Wildlife Detection System - Complete Project Overview

Current Project Status (April 2025)

The Wildlife Detection System has successfully completed initial data annotation phase and is now transitioning to model training using YOLOv8 architecture. The system combines annotation tools, data management, and AI model development for wildlife monitoring from camera trap images.

Development Phase: Phase 2 - Model Training & Enhanced Detection

Implementation Status:

- ✅ Infrastructure (100% Complete)
- ✅ Database Schema (100% Complete)
- ✅ Admin Interface (100% Complete)
- ✅ Annotation Tools (100% Complete)
- 🔄 Data Processing & Annotation (85% Complete)
- 🔄 ML Model Development (In Progress)
- ⏳ Advanced Features (Planned)

System Purpose & Core Functionality

This system allows wildlife researchers to:

- Manage large collections of camera trap images (currently ~1,800+ indexed)
- Annotate wildlife with bounding boxes and species labels
- Export annotations in standard formats (YOLO, COCO) for ML training
- Track annotation progress across multiple datasets
- Analyze wildlife behavior patterns and habitat usage
- Generate insights about species distribution and activity patterns

Technology Stack

Backend

- Flask (Python 3.12.3)
- SQLite Database
- RESTful API architecture

ML Development

- PyTorch 2.6.0+cu124

- Ultralytics YOLOv8 (v8.3.106)
- OpenCV 4.11.0
- Supporting packages: numpy, matplotlib, pandas

Hardware

- Development workstation with NVIDIA GeForce RTX 4050 GPU
- CUDA 12.4 support

System Architecture

```
WildlifeDetectionSystem/
├── api/                    # Flask backend
│   ├── app/                # Application source
│   │   ├── models/         # Database models
│   │   ├── routes/         # API endpoints
│   │   ├── services/       # Business logic
│   │   ├── static/         # Static files (annotation UIs)
│   │   ├── templates/      # HTML templates
│   │   └── utils/          # Helper utilities
│   ├── config.py           # Configuration
│   ├── debug/              # Debugging tools
│   ├── instance/           # SQLite database
│   └── run.py              # Application entry point
├── data/
│   ├── raw_images/         # Original camera trap images
│   │   └── test_01/        # First dataset (~1,882 images)
│   ├── export/             # Export directory for ML data
│   │   └── yolo_export/    # YOLO format exported dataset
│   │       ├── images/     # Copied images
│   │       ├── labels/     # Annotation text files
│   │       └── classes.txt # Class definitions
│   └── processed_images/   # Standardized images
├── models/
│   └── trained/            # Directory for trained models (ready)
└── notebooks/
    ├── training/           # Model training notebooks
    │   └── wildlife_model.ipynb # Active training notebook
    ├── evaluation/         # Model evaluation notebooks
    └── utils/              # Utility notebooks
```

Database Schema

The system uses SQLite with the following core tables:

Image Table

Stores metadata about each camera trap image.

Column	Type	Description
id	INTEGER	Primary key
filename	VARCHAR(255)	Relative path to image
original_path	VARCHAR(512)	Absolute path to original image
processed_path	VARCHAR(512)	Path to processed version (if any)
width	INTEGER	Image width in pixels
height	INTEGER	Image height in pixels
upload_date	DATETIME	When image was added to system
location	VARCHAR(255)	Geographic location data
timestamp	DATETIME	When image was captured
camera_id	VARCHAR(100)	ID of the camera trap

Species Table

Contains wildlife species definitions.

Column	Type	Description
id	INTEGER	Primary key
name	VARCHAR(100)	Common name (e.g., "Red Deer")
scientific_name	VARCHAR(255)	Latin name
description	TEXT	Additional information

Annotation Table

Stores bounding box annotations linking images to species.

Column	Type	Description
id	INTEGER	Primary key
image_id	INTEGER	Foreign key to Image
species_id	INTEGER	Foreign key to Species
x_min	FLOAT	Left boundary (0-1 normalized)
y_min	FLOAT	Top boundary (0-1 normalized)
x_max	FLOAT	Right boundary (0-1 normalized)
y_max	FLOAT	Bottom boundary (0-1 normalized)
confidence	FLOAT	Prediction confidence (for ML)
is_verified	BOOLEAN	Human verification flag
created_at	DATETIME	Creation timestamp
updated_at	DATETIME	Last update timestamp

Additional Tables

- **EnvironmentalData**: Environmental factors for diurnal activity analysis
- **BehavioralNote**: Notes about animal behavior for behavioral tracking
- **SequenceEvent**: Tracks chronological appearances for predator-prey analysis

Core Features & Components

1. Annotation Interfaces

The system includes multiple annotation interfaces:

- **Advanced Annotator**: Full-featured interface with drawing tools, keyboard shortcuts, zoom/pan functionality
- **Ground Truth Annotator**: Three-panel layout similar to Amazon SageMaker
- **Simple Annotator**: Simplified version for basic tasks

Key features:

- Smart tool for drawing and manipulating bounding boxes
- Species selection panel
- "No Animals Present" functionality
- Keyboard shortcuts for efficient workflow
- Progress tracking

2. Admin Interface

Recently improved admin interface with:

- Species management (fixed to properly create/edit species)
- Image management with thumbnail previews
- Annotation management with relation visualization
- Environmental data management
- Database browser and SQL query tool
- Statistics dashboard showing annotation progress

3. Export Functionality

The system can export annotations in multiple formats:

- **COCO Format**: JSON format with images, categories, and annotations
- **YOLO Format**: Text files with normalized coordinates for direct training

4. Species Classification System

The system currently manages approximately 25 species:

- Deer species (Male Roe Deer, Female Roe Deer, Fallow Deer)
- Carnivores (Wolf, Fox, Jackal, Brown Bear)
- Mustelids (Badger, Weasel, Stoat, Polecat, Marten, Otter)
- Small mammals (Rabbit, Hare, Squirrel, Dormouse, Hedgehog)
- Other wildlife (Wild Boar, Chamois, Turtle)
- Birds (Blackbird, Nightingale, Pheasant)
- Human

NEW - Hierarchical Classification Approach:

To address challenges with too many specific classes and limited samples, we're implementing a hierarchical classification system that:

1. Preserves detailed annotations in the database
2. Maps specific species to broader parent categories for model training
3. Enables a multi-stage training approach (broad categories → specific species)

Proposed hierarchy:

- **Deer**: Male Roe Deer, Female Roe Deer, Fallow Deer
- **Large Carnivores**: Wolf, Brown Bear, Wildcat
- **Small Carnivores**: Fox, Jackal
- **Mustelids**: Badger, Weasel, Stoat, Polecat, Marten, Otter

- **Small Mammals**: Rabbit, Hare, Squirrel, Dormouse, Hedgehog
- **Wild Ungulates**: Wild Boar, Chamois
- **Birds**: Blackbird, Nightingale, Pheasant
- **Reptiles**: Turtle
- **Human**: Human

This approach allows for more robust model training while preserving detailed annotations.

Recent Updates & Improvements

1. **Admin Interface Fix**
   - Implemented proper `SpeciesForm` class to resolve form creation issues
   - Added validation and descriptions for species fields
   - Enhanced admin templates for better usability
2. **Hierarchical Classification Design**
   - Developed approach for managing many species classes
   - Created strategy for class mapping during YOLO export
   - Designed multi-stage training approach
3. **Annotation Interface Enhancements**
   - Improved bounding box manipulation
   - Added keyboard shortcuts and zoom functionality
   - Fixed coordinate normalization issues
4. **Export System Optimization**
   - YOLO export now properly handles coordinates
   - Added support for dataset splitting
   - Prepared for hierarchical classification

Development Roadmap

Phase 1: Basic Recognition (Completed)

- ✅ Image indexing system with over 1,800 camera trap images
- ✅ Three functional annotation interfaces
- ✅ Data export functionality (COCO and YOLO formats)
- ✅ Successful annotation and export for training

Phase 2: Model Training & Enhanced Detection (Current)

- 🔄 Configure YOLOv8 training parameters

- 🔄 Implement hierarchical classification export
- 🔄 Train initial model on annotated dataset
- 🔄 Validate on test set
- ⏳ Analyze model performance
- ⏳ Fine-tune hyperparameters
- ⏳ Implement detection of partially visible animals

Phase 3: Advanced Analysis (Planned for Q3-Q4 2025)

- ⏳ Diurnal and seasonal activity analysis
- ⏳ Environmental & microhabitat analysis
- ⏳ Chronological tracking
- ⏳ Behavior analysis patterns

ML Training Strategy

With the new hierarchical approach, a multi-stage training process will be implemented:

1. **Stage 1: Broad Classification**
   - Train YOLOv8 model on parent categories
   - Focus on high detection accuracy for broader groups
2. **Stage 2: Fine-tuning for Specific Species**
   - Use the model from Stage 1 as starting point
   - Fine-tune on specific species with sufficient data
   - Apply transfer learning with lower learning rate
3. **Stage 3: Specialized Models**
   - Train specialized models for specific animal families
   - For example, a "deer specialist" model for deer species
   - Useful for high-accuracy needs in particular groups

Challenges & Future Work

1. **Improving detection accuracy in varying lighting conditions**
   - Addressing day/night camera trap images
   - Handling overexposed and underexposed shots
2. **Handling partially visible animals and motion blur**
   - Detecting animals from partial body parts
   - Recognizing blurry silhouettes (especially wolf/jackal differentiation)

3. **Implementing chronological tracking features**
    - Predator-prey movement patterns
    - Timing analysis between species appearances
4. **Developing the microhabitat analysis module**
    - Vegetation classification
    - Environmental condition assessment
5. **Scaling to additional datasets**
    - Expanding beyond the initial test dataset
    - Incorporating diverse habitats and conditions

Success Criteria

The project's success will be measured by:

- **Accuracy**: Target mAP ≥ 75% for primary wildlife species
- **Performance**: Image processing under 1 second on GPU
- **Robustness**: Reliable detection in varying conditions
- **Efficiency**: ≥ 80% reduction in manual analysis time

Getting Started

Running the Server

```bash
cd ~/Desktop/TU\ PHD/WildlifeDetectionSystem/api
export FLASK_APP=run.py
export FLASK_DEBUG=1
flask run
```

Accessing the Annotation Interface

Open a web browser and navigate to:

```
http://127.0.0.1:5000/advanced-annotator
```

Accessing the Admin Interface

```
http://127.0.0.1:5000/admin/
```

Next Immediate Steps

1. **Complete Annotation of Current Dataset**
    - Continue annotating all images in test_01
    - Use consistent bounding box placement
2. **Implement Hierarchical Classification Export**
    - Update the YOLO export endpoint to support class mapping
    - Test export with different hierarchical configurations
3. **Begin YOLOv8 Model Training**
    - Configure YOLOv8 training parameters
    - Train initial model on annotated dataset
    - Validate on test set

Acknowledgments

- Project developed in collaboration with Prof. Peeva
- Inspired by needs in wildlife conservation research
- Based on camera trap technology for non-invasive wildlife monitoring