Introduction

The project focuses on fine-tuning YOLOv8, a state-of-the-art object detection model, for the specific task of detecting potholes in road images. This is accomplished using a custom dataset derived from the RDD2022 dataset, which is specialized for road damage detection.

Dataset Description

- Training and Validation Sets: The final dataset, after several annotation corrections, includes 6,962 training images and 271 validation images. These images are crucial for training and evaluating the model's performance.
- Data Preparation: The dataset was unzipped and prepared for training. A significant step involved converting bounding box annotations from YOLO format to a standard format (xmin, ymin, xmax, ymax). The image dimensions were also used to denormalize the bounding box coordinates, a key step in data preprocessing.

Training Setup

- Dataset YAML: To train the model, a YAML file defining the paths to the images and the class names was necessary. This file serves as a guide for the model to understand the dataset structure and the classes involved.
- Model Architecture: YOLOv8 was chosen for its efficiency and accuracy in object detection tasks. Its ability to detect small and varied objects like potholes makes it a suitable choice for this application.

As there was no GPU available on the HPC, I ran the notebook on the runpod - https://www.runpod.io/console/gpu-secure-cloud
NVIDIA GeForce RTX 4080, 16079MiB

Ran only 5 Epochs to save the money on runpod charges per hour.

```
# Sample training for 5 epoch.
EPOCHS = 5
!yolo task=detect mode=train model=yolov8n.pt imgsz=1280 data=pothole_v8.yaml epochs={EPOCHS} batch=8 name=yolov8n_v8_50e
```

Python

```
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, 68.9MB/s]
Ultralytics YOLOv8.0.208 🚀 Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (NVIDIA GeForce RTX 4080, 16079MiB)
engine/trainer: task=detect, mode=train, model=yolov8n.pt, data=pothole_v8.yaml, epochs=5, patience=50, batch=8, imgsz=1280, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=yolov8n_v8_50e, exist_ok=Fa
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100%|██████████| 755k/755k [00:00<00:00, 18.1MB/s]
Overriding model.yaml nc=80 with nc=1

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 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    148224  ultralytics.nn.modules.block.C2f             [384, 128, 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     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
...
               all       271       716      0.484      0.328      0.342      0.149
Speed: 0.3ms preprocess, 1.3ms inference, 0.0ms loss, 3.6ms postprocess per image
Results saved to runs/detect/yolov8n_v8_50e
💡 Learn more at https://docs.ultralytics.com/modes/train
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Hyperparameter Choices

- It uses albumentation library for augmentations
- By default, it uses ADAM optimizer.

```
optimizer: AdamW(lr=0.002, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.0)
```

Results and Performance

- Training result:

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
  1/5      4.79G      2.153      4.056      1.814          9       1280: 1
             Class     Images  Instances      Box(P          R      mAP50  m
               all        271        716      0.288      0.207      0.147     0.0512

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
  2/5      4.42G      2.124      2.541      1.822         10       1280: 1
             Class     Images  Instances      Box(P          R      mAP50  m
               all        271        716      0.359      0.203      0.188       0.08

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
  3/5      4.47G      2.085      2.345      1.789          2       1280: 1
             Class     Images  Instances      Box(P          R      mAP50  m
               all        271        716      0.474      0.258      0.243      0.103

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
  4/5      4.44G      1.997      2.172      1.723          4       1280: 1
             Class     Images  Instances      Box(P          R      mAP50  m
               all        271        716      0.482      0.328      0.341      0.149

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
  5/5      4.44G      1.964      2.068      1.688          8       1280: 1
             Class     Images  Instances      Box(P          R      mAP50  m
               all        271        716       0.43      0.304       0.29      0.126
```

- Evaluation on Validation Set and Visualizations:

```
Validating runs/detect/yolov8n_v8_50e/weights/best.pt...
Ultralytics YOLOv8.0.208 🚀 Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (NVIDIA GeForce RTX 4080, 16079MiB)
Model summary (fused): 168 layers, 3005843 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  m
                   all        271        716      0.484      0.328      0.342      0.149
```

```
visualize('runs/detect/yolov8n_v8_50e_infer1280/')
```

COCO evaluation:
Ground_truth_coco_format.json file contains ground truth annotation result and Predictions_coco_format.json file contains predictions in coco format.
I have used the below code but the result was giving the error while comparing the ground truth annotation json with prediction json

```python
from tqdm import tqdm
from sahi.utils.coco import Coco, CocoCategory, CocoImage, CocoAnnotation
from sahi.utils.file import save_json

# Define your class dictionary depending on how many classes you have.
# Here, I'm assuming '0' is for potholes. Adjust as per your dataset.
class_dic = {0: "pothole"}

def yolo_to_coco(x_center, y_center, w, h, image_w, image_h):
    # Convert YOLO format to COCO format
    w = w * image_w
    h = h * image_h
    x1 = ((2 * x_center * image_w) - w) / 2
    y1 = ((2 * y_center * image_h) - h) / 2
    return [x1, y1, w, h]

def convert_to_COCO(image_path, json_name):
    coco = Coco()
    for key, item in class_dic.items():
        coco.add_category(CocoCategory(id=key, name=item))

    # Adjust the glob pattern as per your folder structure
    image_path_list = list(Path(image_path).glob("*.jpg"))
    label_path_list = [Path(str(p).replace('images', 'labels').replace('.jpg', '.txt')) for p in image_path_list]

    print(f'The number of images: {len(image_path_list)}')
    print(f'The number of labels: {len(label_path_list)}')

    no_annotation = 0
    for img_path, l_path in tqdm(zip(image_path_list, label_path_list), total=len(image_path_list)):
        ImageHeight, Imagewidth = Image.open(img_path).size
        if os.path.isfile(l_path):
            coco_image = CocoImage(file_name=str(img_path.name), height=ImageHeight, width=Imagewidth)
            with open(str(l_path)) as f:
                lines = f.readlines()
                for line in lines:
                    line = line.strip()
                    obj_class, x_center, y_center, width_yolo, height_yolo = line.split(' ')
                    bbox = yolo_to_coco(float(x_center), float(y_center), float(width_yolo), float(height_yolo), Imagewidth, ImageHeight)
                    cat_name = class_dic[int(obj_class)]
                    coco_image.add_annotation(CocoAnnotation(bbox=bbox, category_id=int(obj_class), category_name=cat_name))
            coco.add_image(coco_image)
        else:
            no_annotation += 1

    print(f"The number of images that don't have a label: {no_annotation}")
    save_path = json_name
    save_json(data=coco.json, save_path=save_path)
    print('COCO Json File Created.')

# Specify your dataset paths
image_path = '/workspace/datasets/pothole_dataset_v8/valid/images/'
json_name_file = '/workspace/datasets/pothole_dataset_v8/valid/gt3.json'
```

```python
# COCO Evaluation Code
from pycocotools.coco import COCO
from pycocotools.cocoeval import COCOeval
import json

# Load ground truth annotations
ground_truth_annotations_path = '/workspace/datasets/pothole_dataset_v8/valid/gt5.json'  # Correct path
cocoGt = COCO(ground_truth_annotations_path)

# Load YOLOv8 detections (predictions in COCO format)
detections_path = '/workspace/runs/detect/yolov8n_v8_50e_infer12803/labels/yolov8_predictions.json'  # Correct path
cocoDt = cocoGt.loadRes(detections_path)

# Running COCO evaluation
cocoEval = COCOeval(cocoGt, cocoDt, 'bbox')
cocoEval.evaluate()
cocoEval.accumulate()
cocoEval.summarize()
```

```
loading annotations into memory...
---------------------------------------------------------------------------
JSONDecodeError                           Traceback (most recent call last)
```