```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!pip install ultralytics
!pip install opencv-python
!pip install matplotlib
```

```
Collecting ultralytics
  Downloading ultralytics-8.2.95-py3-none-any.whl.metadata (39 kB)
Requirement already satisfied: numpy<2.0.0,>=1.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.26.4)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (3.7.1)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.10.0.84)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.4.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.2)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.32.3)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.13.1)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.4.0+cu121)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.19.0+cu121)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.66.5)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.1.4)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.13.1)
Collecting ultralytics-thop>=2.0.0 (from ultralytics)
  Downloading ultralytics_thop-2.0.6-py3-none-any.whl.metadata (9.1 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (24.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2024.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2024.8.3
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.16.0)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (4.12.2
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (1.13.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (2024.6.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.3.0->ultralyti
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.8.0->ultralytics) (2.1.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.8.0->ultralytics) (1.3.0
Downloading ultralytics-8.2.95-py3-none-any.whl (872 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 872.8/872.8 kB 29.7 MB/s eta 0:00:00
Downloading ultralytics_thop-2.0.6-py3-none-any.whl (26 kB)
Installing collected packages: ultralytics-thop, ultralytics
Successfully installed ultralytics-8.2.95 ultralytics-thop-2.0.6
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.26.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
```

```
!pip install xmltodict
```

```
Collecting xmltodict
  Downloading xmltodict-0.13.0-py2.py3-none-any.whl.metadata (7.7 kB)
Downloading xmltodict-0.13.0-py2.py3-none-any.whl (10.0 kB)
Installing collected packages: xmltodict
Successfully installed xmltodict-0.13.0
```

Below we are converting the annotations from xml files to yolo format labels, which will be stored in the 'labels' folder

```
import xmltodict
import os

def convert_annotations(xml_dir, output_dir):
    # Ensuring output directory exists
    os.makedirs(output_dir, exist_ok=True)

    # Listing all XML files in the directory
    xml_files = [f for f in os.listdir(xml_dir) if f.endswith('.xml')]

    for xml_file in xml_files:
```

```
xml_path = os.path.join(xml_dir, xml_file)

with open(xml_path) as f:
    xml_dict = xmltodict.parse(f.read())

# Checking if the XML has the expected structure
if 'annotation' not in xml_dict:
    print(f"Skipping {xml_file}, not a valid annotation.")
    continue

# Extracting image details
annotation = xml_dict['annotation']
filename = annotation.get('filename', 'unknown.jpg')
width = int(annotation['size']['width'])
height = int(annotation['size']['height'])

# Preparing YOLO format label file
output_label_file = os.path.join(output_dir, f'{filename[:-4]}.txt')
with open(output_label_file, 'w') as label_file:
    objects = annotation.get('object', [])
    if not isinstance(objects, list):
        objects = [objects]  # Converting to list if there's only one object

    for obj in objects:
        # Extracting bounding box coordinates
        bbox = obj['bndbox']
        x_min = int(bbox['xmin'])
        y_min = int(bbox['ymin'])
        x_max = int(bbox['xmax'])
        y_max = int(bbox['ymax'])

        # Converting to YOLO format
        x_center = (x_min + x_max) / 2 / width
        y_center = (y_min + y_max) / 2 / height
        bbox_width = (x_max - x_min) / width
        bbox_height = (y_max - y_min) / height

        # YOLO class ID
        category_name = obj['name']
        category_id = category_name_to_id.get(category_name, -1)  # Replacing with our mapping

        if category_id != -1:
            label_file.write(f"{category_id} {x_center} {y_center} {bbox_width} {bbox_height}\n")

# Defining the category name to ID mapping
category_name_to_id = {
    "D00": 0,
    "D10": 1,
    "D20": 2,
    "D40": 3,
    "D01": 4,
    "D11": 5,
    "D43": 6,
    "D44": 7
}

# Providing the path to the folder containing XML files and the output directory for YOLO format labels
convert_annotations('/content/drive/MyDrive/India2_modified/train/annotations', '/content/drive/MyDrive/India2_modified/train/labels')
```

Next, we are creating a yaml file which contains all the paths,number of classes along with the class names

```
%%writefile /content/rddetector.yaml
path: /content/drive/MyDrive/India2_modified
train: train/images
val: train/images
test: test/images

nc: 8  # Number of classes
names:
  - D00  # Longitudinal Crack
  - D10  # Transverse Crack
  - D20  # Alligator Crack
  - D40  # Potholes
  - D01  # Small Longitudinal crack
  - D11  # Small Transverse crack
  - D43  # Repaired road section
  - D44  # Other road damages
```

```
Writing /content/rddetector.yaml
```

Training: Below we will use a pretrained model and fine-tune it on a smaller dataset we selected, in order to allow the model to learn from specific data. (Transfer learning)

```
from ultralytics import YOLO

# Loading the YOLOv8 model and training
model = YOLO('/content/drive/MyDrive/YOLOv8s_rdd.pt')  # model
model.train(data='/content/rddetector.yaml', epochs=1, imgsz=640, batch=8,pretrained=True)
```

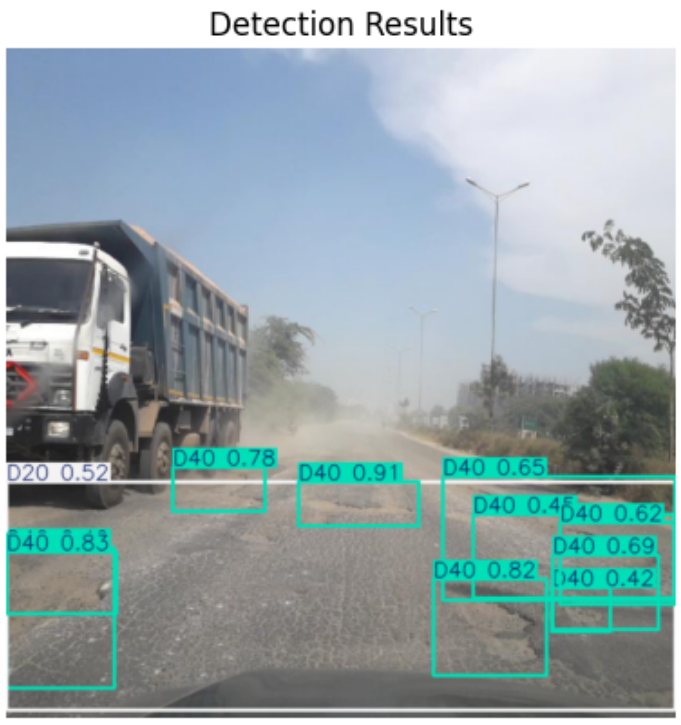
```
Ultralytics YOLOv8.2.95 🚀 Python-3.10.12 torch-2.4.0+cu121 CPU (Intel Xeon 2.20GHz)
engine/trainer: task=detect, mode=train, model=/content/drive/MyDrive/YOLOv8s_rdd.pt, data=/content/rddetector.yaml, epochs=1, time=None, p
Overriding model.yaml nc=4 with nc=8

                   from  n    params  module                                       arguments
  0                  -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                  -1  1     29056  ultralytics.nn.modules.block.C2f             [64, 64, 1, True]
  3                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  4                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  5                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  6                  -1  2    788480  ultralytics.nn.modules.block.C2f             [256, 256, 2, True]
  7                  -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                  -1  1   1838080  ultralytics.nn.modules.block.C2f             [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    591360  ultralytics.nn.modules.block.C2f             [768, 256, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 16                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 19                  -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1   1969152  ultralytics.nn.modules.block.C2f             [768, 512, 1]
 22        [15, 18, 21]  1   2119144  ultralytics.nn.modules.head.Detect           [8, [128, 256, 512]]
Model summary: 225 layers, 11,138,696 parameters, 11,138,680 gradients, 28.7 GFLOPs

Transferred 349/355 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/train3', view at http://localhost:6006/
Freezing layer 'model.22.dfl.conv.weight'
train: Scanning /content/drive/MyDrive/India2_modified/train/labels.cache... 1058 images, 529 backgrounds, 0 corrupt: 100%|██████████| 1058

val: Scanning /content/drive/MyDrive/India2_modified/train/labels.cache... 1058 images, 529 backgrounds, 0 corrupt: 100%|██████████| 1058/1

optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automati
optimizer: AdamW(lr=0.000833, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.0)
TensorBoard: model graph visualization added ✅
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs/detect/train3
Starting training for 1 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        1/1         0G      1.913      19.85       1.64          2        640: 100%|██████████| 133/133 [43:20<00:00, 19.55s/it]
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 67/67 [14:52<00:00, 13.31s/it]
                    all       1058       1141      0.775      0.146       0.15     0.0745

1 epochs completed in 0.975 hours.
Optimizer stripped from runs/detect/train3/weights/last.pt, 22.5MB
Optimizer stripped from runs/detect/train3/weights/best.pt, 22.5MB

Validating runs/detect/train3/weights/best.pt...
Ultralytics YOLOv8.2.95 🚀 Python-3.10.12 torch-2.4.0+cu121 CPU (Intel Xeon 2.20GHz)
Model summary (fused): 168 layers, 11,128,680 parameters, 0 gradients, 28.5 GFLOPs
```

Testing: 1) first we will test the trained model on an image from the training dataset itself

```
from ultralytics import YOLO
import numpy as np
import cv2
import matplotlib.pyplot as plt

# Loading the trained YOLOv8 model
model = YOLO('/content/runs/detect/train3/weights/best.pt')

# List of image paths for testing
IMAGE_PATHS = [
    '/content/drive/MyDrive/India2_modified/train/images/India_000209.jpg'
]

def load_image_into_numpy_array(path):
    """Loading image from file path into a numpy array."""
    return cv2.imread(path)

def resize_image(image, size=(640, 640)):
    """Resizing image to the specified size."""
    return cv2.resize(image, size)

# Looping through each image and running inference
for image_path in IMAGE_PATHS:
```

```
print(f'Running inference for {image_path}... ', end='')

# Loading image as numpy array
image_np = load_image_into_numpy_array(image_path)

# Resizing the image to 640x640
image_resized = resize_image(image_np)

# Performing inference using YOLOv8 model
results = model.predict(source=image_resized, conf=0.4)  # Adjusting confidence threshold as needed

# Extracting results for the first image (YOLOv8 outputs in list format)
result = results[0]

# Visualizing the detections
# YOLOv8 already provides bounding boxes, class names, and confidence scores
annotated_img = result.plot()  # Getting annotated image with bounding boxes

# Showing the image with detections
plt.imshow(cv2.cvtColor(annotated_img, cv2.COLOR_BGR2RGB))
plt.title("Detection Results")
plt.axis('off')
plt.show()

print('Done')
```

```
Running inference for /content/drive/MyDrive/India2_modified/train/images/India_000209.jpg...
0: 640x640 1 D20, 10 D40s, 623.0ms
Speed: 5.7ms preprocess, 623.0ms inference, 1.2ms postprocess per image at shape (1, 3, 640, 640)
```



Saving the trained model to the drive with the below command

```
!cp /content/runs/detect/train3/weights/best.pt /content/drive/MyDrive/rdd1_yolov8_.pt
```

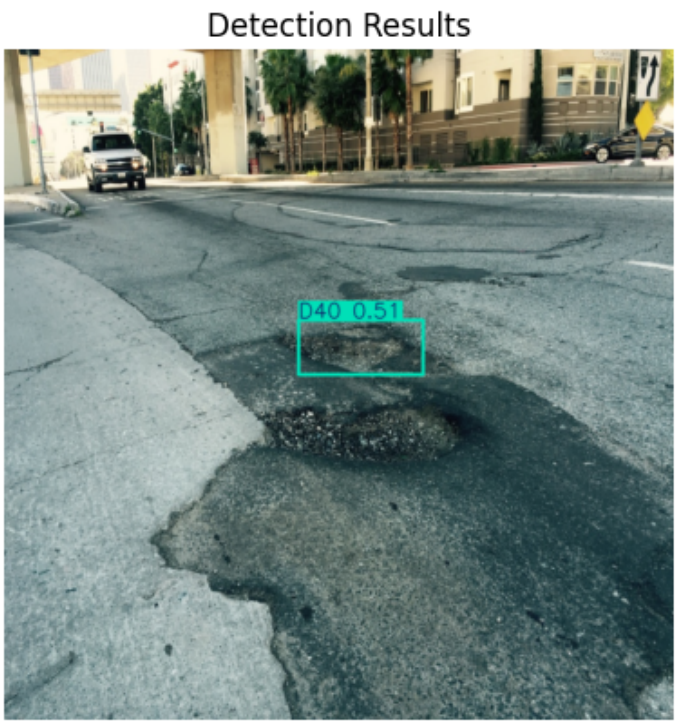
Testing: 2) Now testing the model with an image from the test dataset and another random image

```
from ultralytics import YOLO
import numpy as np
import cv2
import matplotlib.pyplot as plt

# Loading the trained YOLOv8 model
model = YOLO('/content/drive/MyDrive/rdd1_yolov8_.pt')

# List of image paths for testing
IMAGE_PATHS = [
    '/content/drive/MyDrive/India2_modified/test/images/India_001197.jpg',
    '/content/road_dmg.jpg'
]

def load_image_into_numpy_array(path):
    """Loading image from file path into a numpy array."""
    return cv2.imread(path)

def resize_image(image, size=(640, 640)):
    """Resizing image to the specified size."""
    return cv2.resize(image, size)

# Looping through each image and running inference
for image_path in IMAGE_PATHS:
    print(f'Running inference for {image_path}... ', end='')
```

```
# Loading image as numpy array
image_np = load_image_into_numpy_array(image_path)

# Resizing the image to 640x640
image_resized = resize_image(image_np)

# Performing inference using YOLOv8 model
results = model.predict(source=image_resized, conf=0.4)  # Adjusting confidence threshold as needed

# Extracting results for the first image (YOLOv8 outputs in list format)
result = results[0]

# Visualizing the detections
# YOLOv8 already provides bounding boxes, class names, and confidence scores
annotated_img = result.plot()  # Getting annotated image with bounding boxes

# Showing the image with detections
plt.imshow(cv2.cvtColor(annotated_img, cv2.COLOR_BGR2RGB))
plt.title("Detection Results")
plt.axis('off')
plt.show()

print('Done')
```

```
Running inference for /content/drive/MyDrive/India2_modified/test/images/India_001197.jpg...
0: 640x640 1 D40, 617.2ms
Speed: 3.9ms preprocess, 617.2ms inference, 1.2ms postprocess per image at shape (1, 3, 640, 640)
```


```
Done
Running inference for /content/road_dmg.jpg...
0: 640x640 1 D40, 635.0ms
Speed: 3.9ms preprocess, 635.0ms inference, 1.1ms postprocess per image at shape (1, 3, 640, 640)
```



Below is the results from the pre-trained model which shows how it's results differ from the model which we have fine-tuned above

```
from ultralytics import YOLO
import numpy as np
import cv2
import matplotlib.pyplot as plt

# Loading the trained YOLOv8 model
```

```
model = YOLO('/content/drive/MyDrive/YOLOv8s_rdd.pt')  # model path (here, pre-trained)

# List of image paths for testing
IMAGE_PATHS = [
    '/content/drive/MyDrive/India2_modified/train/images/India_000209.jpg',

]

def load_image_into_numpy_array(path):
    """Loading image from file path into a numpy array."""
    return cv2.imread(path)

# Looping through each image and running inference
for image_path in IMAGE_PATHS:
    print(f'Running inference for {image_path}... ', end='')

    # Loading image as numpy array
    image_np = load_image_into_numpy_array(image_path)

    # Performing inference using YOLOv8 model
    results = model.predict(source=image_np, conf=0.4)  # Adjusting confidence threshold as needed

    # Extracting results for the first image (YOLOv8 outputs in list format)
    result = results[0]

    # Visualizing the detections
    # YOLOv8 already provides bounding boxes, class names, and confidence scores
    annotated_img = result.plot()  # Getting annotated image with bounding boxes

    # Showing the image with detections
    plt.imshow(cv2.cvtColor(annotated_img, cv2.COLOR_BGR2RGB))
    plt.title("Detection Results")
    plt.axis('off')
    plt.show()

    print('Done')
```

```
Running inference for /content/drive/MyDrive/India2_modified/train/images/India_000209.jpg...
0: 640x640 1 Alligator Crack, 8 Potholess, 666.5ms
Speed: 5.7ms preprocess, 666.5ms inference, 1.8ms postprocess per image at shape (1, 3, 640, 640)
```