Wildlife Detection System - ML Development Phase

Project Current Status (April 29, 2025)

The Wildlife Detection System project has reached a significant milestone - all annotation work is complete, and we're now transitioning to the ML development phase. Here's the current state of the project:

- **Annotation Phase**: 100% complete (1882 images fully annotated)
- **Data Export**: Successfully exported to YOLO format with 80/20 train/validation split
- **Species Organization**: 30 unique species identified and categorized
- **Admin & Tools**: All admin interfaces and export/reporting tools functioning
- **Next Phase**: Machine Learning model development with YOLOv8

Dataset Analysis

The exported YOLO dataset (from report_yolo_default_20250429_085425_20250429_085427.pdf) has the following characteristics:

Dataset Structure

- **Total Images**: 445 images
- **Training Images**: 356 (80%)
- **Validation Images**: 89 (20%)
- **Total Annotations**: 452 bounding boxes
- **Training Annotations**: 361
- **Validation Annotations**: 91
- **Classes**: 30 unique species

Class Distribution

The distribution of annotations across species shows an imbalanced dataset:

Class	Train	Val	Total	Train %	Val %
Rabbit	108	27	135	80.0%	20.0%
Female Roe Deer	66	14	80	82.5%	17.5%
Male Roe Deer	56	23	79	70.9%	29.1%
Human	53	13	66	80.3%	19.7%
Fox	36	8	44	81.8%	18.2%
Jackal	14	4	18	77.8%	22.2%
Other species	28	2	30	93.3%	6.7%

Many species have very few examples (1-4 instances), which reinforces the need for our hierarchical classification approach.

Data Preprocessing

- During the export process, the system automatically filtered out full-image "Background" annotations (1437) from the admin panel
- Only the 452 actual wildlife annotations were included in the YOLO export
- This filtering is methodologically correct for object detection training

YOLOv8 Training Setup

I'm now configuring the YOLOv8 training environment and preparing the model architecture. The training will be implemented in `/home/peter/Desktop/TU PHD/WildlifeDetectionSystem/notebooks/training/wildlife_model.ipynb`.

Training Strategy

I've developed a multi-stage training approach to address the class imbalance problem:

1. **Stage 1: Parent Category Model**
   - Group species into broader taxonomic categories:
     - Deer (Red Deer, Male Roe Deer, Female Roe Deer, Fallow Deer)
     - Carnivores (Wolf, Fox, Jackal, etc.)
     - Small Mammals (Rabbit, Hare, Squirrel, etc.)
     - Birds (Blackbird, Nightingale, Pheasant, etc.)
   - Train YOLOv8 to recognize these broader categories first
   - Focus on high detection accuracy regardless of specific species
2. **Stage 2: Fine-tuning for Specific Species**
   - Use the Stage 1 model weights as starting point
   - Fine-tune on species with sufficient examples (Rabbit, Deer, Fox, etc.)

- Apply transfer learning with lower learning rate
- Implement class-weighted loss functions to handle imbalance

3. **Stage 3: Specialized Models (if needed)**
   - Train specialized models for particular taxonomic groups
   - Implement ensemble methods for improved accuracy

YOLOv8 Configuration

The configuration for YOLOv8 training includes:

```python
# YOLOv8 base configuration
model_config = {
    'model': 'yolov8m.pt',  # Medium-sized model for balance of speed and accuracy
    'imgsz': 640,           # Input image size
    'batch': 16,            # Batch size
    'epochs': 100,          # Maximum epochs
    'patience': 20,         # Early stopping patience
    'optimizer': 'Adam',    # Optimizer
    'lr0': 0.001,           # Initial learning rate
    'lrf': 0.01,            # Final learning rate factor
    'momentum': 0.937,      # SGD momentum/Adam beta1
    'weight_decay': 0.0005, # Optimizer weight decay
    'warmup_epochs': 3.0,   # Warmup epochs
    'warmup_momentum': 0.8, # Warmup initial momentum
    'warmup_bias_lr': 0.1,  # Warmup initial bias lr
    'box': 7.5,             # Box loss gain
    'cls': 0.5,             # Cls loss gain
    'hsv_h': 0.015,         # Image HSV-Hue augmentation (fraction)
    'hsv_s': 0.7,           # Image HSV-Saturation augmentation (fraction)
    'hsv_v': 0.4,           # Image HSV-Value augmentation (fraction)
    'degrees': 0.0,         # Image rotation (+/- deg)
    'translate': 0.1,       # Image translation (+/- fraction)
    'scale': 0.5,           # Image scale (+/- gain)
    'fliplr': 0.5,          # Image flip left-right (probability)
    'mosaic': 1.0,          # Image mosaic (probability)
    'mixup': 0.0            # Image mixup (probability)
}
```

Hardware Resources

- Local training on NVIDIA GeForce RTX 4050 Laptop GPU
- CUDA 12.4 support configured
- 16GB RAM and SSD storage for efficient data processing

Data Augmentation Strategies

To address the limited examples for some species, I'm implementing several data augmentation techniques:

1. **Standard Augmentation**
   - Random horizontal flips
   - Small translations and scaling
   - Mosaic augmentation (combining 4 images)
   - HSV color space adjustments
2. **Wildlife-Specific Augmentation**
   - Simulating different lighting conditions (day/night)
   - Adding motion blur to mimic camera trap conditions
   - Crop and zoom to simulate partial visibility
   - Noise addition to simulate low-light conditions
3. **Class Balancing**
   - Oversampling rare species
   - Instance-aware augmentation (more augmentation for rare classes)
   - Copy-paste augmentation for very rare species

Evaluation Framework

The evaluation framework will measure:

1. **Detection Metrics**
   - mAP50 (mean Average Precision at 50% IoU)
   - mAP50-95 (mean Average Precision at IoU from 50% to 95%)
   - Precision and Recall per class
   - F1-score per class
2. **Classification Accuracy**
   - Confusion matrix between species
   - Top-1 and Top-3 accuracy for species identification
   - Hierarchical classification accuracy
3. **Inference Performance**
   - Frames per second on target hardware
   - Memory usage during inference
   - Model size and deployment requirements

Next Steps

1. **Configure and Train Base Model** (Current Task)
   - Finish setting up wildlife_model.ipynb with proper configuration
   - Train initial model on parent categories
   - Validate detection accuracy on test set
2. **Optimize Model for Wildlife-Specific Challenges**
   - Handle partially visible animals
   - Improve detection in low-light conditions
   - Fine-tune for small animals and distant subjects
3. **Integrate with Main System**
   - Create inference API for the trained model
   - Develop visualization tools for detection results
   - Implement batch processing for large image collections

Questions for Model Development

1. What augmentation strategies would be most effective for wildlife detection in camera trap images?
2. How to best handle the extreme class imbalance where some species have 100+ examples while others have only 1-3?
3. What's the optimal hierarchical grouping for the 30 species to maximize training effectiveness?
4. Are there any pre-trained weights specifically for wildlife detection that might provide better starting points than standard COCO weights?
5. Should we implement any domain-specific loss functions for better handling of wildlife detection challenges?