Reports

Approach

This report outlines the approach taken to implement an object detection system using the YOLOv8 model with the Ultralytics library. The system includes training the model, exporting it to optimized TensorRT format for inference, and performing real-time object detection on both images and videos.

Implementation Details

1. **Environment Setup**
   - The environment is configured with necessary libraries including Ultralytics for YOLO implementation, OpenCV for image and video processing, and PyYAML for YAML configuration parsing.
2. **Model Loading and Training**
   - The YOLOv8 model is loaded from a pretrained checkpoint (`yolov8n.pt`) and subsequently trained using a dataset configuration (`config.yaml`). The training process is executed for 50 epochs, with results saved to `/content/drive/MyDrive/image processing/image processing/image processing/runs/detect`.

```
Python code
model = YOLO('yolov8n.pt')
result = model.train(data="/content/drive/MyDrive/image
processing/image processing/image processing/config.yaml", epochs=50,
project='/content/drive/MyDrive/image processing/image
processing/image processing/runs/detect')
```

3. **Optimized Inference with TensorRT**
   - The trained model (`best.pt`) is exported to TensorRT format (`best.engine`) for optimized inference on a specified device (CPU).

```
Python code
model.export(format="engine", device=0)
```

4. **Object Detection on Images**
   - Object detection is performed on a single input image (`image1.jpg`) using the pretrained YOLOv8 model. Detected objects are annotated and saved back as an image with bounding boxes.

```
Python code
image_path = '/content/drive/MyDrive/image processing/image
processing/image processing/image1.jpg'
image = cv2.imread(image_path)
results = model.predict(source=image_path, save=True)
```

5. **Real-time Object Detection on Videos**
   - Object detection is applied to a video file (`videoplayback.mp4.crdownload`). The video frames are processed one by one, with bounding boxes drawn around detected objects (specifically focusing on 'person' class detections). The processed video is saved to

```
/content/drive/MyDrive/image processing/image processing/image
processing/output/output_video7.mp4.
```

```
Python code
video_path = '/content/drive/MyDrive/image processing/image
processing/image processing/videoplayback.mp4.crdownload'
output_path = '/content/drive/MyDrive/image processing/image
processing/image processing/output/output_video7.mp4'
```

Evaluation Results

- **Training Performance**: The model achieves an acceptable performance after training for 50 epochs, as indicated by the training logs and metrics (not detailed here).
- **Inference Speed**: The exported TensorRT model (`best.engine`) enhances inference speed compared to the original PyTorch model (`best.pt`), although specific benchmarking results are not provided in this report.
- **Object Detection Accuracy**: The object detection accuracy on both images and videos appears sufficient based on visual inspection of the results. However, detailed quantitative evaluation metrics (e.g., precision, recall, mAP) are not included here but would be crucial for a comprehensive evaluation.

Conclusion

This implementation successfully demonstrates the integration of YOLOv8 model with Ultralytics for object detection tasks, encompassing model loading, training, optimization with TensorRT, and real-time application on both images and videos. Further improvements could include detailed performance benchmarking and evaluation metrics to quantitatively assess model accuracy and efficiency.