Wildlife Detection System: Comprehensive Project Report & Development Plan

Executive Summary

This comprehensive report documents the current state, achievements, and future plans for the "Wildlife Detection System" project, which aims to develop an AI-powered platform for automatically detecting and analyzing wildlife from camera trap images. The system has successfully processed 1,882 images from the Sliven Balkan region (provided by Nikolay Dolapchiev), and created 1,889 annotations across 30 species classes. The current best-performing model (YOLOv8n) has achieved a mAP50 of 0.623 with strong precision (0.788) but more moderate recall (0.514).

The report details both technical achievements and limitations, identifies significant opportunities for improvement through new datasets (iWildCam-2022, MIT River Herring, UNSW-predators), and outlines a concrete development plan to meet Prof. Peeva's specialized requirements for wildlife research.

1. Current System Architecture

The Wildlife Detection System consists of several interconnected components developed to form a complete pipeline from data ingestion to analysis:

1.1 Backend System

- **Flask Application Server**: RESTful API endpoints for all system functions
- **SQLite Database**: Structured storage for images, species, annotations and environmental data
- **File Storage System**: Organized hierarchy for raw and processed images

1.2 User Interfaces

- **Annotation Tools**: Advanced web interfaces for precise wildlife labeling
- **Administrative Dashboard**: System monitoring and management tools
- **ML Workbench**: Jupyter notebooks for model training and evaluation
- **Bilingual Support**: Full Bulgarian and English language interfaces

1.3 Machine Learning Pipeline

- **Data Preprocessing**: Tools for image standardization and augmentation
- **Training Pipeline**: Configurable training workflows for YOLO models
- **Evaluation System**: Comprehensive metrics and visualizations for model assessment
- **Export Functionality**: Support for standard formats (COCO, YOLO) used in ML

1.4 Hierarchical Classification Approach

- **Taxonomic Grouping**: Species organized into 5 primary groups:
  - Deer (Red Deer, Male Roe Deer, Female Roe Deer, Fallow Deer)
  - Carnivores (Fox, Wolf, Jackal, etc.)
  - Small Mammals (Rabbit, Hare, Squirrel, etc.)
  - Birds (Blackbird, Nightingale, Pheasant, Woodpecker)
  - Other (Wild Boar, Chamois, Turtle, Human, etc.)

2. Dataset and Annotation Analysis

2.1 Current Dataset Statistics

The system has processed 1,882 camera trap images with the following annotation distribution:

Category	Count	Percentage
Background (no animals)	1,437	76.1%
Rabbit	135	7.1%
Female Roe Deer	80	4.2%
Male Roe Deer	79	4.2%
Human	66	3.5%
Fox	44	2.3%
Jackal	18	1.0%
Badger	4	0.2%
Weasel	4	0.2%
Squirrel	4	0.2%
Wildcat	3	0.2%
Turtle	3	0.2%
Blackbird	3	0.2%
Dog	3	0.2%
Fallow Deer	2	0.1%
Woodpecker	2	0.1%
Red Deer	1	0.1%
Nightingale	1	0.1%

2.2 Class Imbalance Analysis

The dataset exhibits severe class imbalance:

- Top 3 wildlife species (Rabbit, Female Roe Deer, Male Roe Deer) account for 65% of all wildlife annotations

- 11 species have fewer than 5 annotations each
- Taxonomic imbalance: Birds category has only 6 total annotations

2.3 Data Organization and Split Strategy

- **Training Set**: 356 images (75%)
- **Validation Set**: 89 images (25%)
- **Directory Structure**: Standardized YOLO format with images and labels folders
- **Quality Issues**: Several corrupted JPEG files requiring restoration

3. Machine Learning Models and Performance

3.1 Model Architecture Evolution

The project has experimented with several YOLOv8 model variants:

Model Variant	Parameters	Image Size	Advantages	Disadvantages
YOLOv8n	3.0M	416×416	Fast, memory efficient	Lower accuracy
YOLOv8s	11.2M	640×640	Better accuracy	Higher memory requirements
YOLOv8m	25.9M	640×640	Strong accuracy	Slower, resource intensive

Current production model: **YOLOv8n** with customized training parameters.

3.2 Best Model Configuration

- **Base Architecture**: YOLOv8n with transfer learning from COCO dataset
- **Input Size**: 416×416 pixels (optimized for GPU memory constraints)
- **Loss Weightings**: Box loss (7.5), Class loss (3.0), DFL loss (1.5)
- **Training Parameters**:
  - Epochs: 100 with early stopping (patience: 25)
  - Batch Size: 2 (memory optimized)
  - Optimizer: AdamW with initial learning rate 0.001
  - Data Augmentation: Mosaic (1.0), Mixup (0.1), Copy-paste (0.1)
  - GPU Memory Optimizations: PYTORCH_CUDA_ALLOC_CONF with expandable_segments

3.3 Performance Metrics

The best model demonstrates the following metrics:

- **mAP50-95**: 0.413 (average precision across IoU thresholds)
- **mAP50**: 0.623 (precision at IoU threshold 0.5)
- **Precision**: 0.788 (overall precision)

- **Recall**: 0.514 (overall recall)

3.4 Per-Class Performance Analysis

Performance varies significantly across species:

- **Excellent Performance** (mAP50 > 0.75):
  - Human: mAP50 = 0.914, Precision = 0.900, Recall = 0.900
  - Rabbit: mAP50 = 0.799, Precision = 0.838, Recall = 0.764
  - Male Roe Deer: mAP50 = 0.764, Precision = 0.760, Recall = 0.687
  - Jackal: mAP50 = 0.748, Precision = 0.743, Recall = 0.728
- **Moderate Performance** (0.45 < mAP50 < 0.75):
  - Female Roe Deer: mAP50 = 0.575, Precision = 0.550, Recall = 0.571
  - Fox: mAP50 = 0.469, Precision = 0.500, Recall = 0.375
- **Poor Performance** (mAP50 < 0.45):
  - Weasel: mAP50 = 0.000, Precision = 0.000, Recall = 0.000
  - Wildcat: mAP50 = 0.000, Precision = 0.000, Recall = 0.000
  - All other rare species with fewer than 5 annotations

3.5 Confidence Threshold Analysis

The model's performance varies with different confidence thresholds:

Threshold	mAP50	Precision	Recall
0.50	0.506	0.532	0.441
0.25	0.534	0.536	0.503
0.10	0.545	0.788	0.514
0.05	0.613	0.788	0.514

Lower thresholds (0.05-0.10) provide optimal balance between precision and recall, particularly for rare species.

3.6 Real-World Testing Results

Testing on 20 previously unseen images showed:

- 35% detection rate (7 images with successful detections)
- Average processing time: 0.0295 seconds per image
- Detection distribution by group: Deer (55.6%), Other (33.3%), Small Mammals (11.1%)

4. System Interface and Admin Tools

4.1 Main Dashboard Interface

The system dashboard provides a comprehensive overview:

- **System Status**: Total images (1,882), annotated images, species count (30), folder count
- **Quick Access Panels**:
  - Annotation Tools
  - Export Tools
  - ML Workbench
  - Administrative Tools

4.2 Admin Monitoring System Enhancement (Planned)

We're developing enhanced model monitoring capabilities for the Admin interface:

- **Current Model Status Panel**:
  - Currently loaded model (architecture, size, date trained)
  - Key performance metrics (mAP50, precision, recall)
  - Resource utilization (memory, GPU)
  - Inference speed (ms/image)
- **Model Comparison Dashboard**:
  - Side-by-side comparison of model versions
  - Performance trends over time
  - Class-specific metric changes
  - Confidence threshold visualization
- **Real-Time Model Diagnostics**:
  - Confusion matrix visualization
  - Failure case analysis
  - Detection confidence distribution
  - Processing pipeline performance
- **Deployment Controls**:
  - Model version selection
  - Confidence threshold adjustment
  - Batch processing configuration
  - Export format selection

This enhancement will give Prof. Peeva and researchers complete visibility into model performance and allow fine-tuning without technical expertise.

4.3 Annotation Interface Capabilities

The annotation system provides specialized tools:

- **Advanced Annotator**: Precision bounding box tools with zoom and pan
- **Seasonal Analysis**: Tools for temporal analysis (to be expanded)
- **Environmental Editor**: Habitat and environmental condition tracking
- **Batch Processing**: Tools for working with multiple images

4.4 Export and Reporting System

The system features comprehensive reporting tools:

- **Export Formats**: COCO and YOLO standard formats
- **Index Management**: Tools for adding new images
- **Species Management**: Interface for managing taxonomy
- **Report Generation**: Customizable report output

5. Notebooks and ML Tools

5.1 Current Notebook System

The project currently utilizes two primary Jupyter notebooks:

1. **Training Notebook**:
   - Data loading and preprocessing
   - Model configuration
   - Training execution
   - Basic evaluation
2. **Evaluation and Analysis Notebook**:
   - Detailed model evaluation
   - Class performance analysis
   - Confidence threshold testing
   - Visualization generation

5.2 Planned Third Notebook: Wildlife Monitoring and Environmental Analysis

We're developing a third specialized notebook focused on Prof. Peeva's requirements:

- **Seasonal and Annual Activity Analysis**:
  - Temporal patterns by species and taxonomic group
  - Day/night activity ratio analysis

    - Seasonal migration and behavior changes
    - Annual comparison of wildlife presence
- **Microhabitat Analysis**:
    - Vegetation type classification
    - Habitat preference by species
    - Environmental condition correlation
    - Snow cover and weather impact analysis
- **Chronological Tracking**:
    - Species appearance sequence analysis
    - Predator-prey relationship visualization
    - Time interval metrics between species
    - Territorial overlap analysis
- **Behavioral Pattern Recognition**:
    - Basic behavioral classification (feeding, passing, resting)
    - Group dynamics analysis
    - Interaction detection between species
    - Partial visibility and movement analysis

This notebook will integrate with the database to leverage all available metadata and provide the specialized analysis capabilities required by Prof. Peeva.

6. Challenges and Limitations

6.1 Data-Related Challenges

- **Class Imbalance**: Extreme disparity between common and rare species
- **Annotation Quality**: Some bounding boxes extend beyond image boundaries
- **Limited Environmental Data**: Insufficient metadata about lighting conditions, habitat types, and seasonal factors
- **Fragmented Chronology**: Incomplete temporal sequences for tracking animal movements

6.2 Technical Challenges

- **Image Quality Issues**: Motion blur, night images, corruption in JPEG files
- **GPU Memory Constraints**: Limited by 6GB VRAM on RTX 4050 laptop GPU
- **Processing Bottlenecks**: Balance between throughput and accuracy
- **Data Transfer Limitations**: Large dataset handling challenges

6.3 Domain-Specific Challenges

- **Silhouette Recognition**: Difficulty with blurred or partial animal shapes
- **Species Differentiation**: Challenges distinguishing similar species (wolf/jackal)
- **Extreme Lighting Variation**: Performance degradation in low-light conditions
- **Microhabitat Classification**: Limited data for environmental analysis

7. New Dataset Analysis and Integration Plan

7.1 Available New Datasets

Three promising datasets have been identified:

1. **iWildCam-2022.zip** (169 GB):
   - Global wildlife camera trap dataset
   - Contains diverse species across varied environments
   - Includes comprehensive metadata on time and conditions
   - Potential source for rare species examples
2. **mit_river_herring.zip** (38.7 GB):
   - Specialized aquatic wildlife dataset from MIT
   - Contains examples of herring and other fish species
   - May have limited direct relevance but useful techniques
   - High-quality annotations and methodology
3. **unsw-predators.json.zip** (1.72 MB):
   - Focused dataset on predator species from UNSW
   - Particularly valuable for carnivores category
   - JSON format suggests rich metadata
   - Compact size indicates well-curated data

7.2 Strategic Integration Approach

Rather than complete re-annotation, we propose a selective enhancement approach:

1. **Analysis Phase**:
   - Extract metadata structure from each dataset
   - Identify images with relevant species for our taxonomic groups
   - Focus on underrepresented classes (e.g., Wildcat, Weasel, birds)
   - Assess lighting conditions and environmental variety
2. **Selection Criteria**:

- Target species with < 5 examples in current dataset
- Diverse lighting conditions (night, twilight, daylight)
- Various microhabitats matching Bulgaria's ecosystems
- Quality of annotations and metadata

3. **Integration Process**:
   - Convert annotations to our standard format
   - Add to database with source dataset attribution
   - Preserve original metadata and enhance with our schema
   - Maintain separation for controlled evaluation

7.3 Specific Dataset Enhancement Targets

Our integration will focus on:

1. **From iWildCam**:
   - Carnivore examples (fox, wolf, wildcat)
   - Night and twilight images
   - Sequence captures for chronological analysis
2. **From UNSW-predators**:
   - Detailed predator annotations
   - Behavioral metadata
   - Interaction patterns
3. **Methodology from MIT**:
   - Advanced annotation techniques
   - Temporal analysis approaches
   - Detection confidence calibration

7.4 Technical Integration Steps

1. **Dataset Parsing**:

```python
# Example pseudo-code for dataset integration
def process_iwildcam(zip_path, output_dir):
    # Extract archive
    with ZipFile(zip_path) as archive:
        archive.extractall(temp_dir)

    # Parse metadata
    metadata = json.load(open(os.path.join(temp_dir, 'metadata.json')))

    # Filter for target species
    target_species = ['fox', 'wolf', 'wildcat', 'weasel', 'marten']
    selected_images = []

    for image_id, data in metadata.items():
        if any(species in data['categories'] for species in target_species):
            selected_images.append((image_id, data))

    # Process and convert annotations
    for image_id, data in selected_images:
        # Convert to our format
        convert_annotation(data, output_dir)

    return len(selected_images)
```

2. **Database Schema Updates**:

```sql
-- Add metadata fields for new environmental data
ALTER TABLE image ADD COLUMN light_condition VARCHAR(50);
ALTER TABLE image ADD COLUMN habitat_type VARCHAR(100);
ALTER TABLE image ADD COLUMN snow_cover BOOLEAN;
ALTER TABLE image ADD COLUMN source_dataset VARCHAR(100);

-- Create chronology tracking tables
CREATE TABLE sequence_event (
    id INTEGER PRIMARY KEY,
    location VARCHAR(255),
    species_id INTEGER REFERENCES species(id),
    timestamp DATETIME NOT NULL,
    previous_event_id INTEGER REFERENCES sequence_event(id),
    time_since_previous INTEGER
);
```

8. Implementation Plan for Prof. Peeva's Requirements

8.1 Requirements Analysis

Based on our meetings with Prof. Peeva, the following key requirements have been identified:

1. **Seasonal and Annual Activity Analysis** (not monthly patterns)
2. **Recognition of Challenging Cases**:
   - Partial views of animals
   - Body parts and distinctive features
   - Blurred silhouettes of fast-moving animals
   - Similar species differentiation
3. **Adaptation to Various Conditions**:
   - Different habitats (plains/mountains)
   - Varying lighting conditions
   - Different camera positions
4. **Microhabitat Analysis**:
   - Tree species recognition
   - Vegetation density assessment
   - Snow melt patterns
5. **Behavioral and Chronological Tracking**:
   - Multi-object recognition
   - Predator following patterns
   - Time intervals between species
   - Appearance sequence analysis

8.2 Implementation Strategy

To meet these requirements, we're implementing a three-phase approach:

Phase 1: Basic Recognition (Current)

- Identify complete animals in good conditions
- Integrate with existing labeled image database
- Basic differentiation of target species

Phase 2: Advanced Recognition (Next 3-4 Weeks)

- Recognize partial silhouettes and characteristic features
- Adapt to various imaging conditions
- Integrate environmental factors

Phase 3: Complex Analysis (Following 4-5 Weeks)

- Diurnal and seasonal activity
- Species interaction tracking
- Behavioral pattern analysis

8.3 Technical Implementation Specifics

1. **Light Condition Classification Module**:

   python

   ```python
   def classify_light_condition(image_path):
       img = cv2.imread(image_path)
       gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
       brightness = np.mean(gray)

       if brightness < 40:
           return "night"
       elif brightness < 120:
           return "twilight"
       else:
           return "daylight"
   ```

2. **Microhabitat Analysis Module**:

python

```python
def analyze_microhabitat(image_region):
    # Color segmentation for vegetation
    hsv = cv2.cvtColor(image_region, cv2.COLOR_BGR2HSV)

    # Green vegetation
    lower_green = np.array([35, 50, 50])
    upper_green = np.array([90, 255, 255])
    green_mask = cv2.inRange(hsv, lower_green, upper_green)

    # Calculate coverage percentages
    green_percent = np.sum(green_mask > 0) / (green_mask.shape[0] * green_mask.shap

    # Determine dominant vegetation type
    if green_percent > 0.3:
        if brown_percent > 0.2:
            return "Mixed forest"
        else:
            return "Dense vegetation"
    elif brown_percent > 0.3:
        return "Sparse trees"
    else:
        return "Open area"
```

3. **Chronological Tracking Module**:

```
python
```

```
def track_species_sequence(location, days_lookback=30):
    """Track species appearance sequence at a location."""
    cutoff_date = datetime.utcnow() - timedelta(days=days_lookback)

    # Get all images with annotations at the location, ordered by timestamp
    annotations = (db.session.query(Annotation, Image, Species)
                   .join(Image, Annotation.image_id == Image.id)
                   .join(Species, Annotation.species_id == Species.id)
                   .filter(Image.location == location)
                   .filter(Image.timestamp >= cutoff_date)
                   .order_by(Image.timestamp)
                   .all())

    # Process into sequence events
    events = []
    prev_event = None

    for annotation, image, species in annotations:
        # Skip background annotations
        if species.name.lower() == 'background':
            continue

        # Create sequence event
        event = SequenceEvent(
            location=location,
            species_id=species.id,
            timestamp=image.timestamp
        )

        # Link to previous event if it exists
        if prev_event:
            event.previous_event_id = prev_event.id

            # Calculate time difference in seconds
            if prev_event.timestamp:
                time_diff = (image.timestamp - prev_event.timestamp).total_seconds(
                event.time_since_previous = int(time_diff)

        db.session.add(event)
        events.append(event)
        prev_event = event

    db.session.commit()
```

```
        # Return sequence data
        return events
```

9. Extended Roadmap and Timeline

9.1 Immediate Actions (Within 1-2 Weeks)

1. **Data Enhancement**:
   - Parse new datasets and identify valuable examples
   - Correct problematic annotations in current dataset
   - Implement light condition classification
   - Add initial environmental metadata
2. **Admin Monitoring Panel**:
   - Develop model status monitoring dashboard
   - Implement real-time performance tracking
   - Add model switching functionality
   - Create visualization components
3. **Third Notebook Framework**:
   - Initialize structure for environmental analysis notebook
   - Implement basic seasonal activity tracking
   - Set up microhabitat analysis framework
   - Create visualization templates

9.2 Short-Term Goals (3-4 Weeks)

1. **Enhanced Training**:
   - Integrate selected examples from new datasets
   - Implement advanced augmentation techniques
   - Train models with balanced class representation
   - Optimize for different lighting conditions
2. **Advanced Recognition**:
   - Implement partial visibility detection
   - Train classifiers for microhabitat analysis
   - Develop silhouette recognition for fast-moving animals
   - Improve similar species differentiation
3. **System Architecture Updates**:
   - Extend database schema for environmental data

- Implement chronological tracking system
- Create specialized visualization tools
- Optimize data processing pipeline

9.3 Medium-Term Goals (1-2 Months)

1. **Complex Analysis Systems**:
   - Complete seasonal and annual activity analysis
   - Implement full chronological tracking
   - Develop comprehensive habitat analysis
   - Create behavioral pattern recognition
2. **Integration and Testing**:
   - Unify all system components
   - Conduct extensive testing with real data
   - Optimization for performance
   - Documentation and training materials
3. **Publication Preparation**:
   - Document methodology and results
   - Prepare comparative analysis with existing approaches
   - Highlight innovations in wildlife monitoring
   - Draft academic paper with Prof. Peeva

10. Conclusion and Next Steps

The Wildlife Detection System has made significant progress with a functional annotation platform and model training pipeline. The current YOLOv8n model demonstrates promising results for common species but requires improvement for rare species and challenging conditions. The existing 1,882 images provide a foundation, but strategic enhancement with selected examples from new datasets will be crucial.

10.1 Immediate Priorities

1. **Begin Dataset Integration**: Start analyzing the three new datasets (iWildCam-2022, MIT River Herring, UNSW-predators) to identify valuable examples for enhancing our training data.
2. **Implement Admin Monitoring System**: Develop the planned admin panel for real-time model monitoring and performance tracking.
3. **Initialize Third Notebook**: Begin development of the environmental analysis notebook focusing on Prof. Peeva's requirements.

10.2 Key Metrics for Success

1. **Model Performance**:
   - Improve mAP50 from 0.623 to >0.75
   - Achieve >0.5 mAP50 for all species with >5 examples
   - Maintain >0.7 precision while improving recall to >0.65
2. **Functional Capabilities**:
   - Successfully classify images by light condition with >90% accuracy
   - Implement microhabitat analysis for at least 4 habitat types
   - Develop chronological tracking with species sequence analysis
3. **System Usability**:
   - Complete admin monitoring dashboard
   - Streamline annotation workflow
   - Enhance visualization capabilities

By methodically addressing these priorities and focusing on the specific requirements outlined by Prof. Peeva, we can transform the Wildlife Detection System from a basic annotation and detection tool into a comprehensive platform for wildlife research and conservation.

Report generated: May 5, 2025