Radio Burst Classification using YOLO

This documentation outlines the steps and functionality of the provided Jupyter notebook, which implements radio burst classification using YOLO (You Only Look Once) models. The notebook includes code for setting up the YOLOv10 model, training it, evaluating performance, and managing the necessary dependencies and configurations.

1. Project Overview

The project focuses on using YOLO, a real-time object detection algorithm, to classify radio bursts. YOLOv10 is a specific variant employed here, demonstrating high-speed and accurate detection capabilities.

---

2. Setup and Dependencies

2.1 Imported Libraries

The notebook uses the following libraries:

- os: For directory and file operations.
- json: For handling JSON data.
- cv2 (OpenCV): For image processing.
- yaml: To parse configuration files.
- shutil: For file management.
- glob: To manage file paths.
- ultralytics: Provides YOLO model functionalities.
- matplotlib.pyplot: For plotting and visualizations.
- random: For generating random selections.
- urllib.request: For handling URLs.

2.2 Configuration

- A YAML configuration file (data.yaml) specifies dataset details like classes and paths.

```
yaml_path = r"Data\\data.yaml"
```

---

3. YOLOv10 Process

3.1 Setting Up YOLOv10

The notebook initializes directories for weights and models. Example code for setting up weight directories:

```
weights_dir = os.path.join(os.getcwd(), "yolov10\\weights")
os.makedirs(weights_dir, exist_ok=True)
```

Pretrained weights are downloaded if necessary using URLs. This ensures the model is ready for training or inference.

3.2 Training YOLOv10 Nano

YOLOv10 Nano, a lightweight variant, is trained using the following setup:

```
model = YOLO('path_to_weights/yolov10n.pt')
model.train(data=yaml_path, epochs=30, imgsz=640, project='yolov10')
```

- data: Path to the YAML file defining dataset classes and structure.
- epochs: Number of training iterations.
- imgsz: Image size for training.
- project: Directory to save training outputs.

3.3 Evaluating YOLOv10 Nano

Evaluation involves testing the trained model on validation or test datasets to assess performance metrics such as precision, recall, and mAP (mean Average Precision).

---

4. Additional Functionality

4.1 Visualization

The notebook uses matplotlib to visualize results, such as bounding boxes over detected objects.

```
plt.imshow(image_with_boxes)
plt.show()
```

4.2 Utility Functions

Helper functions handle tasks like image loading, pre-processing, and inference.

---

5. Key Outputs

- **Trained Model Weights**: Saved in the specified directory for future use.
- **Performance Metrics**: Outputs from evaluation steps.
- **Visual Results**: Images with detected objects and classifications.

6. Conclusion

The notebook demonstrates a streamlined process for using YOLO to classify radio bursts. It includes configurations, training, evaluation, and visualization, making it a comprehensive resource for similar object detection tasks.