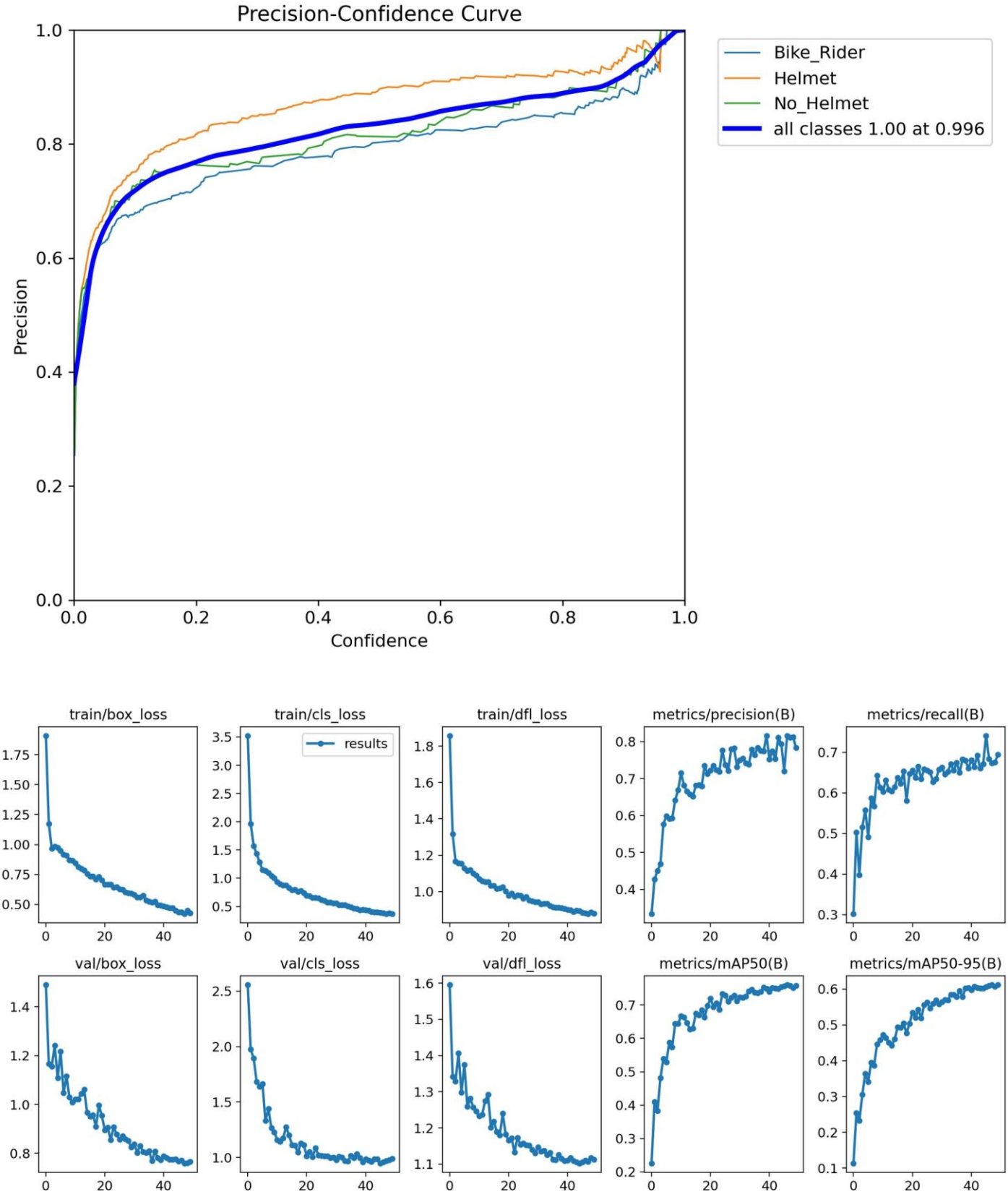
Chapter 7

7. Analysis of result

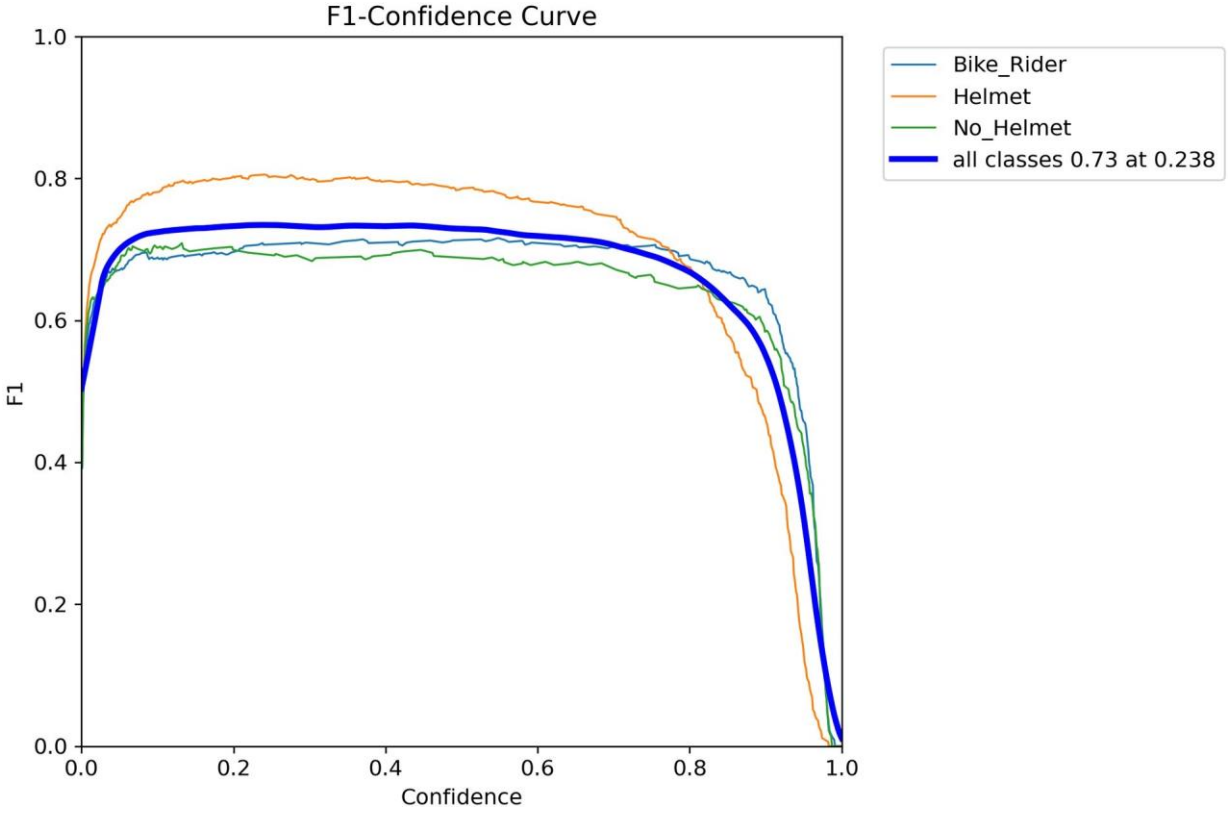
7.1) Confusion Matrix

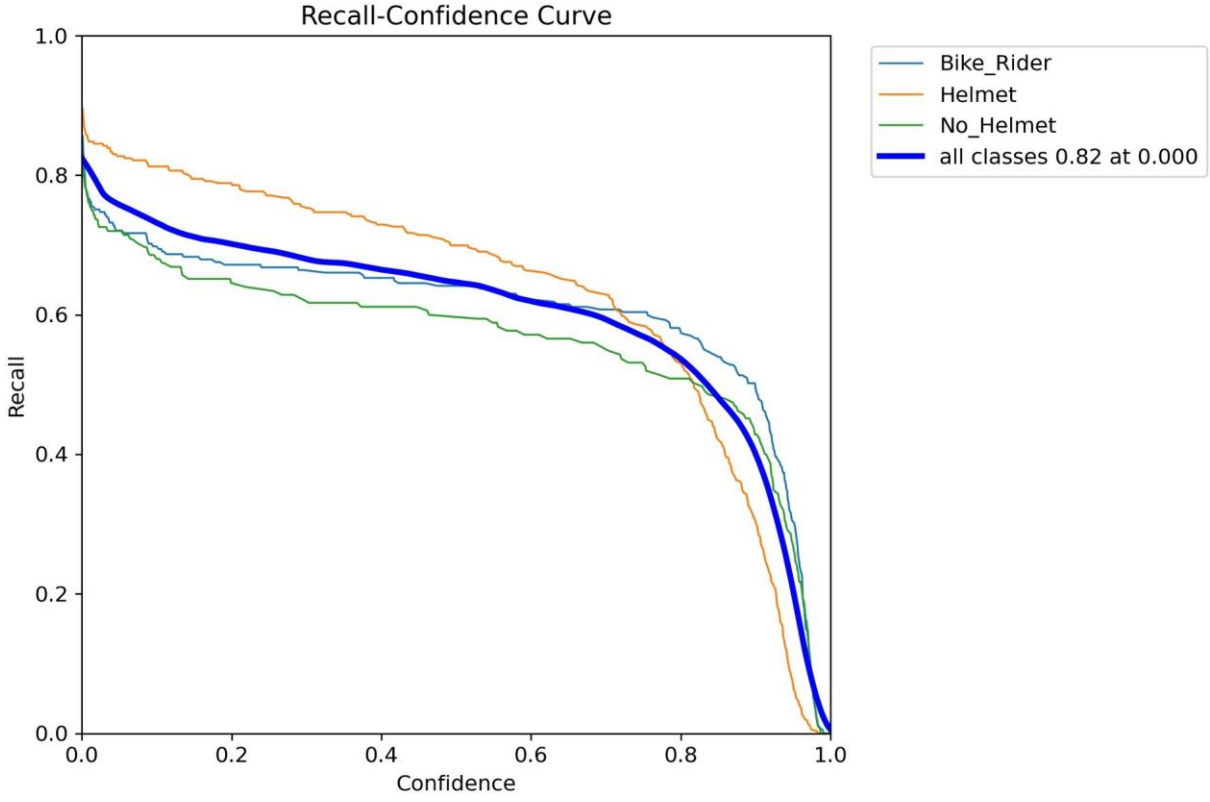


7.2) Precision-Confidence Curve and Train, Validation and Loss metrics

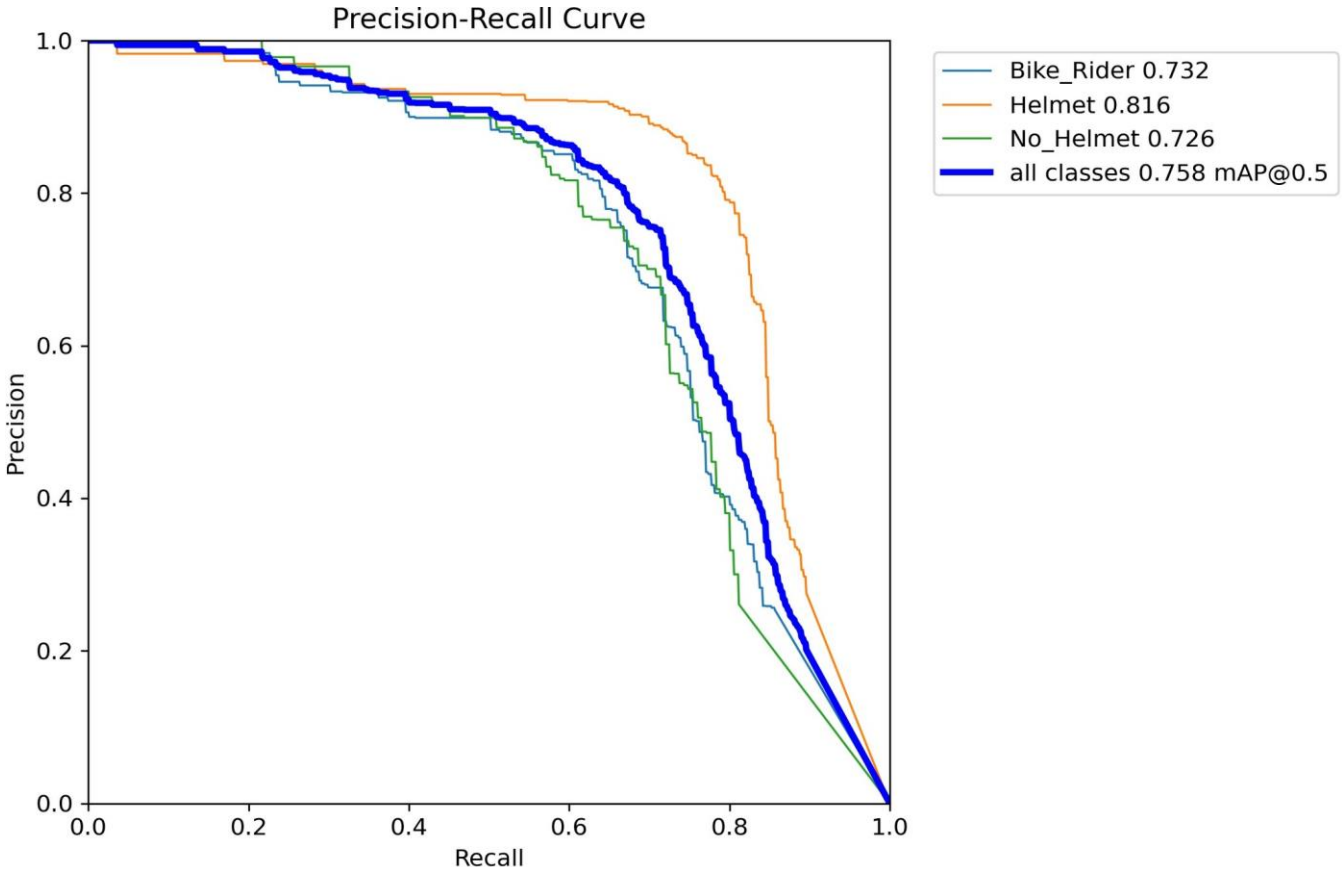


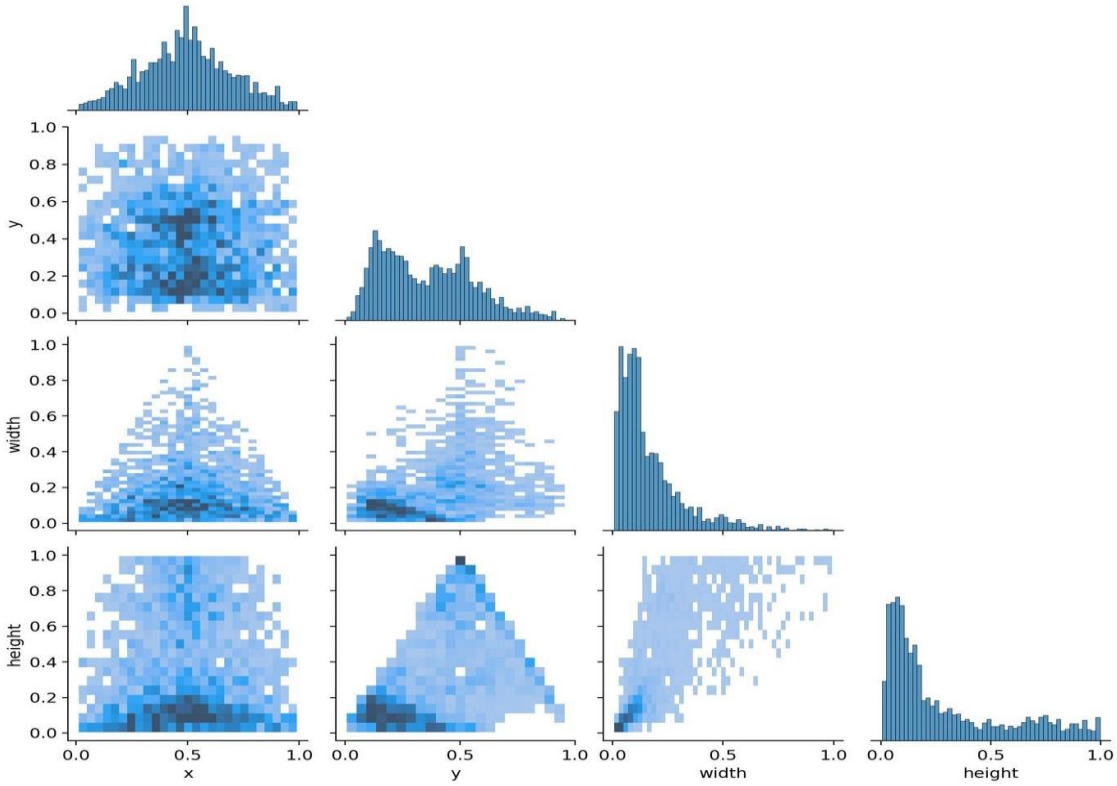
7.3) F1-Confidence and Recall-Confidence Curve



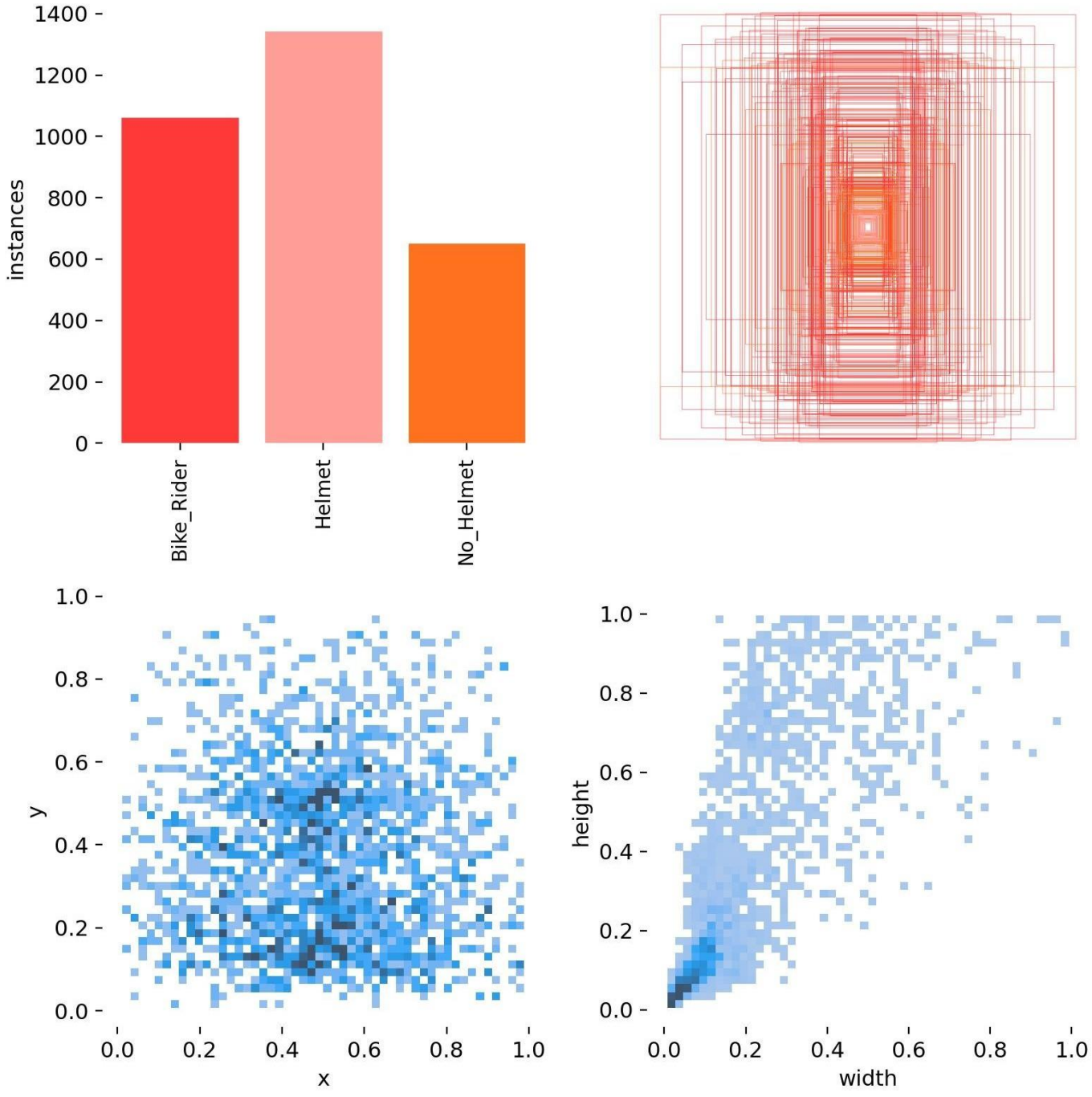


7.4) Precision-Recall Curve





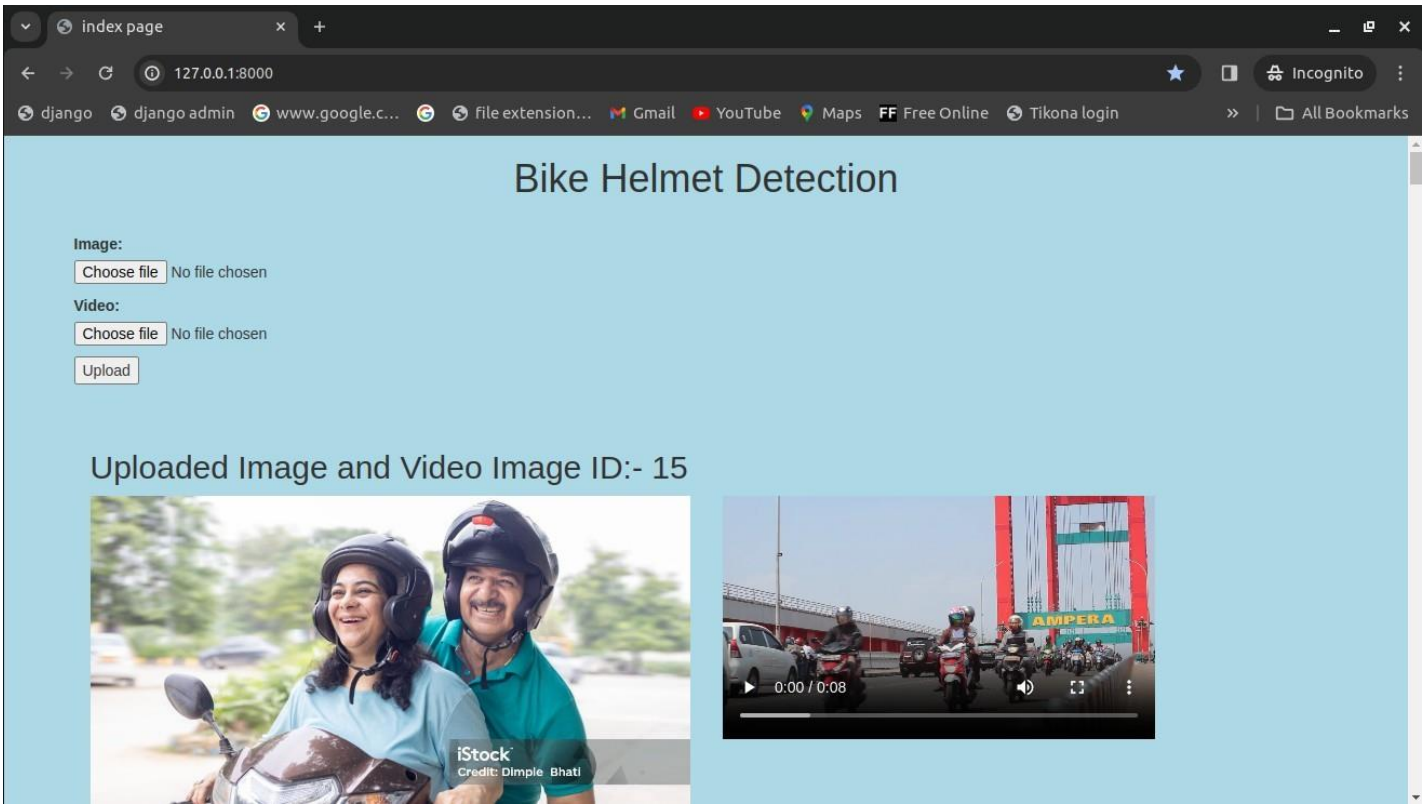
7.5) Labels-Class Chart

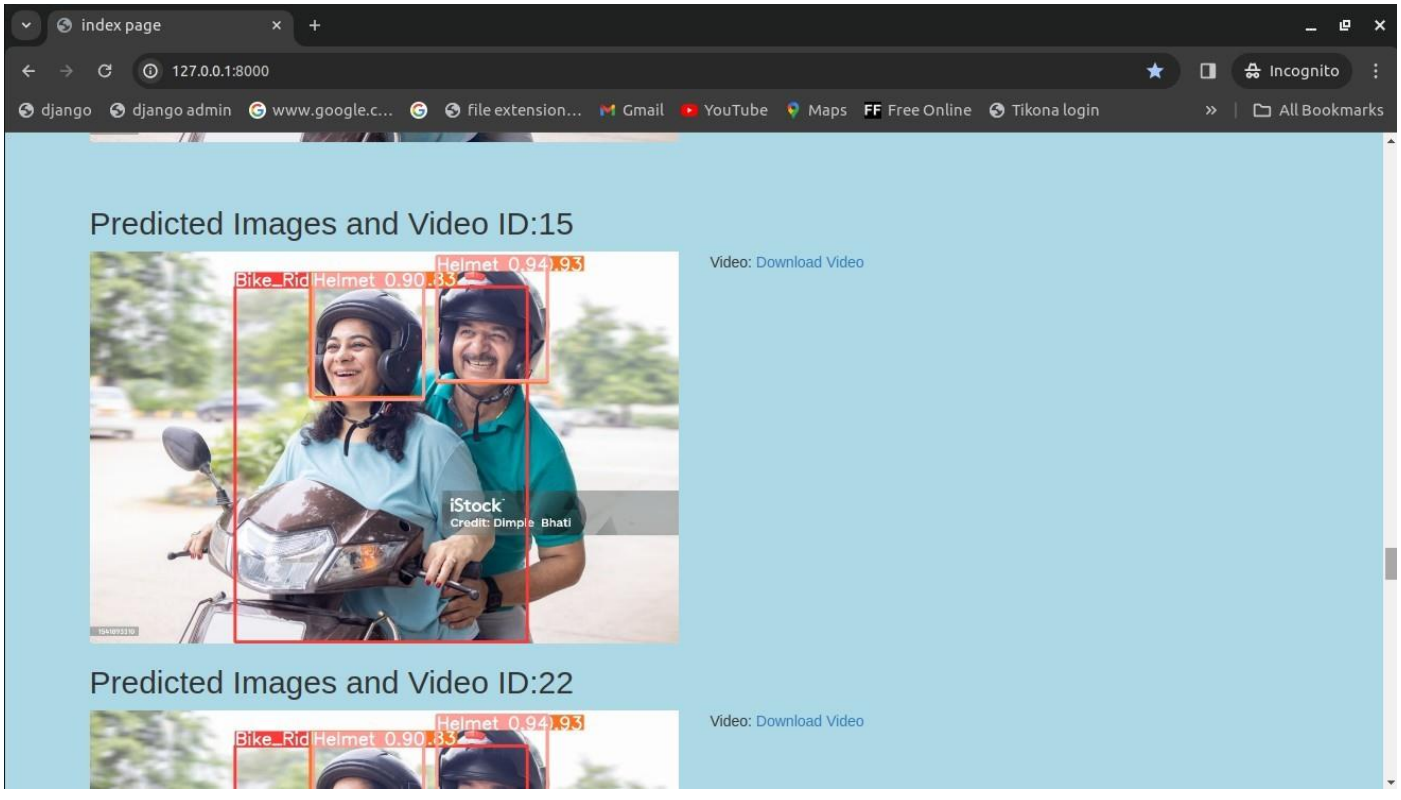


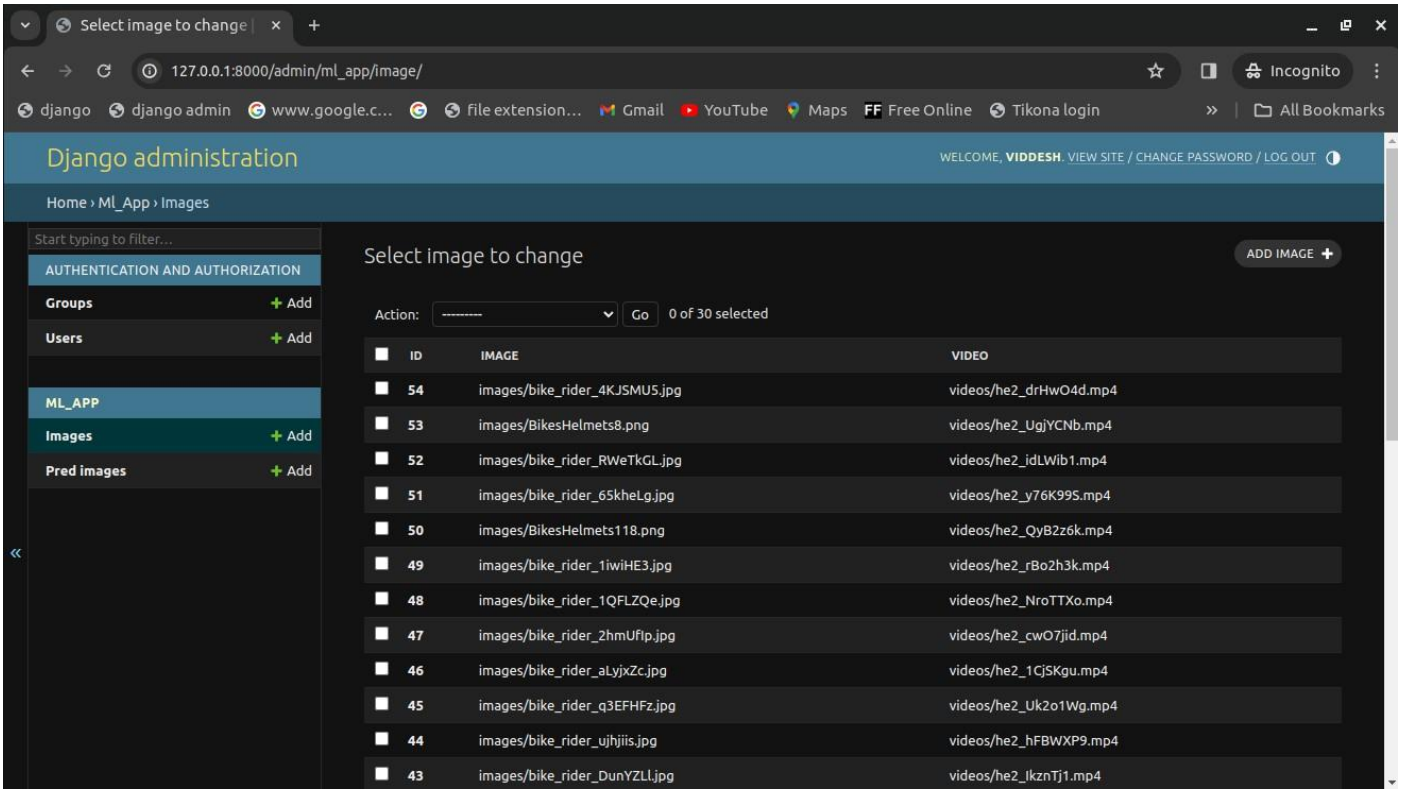
7.6) Inference on Images

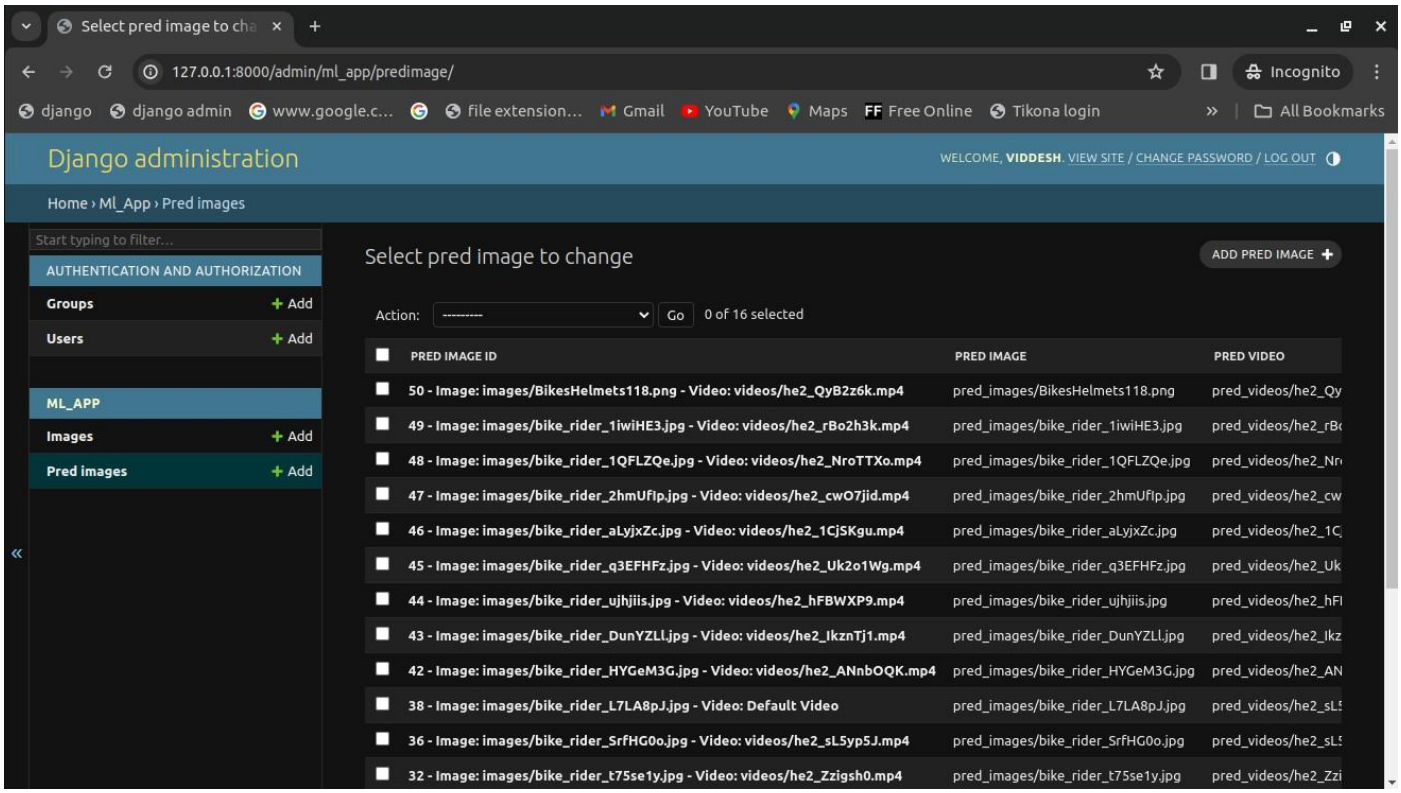
7.7) Inference on Videos

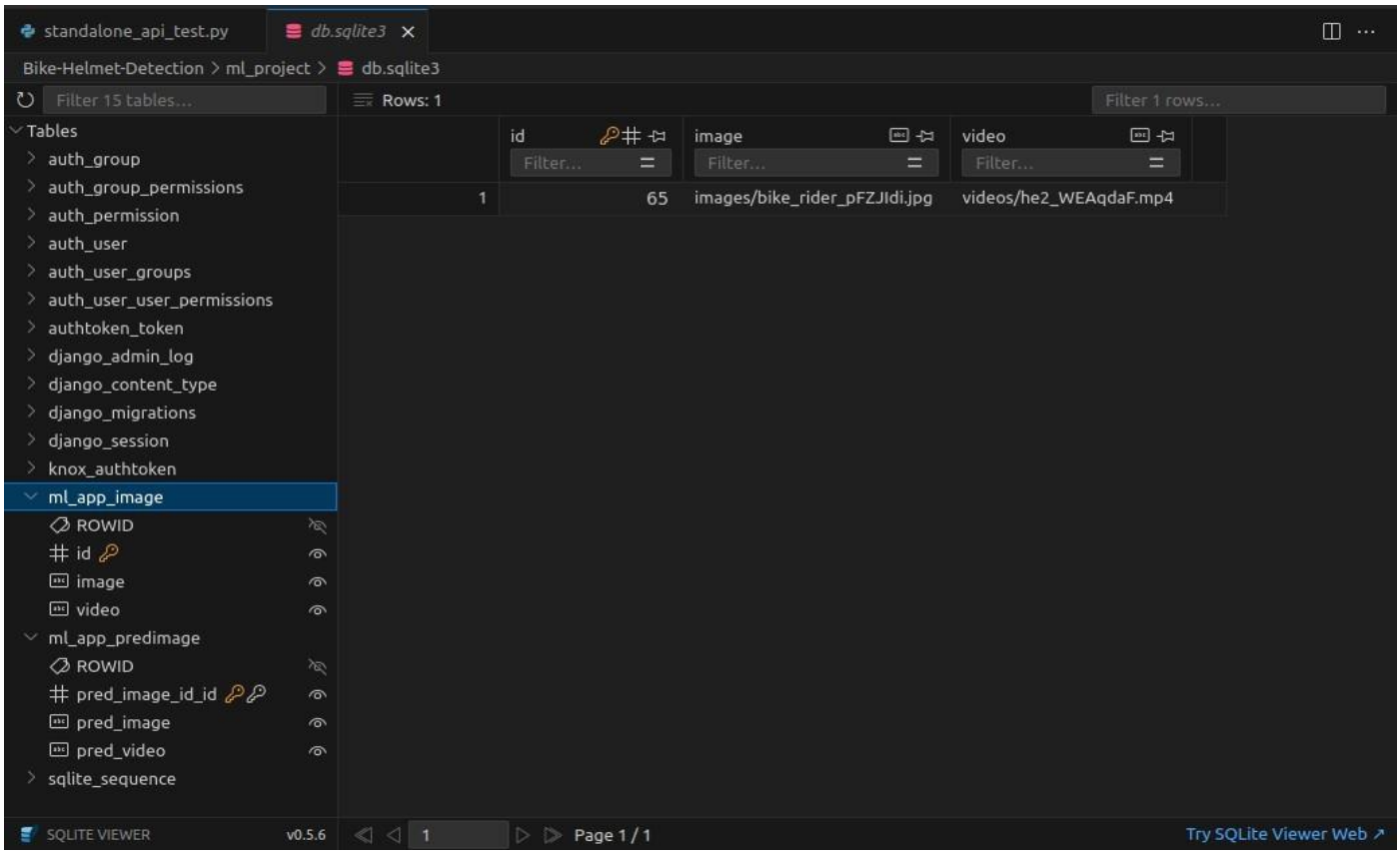
7.8) Django Web application Demo

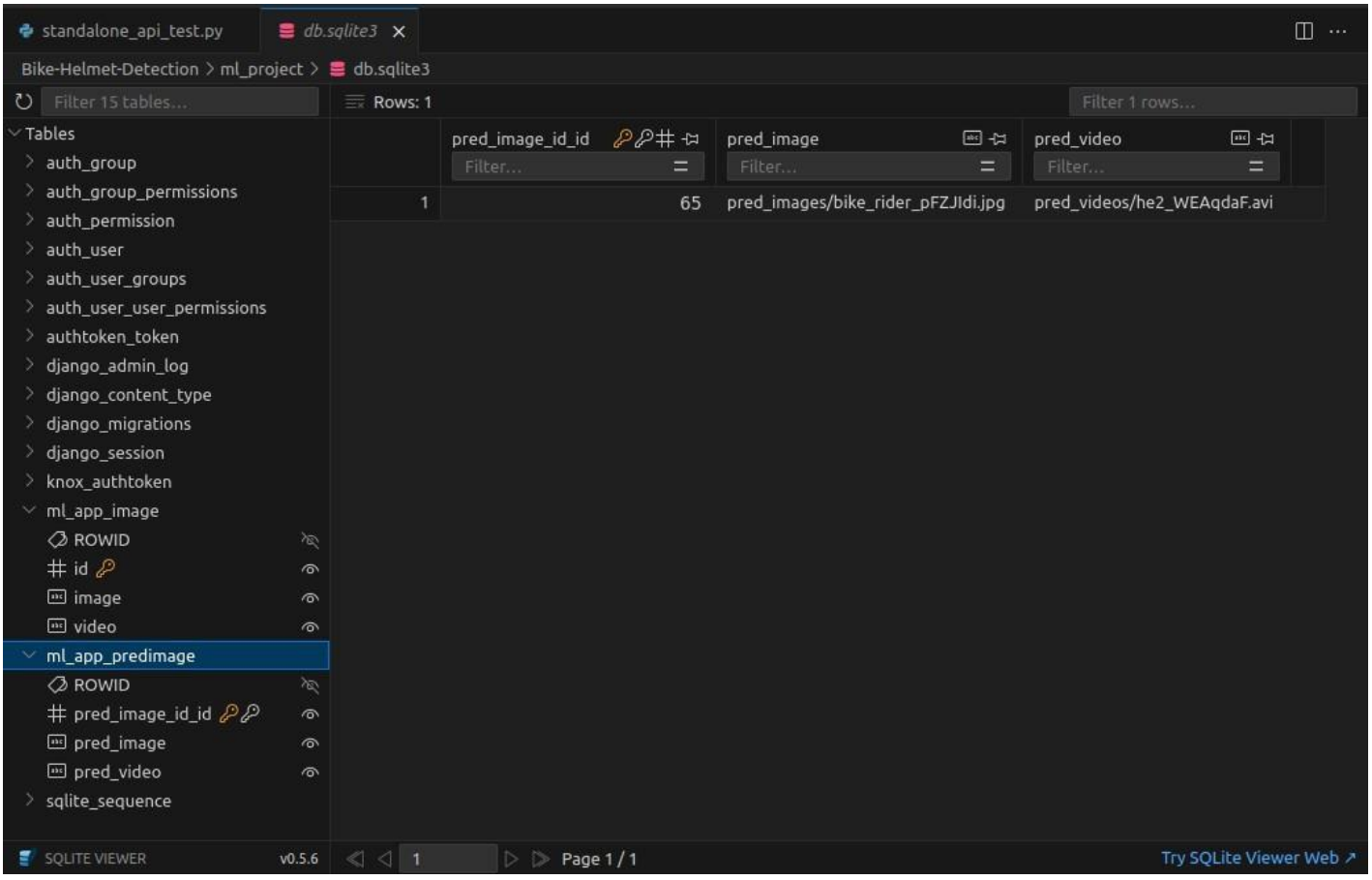





Select image to change
127.0.0.1:8000/admin/ml_app/image/
Incognito
django
django admin
www.google.c...
file extension...
Gmail
YouTube
Maps
Free Online
Tikona login
All Bookmarks
Django administration
WELCOME, VIDDESH. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Ml_App › Images
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
ML_APP
Images + Add
Pred images + Add
Select image to change
ADD IMAGE +
Action: --------- Go 0 of 30 selected
ID IMAGE VIDEO
54 images/bike_rider_4KJSMU5.jpg videos/he2_drHwO4d.mp4
53 images/BikesHelmets8.png videos/he2_UgjYCNb.mp4
52 images/bike_rider_RWeTkGL.jpg videos/he2_idLWib1.mp4
51 images/bike_rider_65kheLg.jpg videos/he2_y76K99S.mp4
50 images/BikesHelmets118.png videos/he2_QyB2z6k.mp4
49 images/bike_rider_1iwiHE3.jpg videos/he2_rBo2h3k.mp4
48 images/bike_rider_1QFLZQe.jpg videos/he2_NroTTXo.mp4
47 images/bike_rider_2hmUfIp.jpg videos/he2_cwO7jid.mp4
46 images/bike_rider_aLyjxZc.jpg videos/he2_1CjSKgu.mp4
45 images/bike_rider_q3EFHFz.jpg videos/he2_Uk2o1Wg.mp4
44 images/bike_rider_ujhjiis.jpg videos/he2_hFBWXP9.mp4
43 images/bike_rider_DunYZLl.jpg videos/he2_IkznTj1.mp4


Select pred image to cha
127.0.0.1:8000/admin/ml_app/predimage/
Incognito
django
django admin
www.google.c...
file extension...
Gmail
YouTube
Maps
Free Online
Tikona login
All Bookmarks
Django administration
WELCOME, VIDDESH. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Ml_App › Pred images
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
ML_APP
Images + Add
Pred images + Add
Select pred image to change
ADD PRED IMAGE +
Action: --------- Go 0 of 16 selected
PRED IMAGE ID PRED IMAGE PRED VIDEO
50 - Image: images/BikesHelmets118.png - Video: videos/he2_QyB2z6k.mp4 pred_images/BikesHelmets118.png pred_videos/he2_Qy
49 - Image: images/bike_rider_1iwiHE3.jpg - Video: videos/he2_rBo2h3k.mp4 pred_images/bike_rider_1iwiHE3.jpg pred_videos/he2_rB
48 - Image: images/bike_rider_1QFLZQe.jpg - Video: videos/he2_NroTTXo.mp4 pred_images/bike_rider_1QFLZQe.jpg pred_videos/he2_Nr
47 - Image: images/bike_rider_2hmUfIp.jpg - Video: videos/he2_cwO7jid.mp4 pred_images/bike_rider_2hmUfIp.jpg pred_videos/he2_cw
46 - Image: images/bike_rider_aLyjxZc.jpg - Video: videos/he2_1CjSKgu.mp4 pred_images/bike_rider_aLyjxZc.jpg pred_videos/he2_1C
45 - Image: images/bike_rider_q3EFHFz.jpg - Video: videos/he2_Uk2o1Wg.mp4 pred_images/bike_rider_q3EFHFz.jpg pred_videos/he2_Uk
44 - Image: images/bike_rider_ujhjiis.jpg - Video: videos/he2_hFBWXP9.mp4 pred_images/bike_rider_ujhjiis.jpg pred_videos/he2_hFI
43 - Image: images/bike_rider_DunYZLl.jpg - Video: videos/he2_IkznTj1.mp4 pred_images/bike_rider_DunYZLl.jpg pred_videos/he2_Ikz
42 - Image: images/bike_rider_HYGeM3G.jpg - Video: videos/he2_ANnbOQK.mp4 pred_images/bike_rider_HYGeM3G.jpg pred_videos/he2_AN
38 - Image: images/bike_rider_L7LA8pJ.jpg - Video: Default Video pred_images/bike_rider_L7LA8pJ.jpg pred_videos/he2_sL5
36 - Image: images/bike_rider_SrfHG0o.jpg - Video: videos/he2_sL5yp5J.mp4 pred_images/bike_rider_SrfHG0o.jpg pred_videos/he2_sL5
32 - Image: images/bike_rider_t75se1y.jpg - Video: videos/he2_Zzigsh0.mp4 pred_images/bike_rider_t75se1y.jpg pred_videos/he2_Zzi


standalone_api_test.py
db.sqlite3
Bike-Helmet-Detection > ml_project > db.sqlite3
Filter 15 tables...
Rows: 1
Filter 1 rows...
Tables
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
authtoken_token
django_admin_log
django_content_type
django_migrations
django_session
knox_authtoken
ml_app_image
ROWID
id
image
video
ml_app_predimage
ROWID
pred_image_id_id
pred_image
pred_video
sqlite_sequence
id
image
video
1
65
images/bike_rider_pFZJIdi.jpg
videos/he2_WEAqdaF.mp4
SQLITE VIEWER
v0.5.6
Page 1 / 1
Try SQLite Viewer Web


standalone_api_test.py
db.sqlite3
Bike-Helmet-Detection > ml_project > db.sqlite3
Filter 15 tables...
Rows: 1
Filter 1 rows...
Tables
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
authtoken_token
django_admin_log
django_content_type
django_migrations
django_session
knox_authtoken
ml_app_image
ROWID
id
image
video
ml_app_predimage
ROWID
pred_image_id_id
pred_image
pred_video
sqlite_sequence
pred_image_id_id
pred_image
pred_video
1
65
pred_images/bike_rider_pFZJIdi.jpg
pred_videos/he2_WEAqdaF.avi
SQLITE VIEWER
v0.5.6
Page 1 / 1
Try SQLite Viewer Web

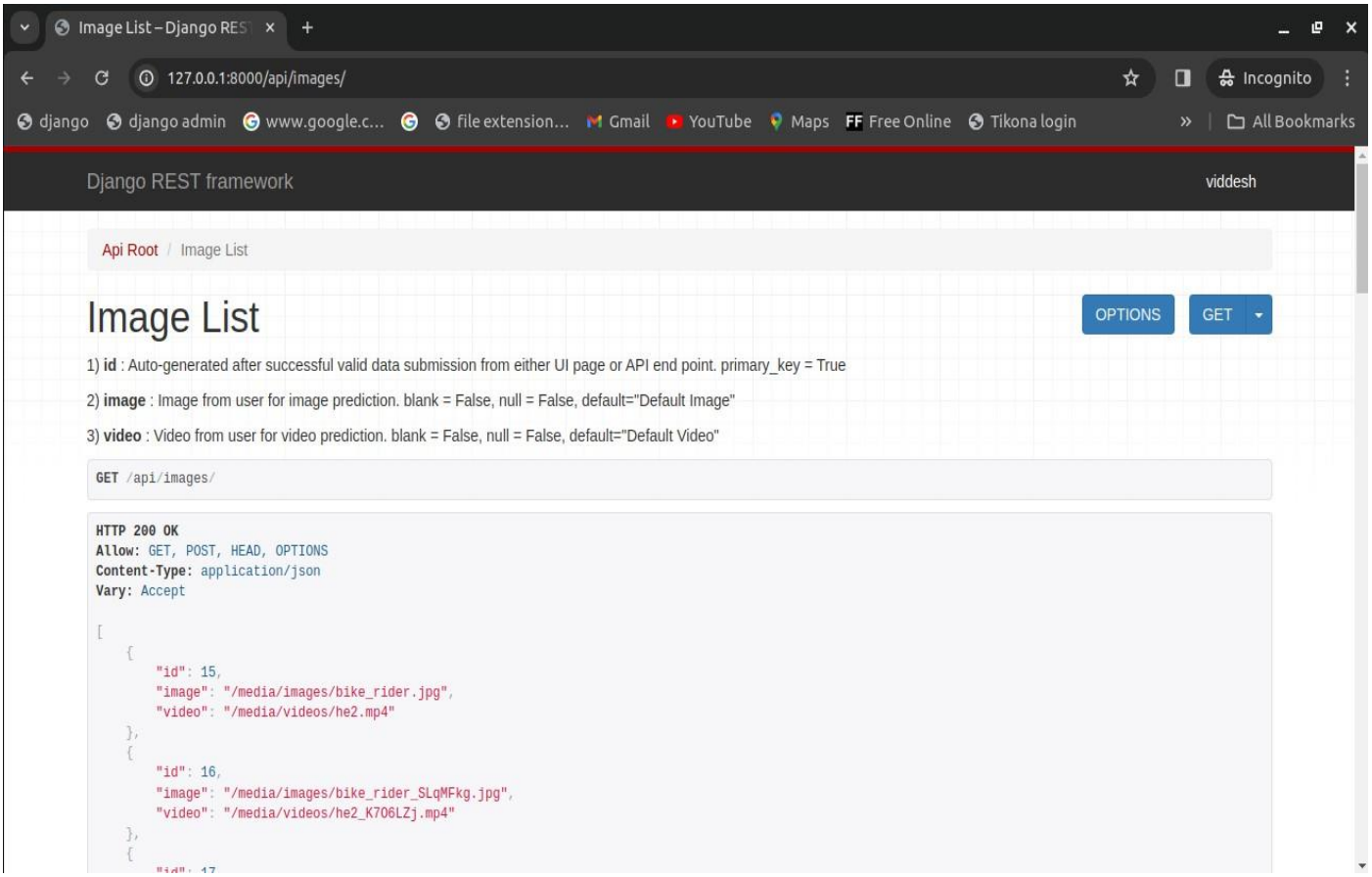
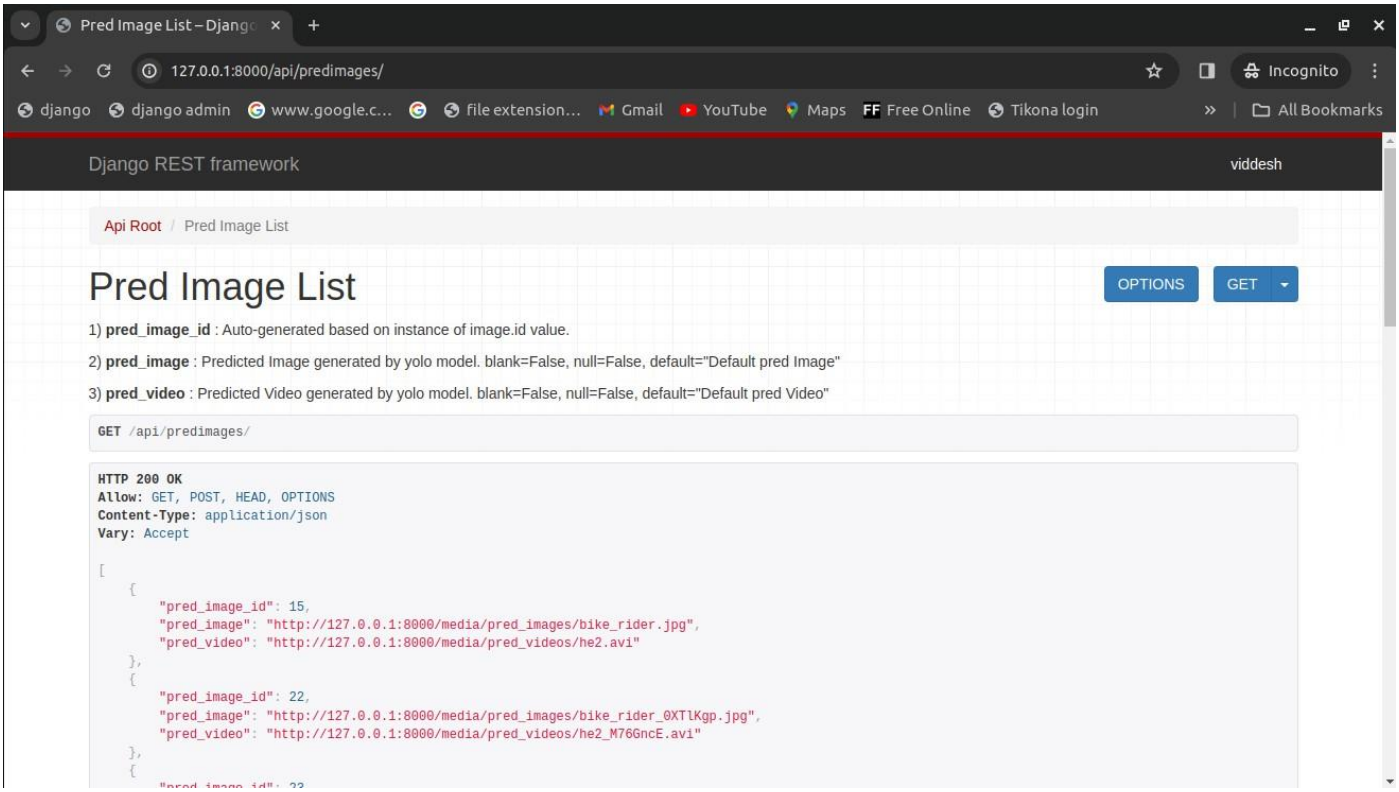
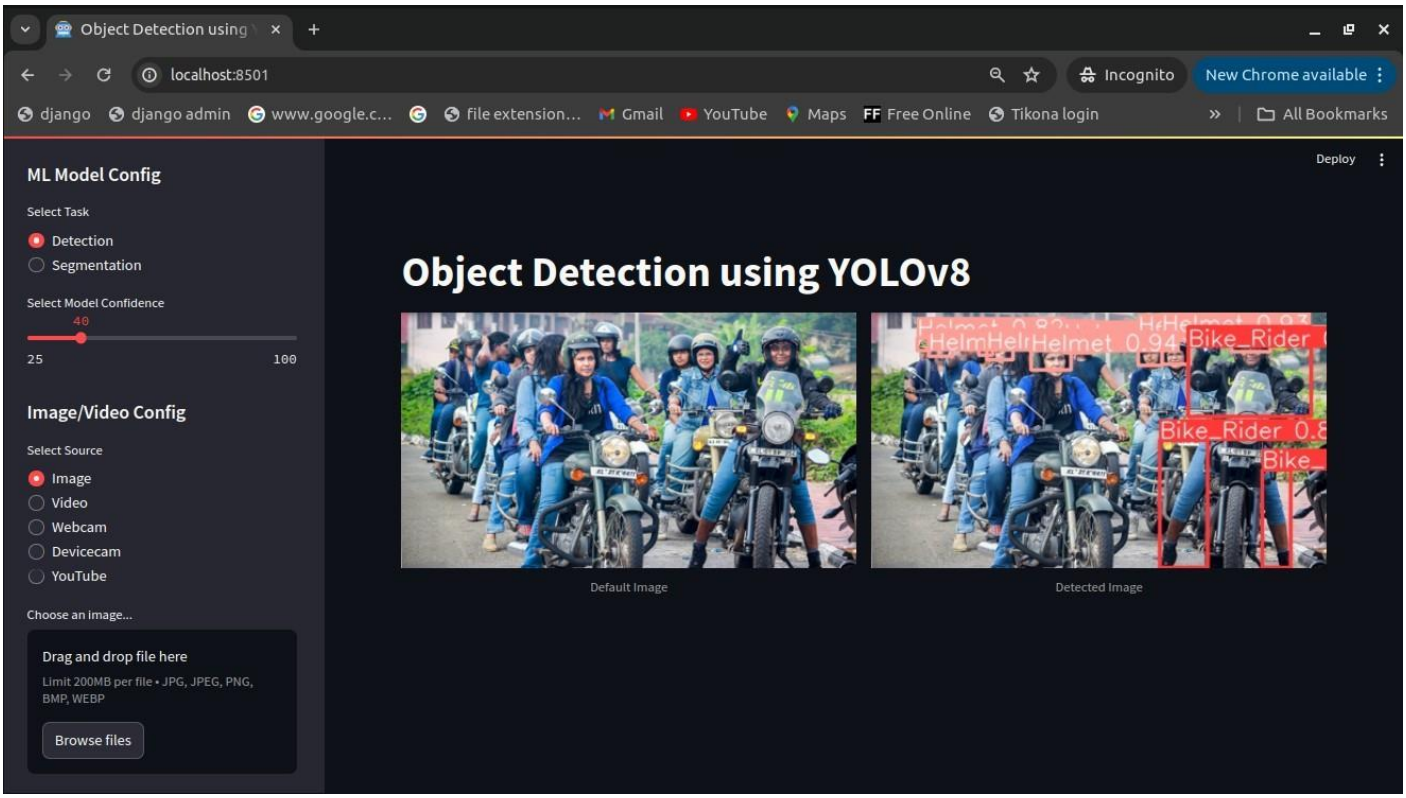
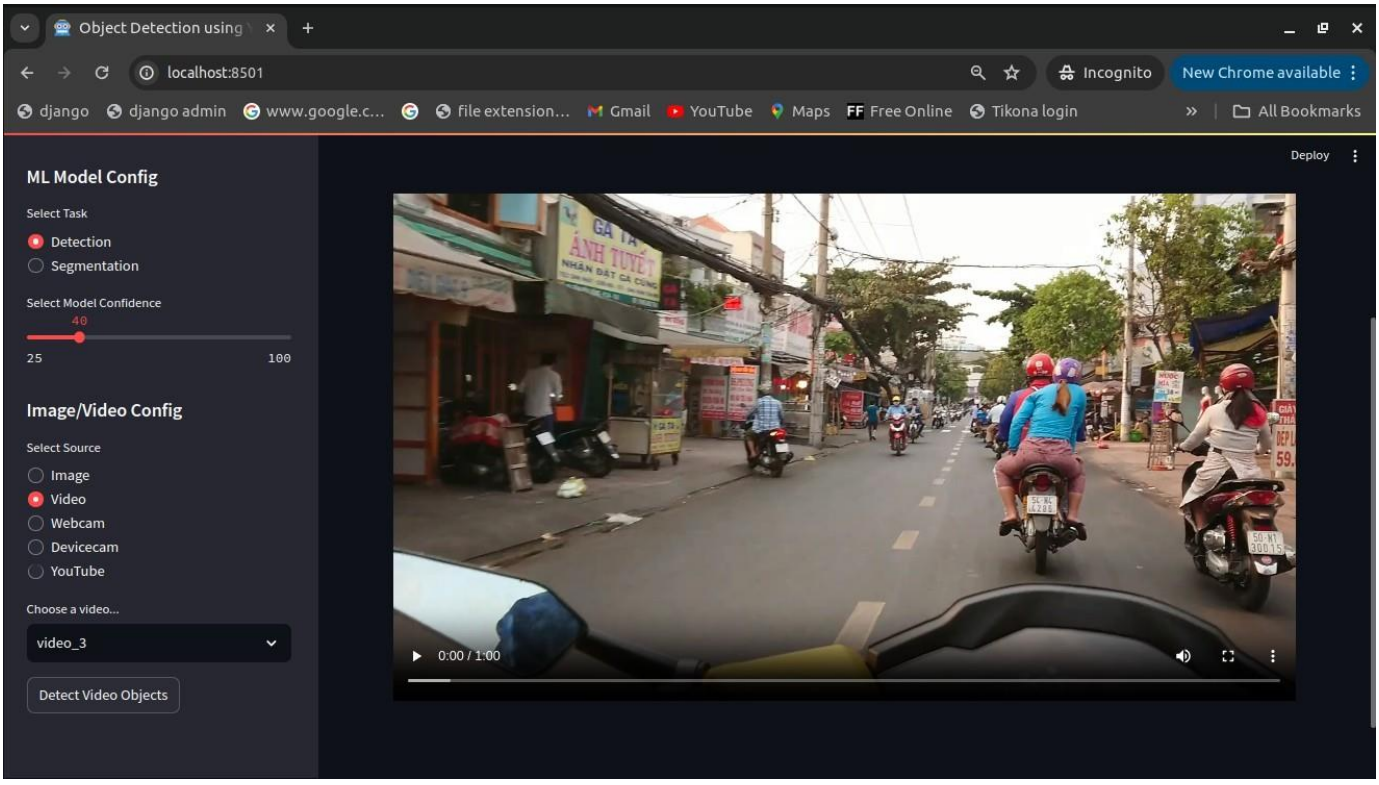


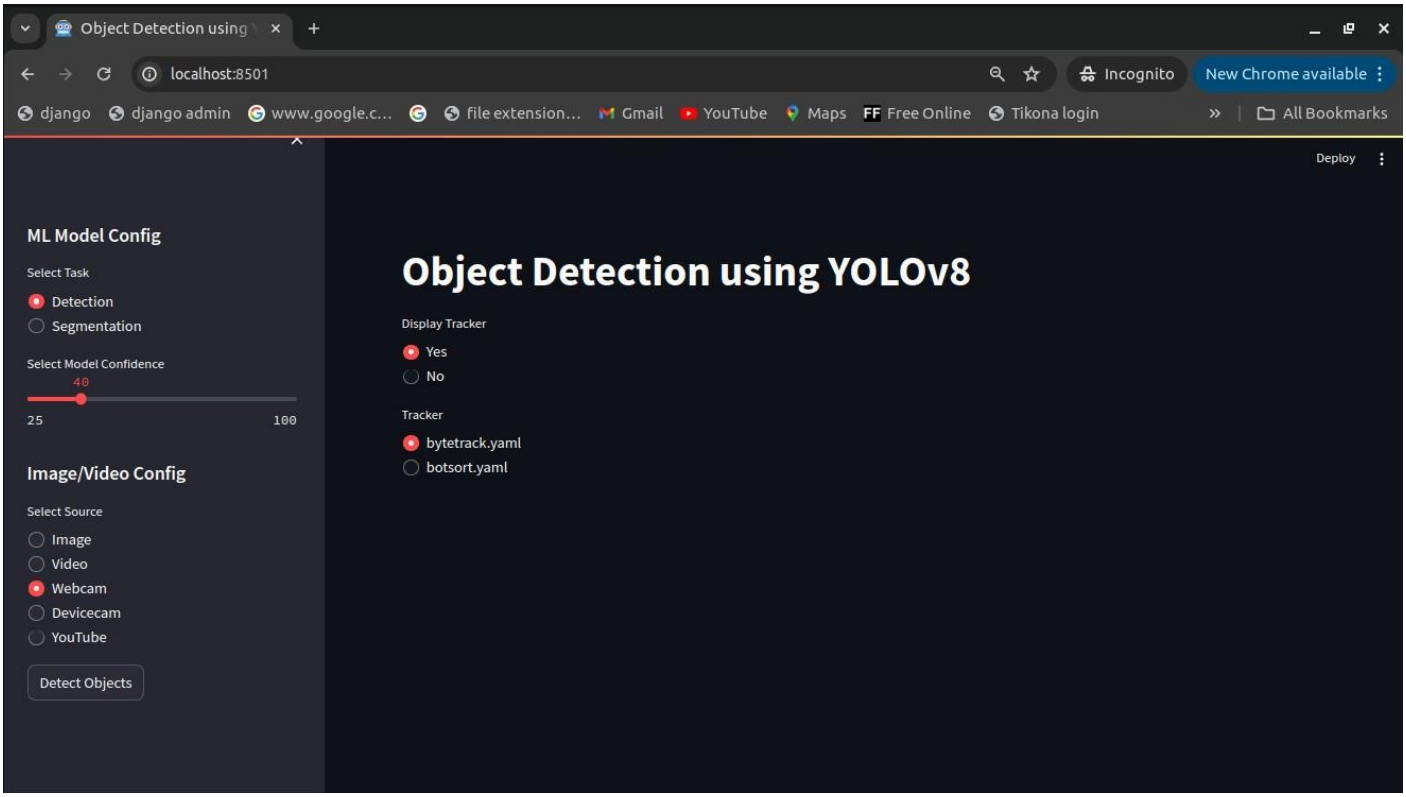
Image List – Django RES
127.0.0.1:8000/api/images/
Incognito
django
django admin
www.google.c...
file extension...
Gmail
YouTube
Maps
Free Online
Tikona login
All Bookmarks
Django REST framework
viddesh
Api Root / Image List
Image List
OPTIONS
GET
1) id : Auto-generated after successful valid data submission from either UI page or API end point. primary_key = True
2) image : Image from user for image prediction. blank = False, null = False, default="Default Image"
3) video : Video from user for video prediction. blank = False, null = False, default="Default Video"
GET /api/images/
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 15,
"image": "/media/images/bike_rider.jpg",
"video": "/media/videos/he2.mp4"
},
{
"id": 16,
"image": "/media/images/bike_rider_SLqMFkg.jpg",
"video": "/media/videos/he2_K7O6LZj.mp4"
},

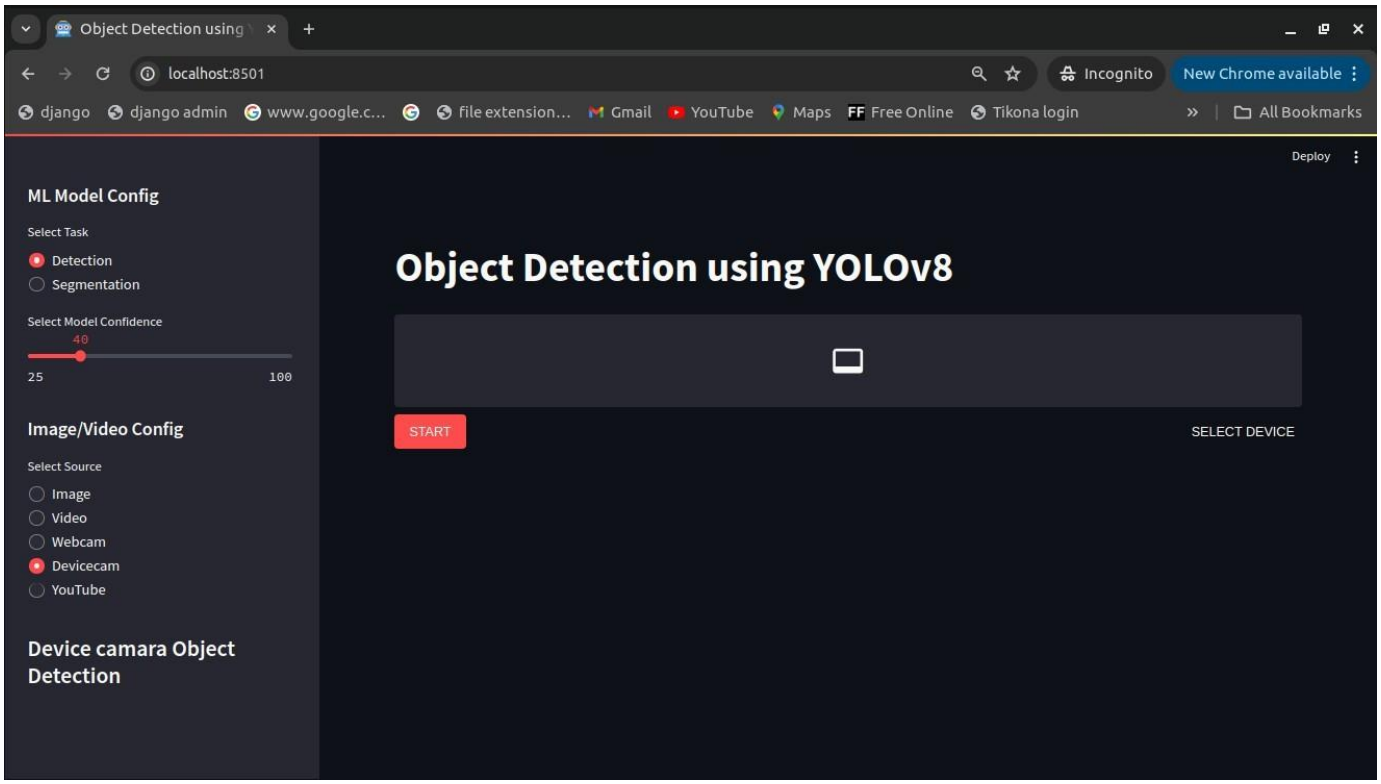

Pred Image List – Djang
127.0.0.1:8000/api/predimages/
Incognito
django
django admin
www.google.c...
file extension...
Gmail
YouTube
Maps
Free Online
Tikona login
All Bookmarks
Django REST framework
viddesh
Api Root / Pred Image List
Pred Image List
OPTIONS
GET
1) pred_image_id : Auto-generated based on instance of image.id value.
2) pred_image : Predicted Image generated by yolo model. blank=False, null=False, default="Default pred Image"
3) pred_video : Predicted Video generated by yolo model. blank=False, null=False, default="Default pred Video"
GET /api/predimages/
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"pred_image_id": 15,
"pred_image": "http://127.0.0.1:8000/media/pred_images/bike_rider.jpg",
"pred_video": "http://127.0.0.1:8000/media/pred_videos/he2.avi"
},
{
"pred_image_id": 22,
"pred_image": "http://127.0.0.1:8000/media/pred_images/bike_rider_0XTlKgp.jpg",
"pred_video": "http://127.0.0.1:8000/media/pred_videos/he2_M76GncE.avi"
},

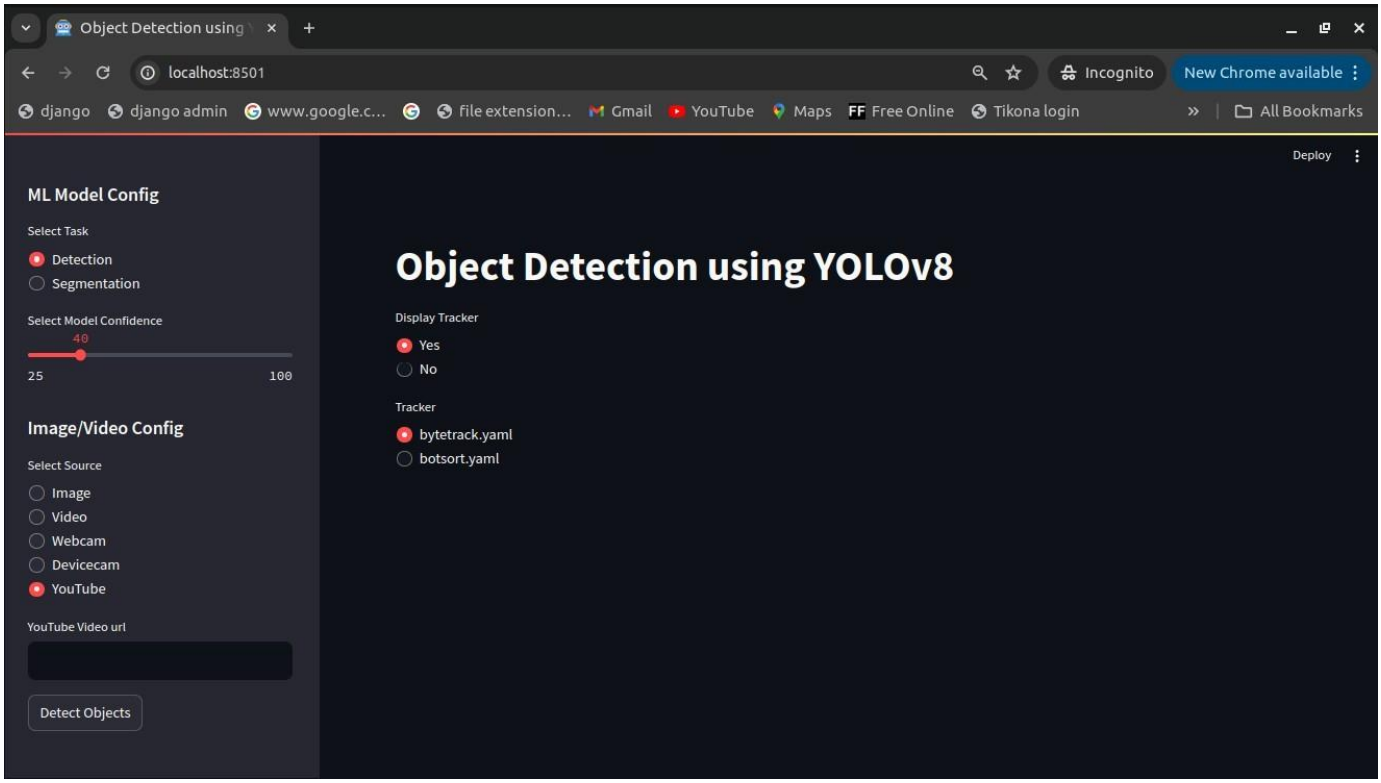
7.9) Streamlit Web application Demo



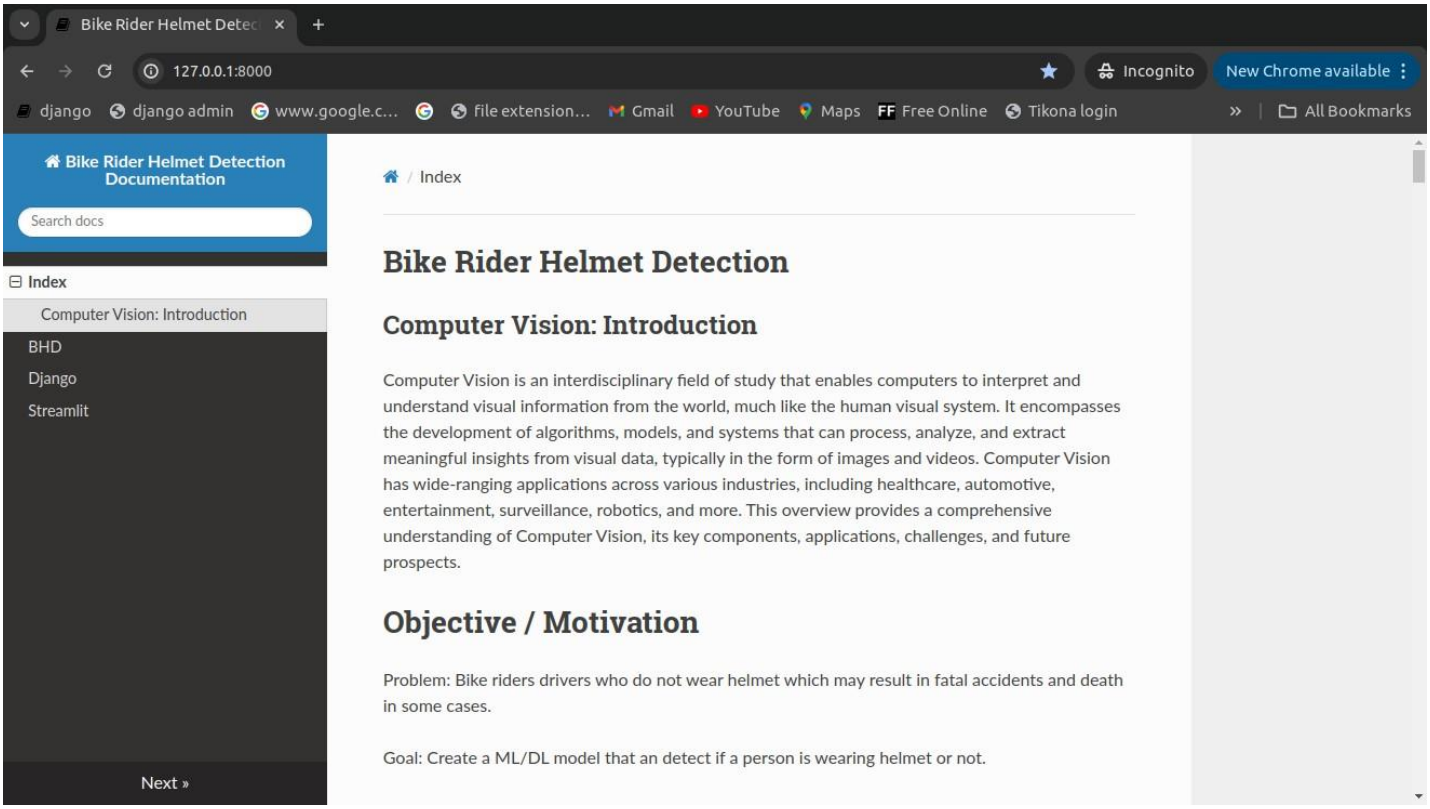


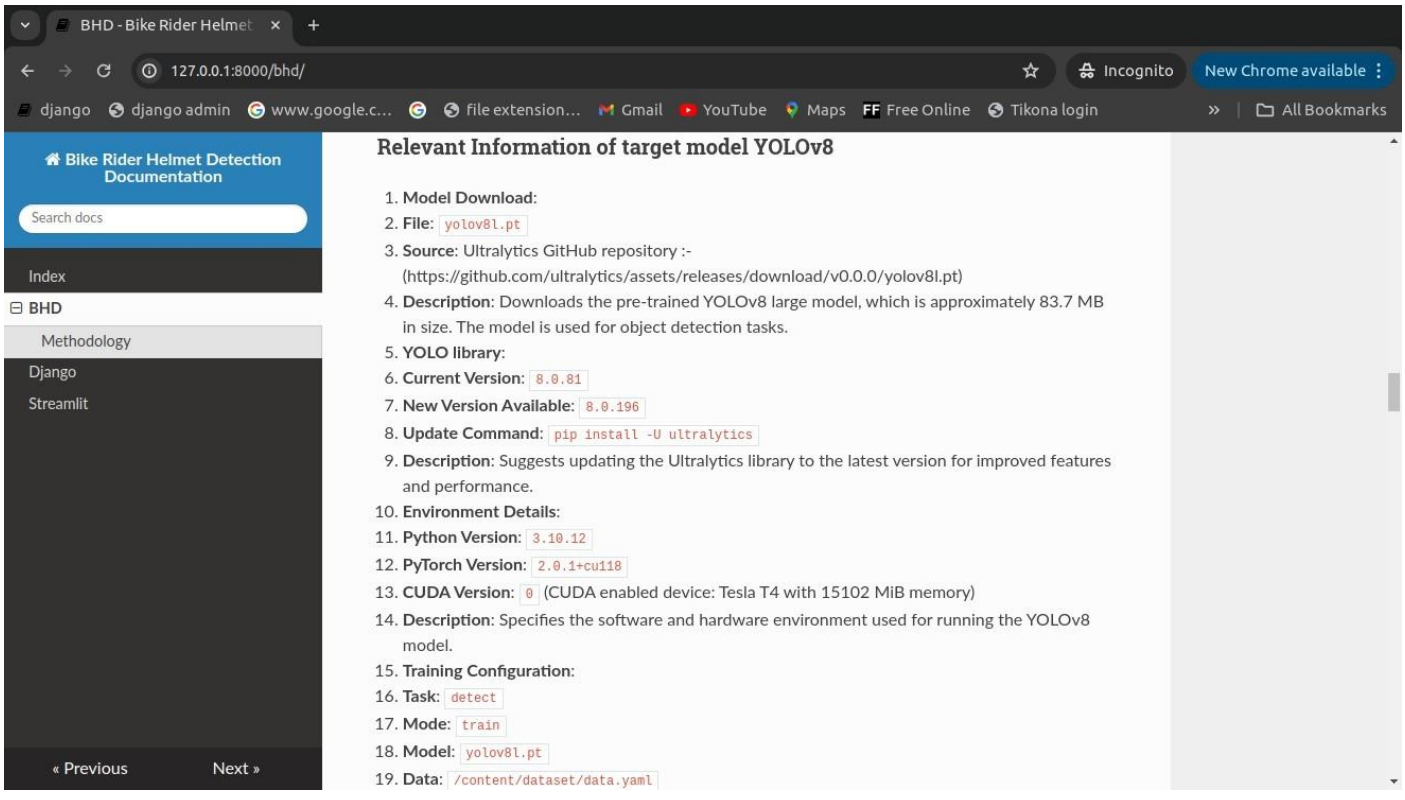

Object Detection using
localhost:8501
Incognito
New Chrome available
django
django admin
www.google.c...
file extension...
Gmail
YouTube
Maps
Free Online
Tikona login
All Bookmarks
Deploy
ML Model Config
Select Task
Detection
Segmentation
Select Model Confidence
40
25
100
Image/Video Config
Select Source
Image
Video
Webcam
Devicecam
YouTube
Detect Objects
Object Detection using YOLOv8
Display Tracker
Yes
No
Tracker
bytetrack.yaml
botsort.yaml


Object Detection using
localhost:8501
Incognito
New Chrome available
django
django admin
www.google.c...
file extension...
Gmail
YouTube
Maps
Free Online
Tikona login
All Bookmarks
Deploy
ML Model Config
Select Task
Detection
Segmentation
Select Model Confidence
40
25
100
Image/Video Config
Select Source
Image
Video
Webcam
Devicecam
YouTube
Device camara Object Detection
Object Detection using YOLOv8
START
SELECT DEVICE


Object Detection using
localhost:8501
Incognito
New Chrome available
django
django admin
www.google.c...
file extension...
Gmail
YouTube
Maps
Free Online
Tikona login
All Bookmarks
Deploy
ML Model Config
Select Task
Detection
Segmentation
Select Model Confidence
40
25
100
Image/Video Config
Select Source
Image
Video
Webcam
Devicecam
YouTube
YouTube Video url
Detect Objects
Object Detection using YOLOv8
Display Tracker
Yes
No
Tracker
bytetrack.yaml
botsort.yaml

7.10) Mkdocs Documentation Demo





Chapter 8

8. Conclusion

In conclusion, the development of a computer vision model using YOLOv8 for bike rider helmet detection represents a significant advancement in enhancing safety measures for riders along with passengers. The model demonstrates impressive accuracy and efficiency in identifying helmet-wearing individuals, which is crucial for ensuring compliance with safety regulations and reducing the risk of head injuries in bike-related accidents.

This project highlights the potential of computer vision technology to address real-world safety challenges effectively.

Chapter 9

9. Future Scope

The Future scope of this project should be focused on latest implementation of **YOLOv9** and recent **YOLOv10**. There should be seperate repository for implementation of YOLOv9 and YOLOv10 along with any demo related to it with any technology for creating web application like django, flask, mesop, streamlit, gradio etc. or any desktop application using various techology as reflex, nicegui, tkinter, kivy etc.

Chapter 10

10. Program code

1) Implementation of Bike Helmet Detection Jupyter Notebook

https://github.com/Viddesh1/Helmet_test_1

2) Output generated by YOLOv8

https://drive.google.com/drive/folders/1M4FckJJeyPQTTWqgo6xWhW8L4tf0EJ4l?usp=sharing

3) Bike Helmet Detection Django Web application

https://github.com/Viddesh1/Bike-Helmet-Detection

4) Bike Helmet Detection Stremlit Web application

https://github.com/Viddesh1/Bike-Helmet-Detectionv2

5) Hosted on Streamlit

https://bike-helmet-detectionv2-dmehozp3lkef4wnssaepjf.streamlit.app/

6) Overall Documentation

https://github.com/Viddesh1/Bike-Helmet-Detection-Docs

Chapter 11

11. References

1) https://github.com/ultralytics/ultralytics

2) https://towardsdatascience.com/dino-emerging-properties-in-self-supervised-vision-transformers-summary-ab91df82cc3c

3) https://github.com/facebookresearch/dino/

4) Emerging Properties in Self-Supervised Vision Transformers : https://arxiv.org/abs/2104.14294/

5) https://dinov2.metademolab.com/

6) https://segment-anything.com/

7) https://blog.roboflow.com/whats-new-in-yolov8/