Wildlife Detection System: Model Performance Dashboard Troubleshooting Summary

1. Problem Overview

The Wildlife Detection System's Model Performance Dashboard was not properly displaying metrics and visualizations despite the API endpoint returning valid data. The dashboard showed partial information (basic metrics and model details) but charts and visualizations were missing or incomplete.

2. System Architecture

Dashboard Implementation

- **Frontend**: HTML/CSS/JavaScript with Chart.js for visualizations
- **Backend**: Flask server with RESTful API endpoints
- **Key API Endpoint**: `/api/system/model-performance`
- **Debug Endpoint**: `/api/system/model-performance-debug`

Critical Files Structure

```
/home/peter/Desktop/TU PHD/WildlifeDetectionSystem/
├── api/
│   ├── app/
│   │   ├── services/
│   │   │   └── model_performance_service.py    # Core service for metrics
generation
│   │   ├── routes/
│   │   │   └── system.py                       # API endpoint definitions
│   │   └── templates/
│   │       └── model_performance.html          # Dashboard template
│   └── run.py                                 # Flask application entry point
└── models/
    └── trained/
        └── wildlife_detector_20250508_2314/    # Model directory
            ├── weights/
            │   ├── best.pt                     # Model weights
            │   └── last.pt
            ├── class_metrics.json              # Empty (problem)
            ├── confusion_matrix.json
            ├── model_comparison.json
            ├── model_details.json
            ├── performance_metrics.json        # Not properly populated (problem)
            ├── performance_summary.json
            ├── results.csv                     # Raw training data with YOLOv8
columns
            ├── training_history.json           # Incomplete (problem)
            └── various image files (.png, .jpg)
```

3. Root Cause Analysis

Key Issues Identified

1. **Empty or Malformed JSON Files**: Several critical JSON files were either empty or improperly formatted:
   - `class_metrics.json`: Empty (just `{}`)
   - `performance_metrics.json`: Not properly populated
   - `training_history.json`: Only contained epoch numbers, missing actual metric values
2. **Column Naming Discrepancy**: YOLOv8 uses non-standard column names in `results.csv`:
   - Expected: `precision`, `recall`, `mAP_0.5`
   - Actual: `metrics/precision(B)`, `metrics/recall(B)`, `metrics/mAP50(B)`
3. **Implementation Inconsistency**: The `ModelPerformanceService` correctly generated metrics dynamically when called through the API, but some dashboard components tried to read directly

from the JSON files.

Data Flow Analysis

1. The API endpoint `/api/system/model-performance` correctly returns data (confirmed via debug endpoint)
2. The `ModelPerformanceService` dynamically extracts metrics from `results.csv` using column pattern matching
3. Dashboard tried to use a mix of API data and direct file access
4. Empty/incomplete JSON files caused visualization failures

4. Data Examination

API Debug Output

The `/api/system/model-performance-debug` endpoint correctly returned:

- Full model details (name, config, etc.)
- Performance metrics (precision: 0.63729, recall: 0.409, mAP50: 0.31307)
- Per-class metrics for all 28 species
- Training history with 60 epochs

JSON Files State

- `class_metrics.json`: Empty (`{}`)
- `performance_metrics.json`: Not properly populated
- `confusion_matrix.json`: Present but potentially incorrect format
- `training_history.json`: Only contained epoch numbers `{"epoch": [1,2,3,...,60]}`

CSV Data

`results.csv` contained all raw training data with YOLOv8-specific column names:

- `epoch`
- `time`
- `train/box_loss`
- `train/cls_loss`
- `train/dfl_loss`
- `metrics/precision(B)`
- `metrics/recall(B)`

- `metrics/mAP50(B)`
- `metrics/mAP50-95(B)`
- etc.

5. Solution Implemented

Solution Approach

Created a Python script to update all JSON files with properly formatted data derived from:

1. API endpoint response data
2. Raw values from `results.csv` with column name mapping
3. Properly structured format for dashboard consumption

Key Script Components

python

```python
# Model directory path
model_path = "/home/peter/Desktop/TU PHD/WildlifeDetectionSystem/models/trained/wildli

# Update performance_metrics.json with complete data structure
performance_metrics = {
  "precision": 0.63729,
  "recall": 0.409,
  "mAP50": 0.31307,
  "mAP50-95": 0,
  "training_epochs": 60,
  "best_epoch": 35,
  "per_class": { /* class metrics */ },
  "history": { /* training history */ }
}

# Extract per_class data for class_metrics.json
class_metrics = performance_metrics["per_class"]

# Map YOLOv8 column names to expected names
column_mapping = {
    "metrics/precision(B)": "precision",
    "metrics/recall(B)": "recall",
    "metrics/mAP50(B)": "mAP50",
    "metrics/mAP50-95(B)": "mAP50-95"
}

# Create confusion matrix data
confusion_matrix = {
    "matrix": [...],  # Matrix data
    "class_names": list(class_metrics.keys())
}

# Write the files
with open(os.path.join(model_path, "performance_metrics.json"), "w") as f:
    json.dump(performance_metrics, f, indent=2)

with open(os.path.join(model_path, "class_metrics.json"), "w") as f:
    json.dump(class_metrics, f, indent=2)

with open(os.path.join(model_path, "training_history.json"), "w") as f:
    json.dump(training_history, f, indent=2)
```

6. Technical Details

Model Information

- **Model Name**: wildlife_detector_20250508_2314
- **Type**: YOLOv8n
- **Training Epochs**: 60 (Best epoch: 35)
- **Image Size**: 320x320
- **Key Metrics**:
  - Precision: 0.63729
  - Recall: 0.409
  - mAP50: 0.31307
  - mAP50-95: 0

ModelPerformanceService Implementation

The service uses a dynamic column detection approach to handle different YOLOv8 output formats:

```python
# Find correct metric columns by pattern matching
for col in results_df.columns:
    col_lower = col.lower()
    if 'precision' in col_lower:
        precision_col = col
    elif 'recall' in col_lower:
        recall_col = col
    elif 'map50' in col_lower or 'map_0.5' in col_lower:
        map50_col = col
    elif 'map50-95' in col_lower or 'map_0.5:0.95' in col_lower:
        map50_95_col = col
```

Dashboard Data Requirements

For full functionality, the dashboard requires properly formatted JSON files:

1. `performance_metrics.json`: Overall metrics and history data
2. `class_metrics.json`: Per-class precision, recall, and mAP50 values
3. `confusion_matrix.json`: Matrix data and class names
4. `training_history.json`: Complete epoch-by-epoch metrics

7. Long-term Improvements

Suggested Enhancements

1. **Fully Dynamic Model**: Modify dashboard to rely exclusively on API data and never directly access files
2. **Automatic File Generation**: Update ModelPerformanceService to automatically write correct JSON files after training
3. **File Validation**: Add validation checks to ensure JSON files are properly formatted
4. **Error Handling**: Improve dashboard error handling to gracefully handle missing or malformed data

Documentation Improvements

1. Document expected file formats and structures
2. Add schema validation for JSON files
3. Create a troubleshooting guide for common dashboard issues