Wildlife Detection System: 4-Notebook Implementation Plan

Project Overview

This document outlines the comprehensive 4-notebook structure for the Wildlife Detection System. Each notebook is designed to handle a specific aspect of the wildlife detection pipeline, from data preparation to deployment-ready dashboard integration.

Directory Structure

```
/home/peter/Desktop/TU
PHD/WildlifeDetectionSystem/notebooks/training/Planned_Notebooks_v2/
├── 01_data_preparation.ipynb     # Data analysis, preprocessing, augmentation,
export
├── 02_model_training.ipynb       # Model configuration, training, checkpointing
├── 03_model_evaluation.ipynb     # Model testing, metrics calculation, error
analysis
└── 04_dashboard_integration.ipynb # Metrics formatting, dashboard file generation
```

Data Flow Between Notebooks

```
[01_data_preparation] → YOLO datasets → [02_model_training] → trained model →
[03_model_evaluation] → metrics → [04_dashboard_integration] → dashboard files
```

1. Data Preparation Notebook (COMPLETED)

Purpose: Analyze dataset characteristics, prepare balanced datasets with augmentation, and export to YOLO format.

Key Features:

- Environment verification and dependency checking
- Image dataset analysis with metadata extraction
- Class distribution analysis and visualization
- Taxonomic grouping for hierarchical classification
- Data augmentation strategies for rare species
- YOLO format export (both standard and hierarchical)

Inputs:

- Raw images in `/data/raw_images/test_01`
- Existing YOLO dataset (optional): `/data/export/yolo_default_20250429_085945`

Outputs:

- Standard YOLO dataset: `/data/export/yolo_export_{dataset}_{timestamp}`
- Hierarchical YOLO dataset: `/data/export/yolo_hierarchical_{dataset}_{timestamp}`
- Configuration data: `/config/notebook_data_{timestamp}.json`
- Export reports: `export_report.md` and `export_report.json` in export directories

2. Model Training Notebook (TO BE IMPLEMENTED)

Purpose: Configure and train YOLOv8 models with optimized hyperparameters for wildlife detection.

Key Features:

- Hardware-aware model selection (model size based on GPU/CPU)
- Memory optimization for resource-constrained environments
- Hierarchical training approach (taxonomic groups → species)
- Progressive learning with transfer learning
- Checkpointing and early stopping
- Training visualization and monitoring

Planned Implementation:

- **Cell 1**: Environment setup and hardware detection
- **Cell 2**: Model configuration and hyperparameter selection
- **Cell 3**: Dataset loading and verification
- **Cell 4**: First-stage training on taxonomic groups
- **Cell 5**: Second-stage fine-tuning on species
- **Cell 6**: Training visualization and analysis
- **Cell 7**: Model export and metadata generation

Inputs:

- YOLO datasets from Notebook 1
- Configuration from `/config/notebook_data_{timestamp}.json`

Outputs:

- Trained model: `/models/trained/wildlife_detector_{timestamp}`
- Training history: `results.csv`
- Model configuration: `args.yaml`

- Training summary: `/reports/training_summary_{timestamp}.md`

3. Model Evaluation Notebook (TO BE IMPLEMENTED)

Purpose: Comprehensively evaluate model performance with metrics calculation, threshold analysis, and error diagnostics.

Key Features:

- Multi-threshold performance evaluation
- Per-class and per-group metrics calculation
- Confusion matrix generation and analysis
- Error pattern identification and visualization
- Comparison with previous models
- Performance reports generation

Planned Implementation:

- **Cell 1**: Environment setup and model loading
- **Cell 2**: Performance metrics calculation
- **Cell 3**: Threshold sweep analysis
- **Cell 4**: Per-class and taxonomic group analysis
- **Cell 5**: Confusion matrix generation
- **Cell 6**: Error case identification and visualization
- **Cell 7**: Comparative analysis with previous models
- **Cell 8**: Comprehensive report generation

Inputs:

- Trained model from Notebook 2
- Validation dataset from Notebook 1
- Raw test images for inference

Outputs:

- Evaluation metrics: `/reports/evaluation_{timestamp}`
- Threshold analysis: `threshold_analysis.json`
- Confusion matrix: `confusion_matrix.json`
- Error analysis: `error_examples.json` and images
- Evaluation report: `evaluation_report.md`

4. Dashboard Integration Notebook (TO BE IMPLEMENTED)

Purpose: Generate properly formatted JSON files for the model performance dashboard and validate dashboard functionality.

Key Features:

- YOLOv8-compatible metrics extraction
- Column name mapping for dashboard compatibility
- JSON file generation with correct format
- Dashboard preview and verification
- Automatic file placement in model directory
- Multi-format metric export (JSON, CSV, Markdown)

Planned Implementation:

- **Cell 1**: Environment setup and model discovery
- **Cell 2**: Results parsing and metrics extraction
- **Cell 3**: Format conversion for dashboard compatibility
- **Cell 4**: JSON file generation for dashboard
- **Cell 5**: Dashboard preview and visualization
- **Cell 6**: File validation and error checking
- **Cell 7**: Deployment to model directory
- **Cell 8**: Integration testing and verification

Inputs:

- Trained model and results from Notebook 2
- Evaluation metrics from Notebook 3

Outputs:

- Dashboard files in model directory:
  - `performance_metrics.json`
  - `class_metrics.json`
  - `confusion_matrix.json`
  - `training_history.json`
  - `model_details.json`
- Dashboard preview visualizations

- Integration report

Implementation Timeline and Dependencies

Dependencies

- **Notebook 1** → First implementation, provides datasets
- **Notebook 2** → Depends on Notebook 1, provides trained model
- **Notebook 3** → Depends on Notebooks 1 & 2, provides metrics
- **Notebook 4** → Depends on Notebooks 2 & 3, finalizes dashboard

Implementation Order

1. ✅ **Notebook 1** - Data Preparation (COMPLETED)
2. 🔜 **Notebook 2** - Model Training (NEXT)
3. 🔜 **Notebook 3** - Model Evaluation
4. 🔜 **Notebook 4** - Dashboard Integration

Key Path Variables to Maintain

Consistent paths between notebooks are crucial. Always use:

```python
PROJECT_DIR = "/home/peter/Desktop/TU PHD/WildlifeDetectionSystem"
data_dir = os.path.join(PROJECT_DIR, "data")
model_save_dir = os.path.join(PROJECT_DIR, "models", "trained")
reports_dir = os.path.join(PROJECT_DIR, "reports")
```

The notebook configuration file (`notebook_data_{timestamp}.json`) provides a mechanism to pass information between notebooks, including:

- Class names
- Taxonomic groups
- Dataset paths
- Timestamps for file naming

Dashboard File Requirements

For proper dashboard functionality, the following files must be correctly formatted:

1. **performance_metrics.json**

json

```
{
  "precision": 0.637,
  "recall": 0.409,
  "mAP50": 0.505,
  "mAP50-95": 0.313,
  "training_epochs": 60,
  "best_epoch": 35,
  "classes": 30,
  "per_class": { ... },
  "thresholds": [ ... ],
  "history": { ... }
}
```

2. **class_metrics.json**

json

```
{
  "Male Roe Deer": {"precision": 0.823, "recall": 0.404, "map50": 0.713},
  "Female Roe Deer": {"precision": 0.301, "recall": 0.786, "map50": 0.322},
  ...
}
```

3. **confusion_matrix.json**

json

```
{
  "matrix": [[10, 2, 0, ...], [1, 15, 3, ...], ...],
  "class_names": ["Male Roe Deer", "Female Roe Deer", ...]
}
```

4. **training_history.json**

json

```
{
  "epoch": [1, 2, 3, 4, ...],
  "precision": [0.01, 0.05, 0.1, ...],
  "recall": [0.01, 0.05, 0.1, ...],
  "mAP50": [0.01, 0.05, 0.1, ...],
  "mAP50-95": [0.005, 0.02, 0.05, ...]
}
```

YOLOv8 Column Name Mapping

One key challenge is mapping YOLOv8's non-standard column names to dashboard-expected names:

YOLOv8 Column Name	Dashboard Expected Name
`metrics/precision(B)`	`precision`
`metrics/recall(B)`	`recall`
`metrics/mAP50(B)`	`mAP50`
`metrics/mAP50-95(B)`	`mAP50-95`

This mapping must be handled in Notebook 4 to ensure proper dashboard functionality.