INSTAWILD

DS SUCHITRA

PES1PG23CA037

GUIDE: MS. SAMYUKTA D KUMTA

ASSISTANT PROFESSOR

MASTER OF COMPUTER APPLICATIONS

INSTAWILD

ABSTRACT

- The system detect animals in images captured by camera, and gives us a count on how many animals are present in each image.
- The system can describe activities of the animals.
- The system process images, saving time and effort compared to manual analysis(traditional approach).
- But imagine when a system does all these for researches/wildlife conservationists, it allows for better and more efficient monitoring of wildlife, helping researchers track animal populations and behavior.

PROBLEM SCENARIO

- Understanding animal populations and behavior is crucial for wildlife research and conservation. Researchers need to track species like deer, elephants, and zebras to study their movement patterns, health, and interactions with their environment. However, observing and collecting data on these animals in the wild is challenging due to vast habitats, unpredictable movements, and environmental factors.Without precise tracking and behavioral analysis, researchers struggle to assess population trends, detect health issues, and understand the impact of environmental changes on these species.

PROPOSED SYSTEM

- The proposed system automates wildlife analysis by processing images to identify animals, count them, and analyze their behavior. It begins by cleaning raw images for better accuracy, count individuals, and classify behavior. The system generates detailed reports, summarizing species, counts, and activities. This saves time, improves accuracy, and provides unique behavior insights, unlike traditional methods. Scalable and efficient, it’s a game-changer for wildlife conservation and research.

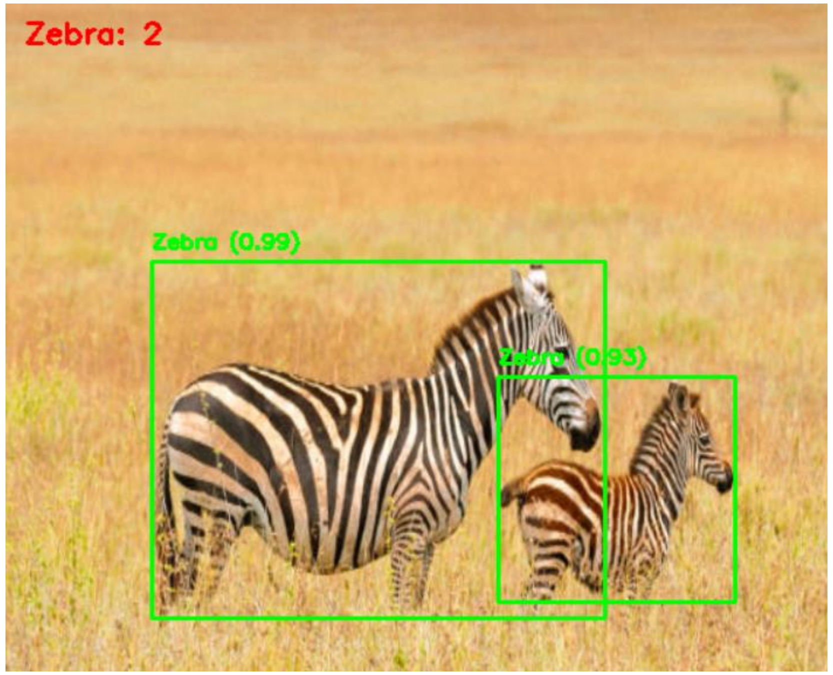
PURPOSE AND SCOPE

- The purpose of this project is to make wildlife monitoring easier, faster, and more accurate for researchers and conservationists. Traditional methods of analyzing images to identify animals, count them, and study their behaviors are slow and often error-prone. This system automates the process, helping researchers focus on understanding wildlife rather than spending hours sorting through images.

- The scope of the project includes creating a tool that can process images from camera traps, identify different animal species, count how many animals are present, and even describe the activity of the animal. Ultimately, this system supports conservation efforts by providing detailed, reliable insights into animal populations and behaviors, making it easier to track trends and protect wildlife.

DOMAIN SURVEY

In wildlife research, tracking animal populations and behaviors provides critical insights into ecosystem health. Deer move in herds, elephants form close herds or parades, and zebras gather in dazzles—each exhibiting unique social structures and behaviors. Deer are skittish and follow hierarchical movement patterns, elephants display strong familial bonds and coordinated decision-making, while zebras rely on group dynamics for survival. Studying these behaviors helps researchers understand migration, stress factors, and habitat changes. Leveraging machine learning, I aim to automate animal detection, counting, and behavior analysis, enabling efficient, large-scale wildlife monitoring.



DETAILED LITERATURE SURVEY

Title : A deep active learning system for species identification and counting in camera trap images

Authors : Mohammad Sadegh Norouzzadeh, Dan Morris, Sara Beery, Neel Joshi, Nebojsa Jojic, Jeff Clune

Published in: 14 October 2020

A deep active learning system for species identification and counting in camera trap images - Norouzzadeh - 2021 - Methods in Ecology and Evolution - Wiley Online Library

Summary: The paper presents an innovative approach to wildlife monitoring. The authors introduce an active deep learning system that significantly reduces manual labour in processing camera trap images by over 99.5%, achieving accuracy with minimal manual labeling. This automation enhances the efficiency of extracting information from large datasets, facilitating the deployment of extensive camera trap networks for wildlife ecology and conservation.

DETAILED LITERATURE SURVEY

Title: Intelligent Detection Method for Wildlife Based on Deep Learning

Authors: Shuang Li, Haiyan Zhang, Fu Xu

Published in: *Sensors*, December 7, 2023

Intelligent Detection Method for Wildlife Based on Deep Learning - PMC

Summary: This paper introduces TMS-YOLO, an enhanced version of the YOLOv7 deep learning model, tailored for wildlife detection in complex outdoor environments. By incorporating optimized modules and attention mechanisms, TMS-YOLO effectively addresses challenges such as insufficient lighting, occlusion, and image blurriness. Experimental results demonstrate that TMS-YOLO outperforms the standard YOLOv7 model, achieving higher mean Average Precision (mAP) scores on both self-built and Turkish wildlife datasets, thereby offering a more accurate tool for wildlife monitoring and conservation efforts.

DETAILED LITERATURE SURVEY

Title: Automated Detection and Counting of Wild Boar in Camera Trap Images

Authors: Anne K. Schütz, Helen Louton, Mareike Fischer, Carolina Probst, Jörn M. Gethmann, Franz J. Conraths.

Published in: *Animals*, May 8, 2024

Automated Detection and Counting of Wild Boar in Camera Trap Images

Summary: This study demonstrates the effectiveness of automated computer vision techniques in wildlife monitoring. By training an algorithm on 1,600 images from wildlife studies, the model achieved a mean average precision of 98.11% in detecting various animal classes in their natural environments. This automation significantly reduces the labor-intensive process of manual image analysis, enhancing the efficiency of wildlife management and conservation efforts.

DETAILED LITERATURE SURVEY

Title: AI-Driven Wildlife Behaviour Monitoring Using Computer Vision

Authors: Pandiselvi R, Jeyaprabhu J, Jerome Immanuel Jebaraj J, Muthupandi L

Published in: International Journal for Multidisciplinary Research, Volume 6, Issue 5, September-October 2024

Summary: This paper presents an AI-driven system using computer vision and YOLOv8 to monitor wildlife behavior in real-time. It detects and classifies animal activities such as feeding, resting, and social interactions, while mapping patterns like migration routes. The system provides real-time alerts for unusual behaviors and generates detailed visual reports for long-term analysis. This approach aims to enhance wildlife conservation through proactive and data-driven management.

DETAILED LITERATURE SURVEY

Title: Target Detection and Cow Standing Behavior Recognition Based on YOLOv5 Algorithm

Authors: Xin Tian, Bomeng Li, Xiaodong Cheng, Xiangyang Shi

Published in: 2022 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS)

Summary: This paper introduces a method to identify and analyze cow standing behavior using the YOLOv5 algorithm. By leveraging a lightweight YOLOv5s model, the system detects cows in real time with 97.6% accuracy. It processes video frames efficiently and extracts standing behavior, overcoming challenges like occlusion, overlapping cows, and variable lighting. This approach reduces labor and improves the monitoring of dairy cow health and activity.

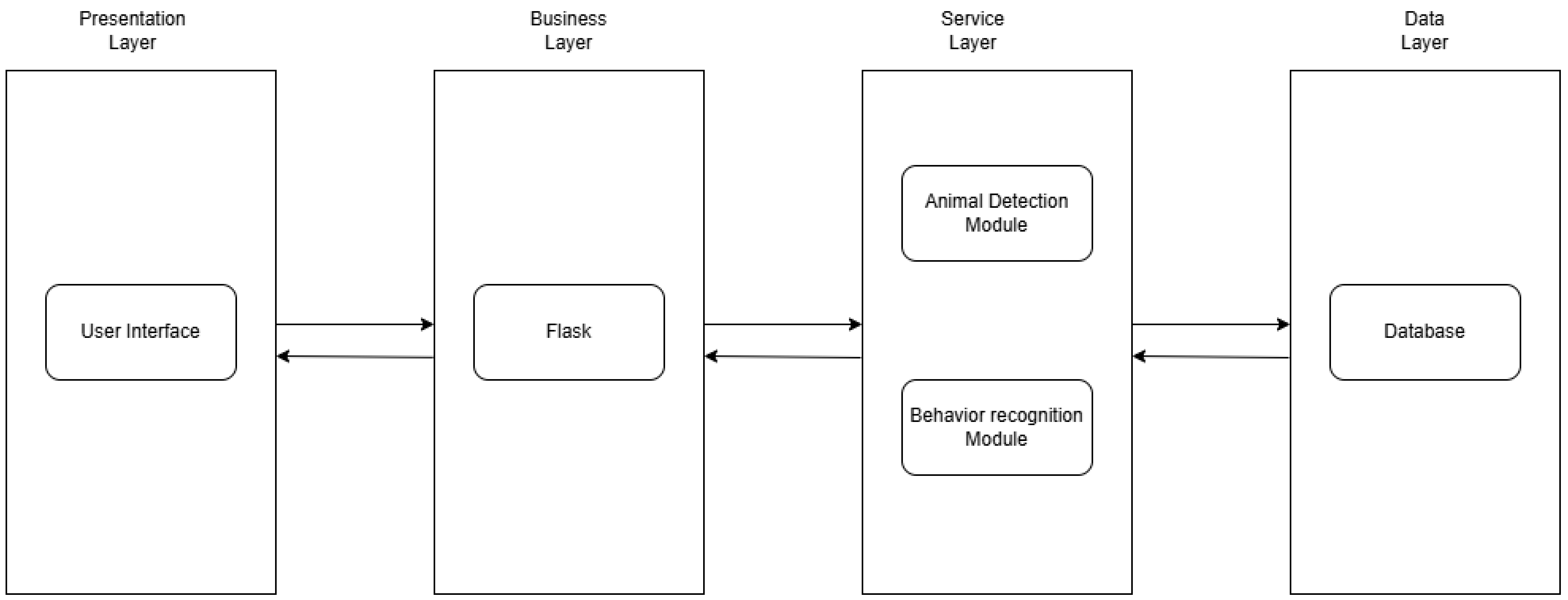
FUNCTIONALITIES

- **Image Preprocessing:** Raw images of animals are cleaned, organized, and prepared for analysis(removing noise, resizing).
- **Image Detection:** The system will detect whether user is giving image as input.
- **Animal Detection:** The system will be able to identify animals from the images.
- **Animal Counting:** It counts how many animals are present in each image.
- **Behavior Identification:** The system can describe what the animals are doing, such as resting, in motion , eating, or interacting with each other.

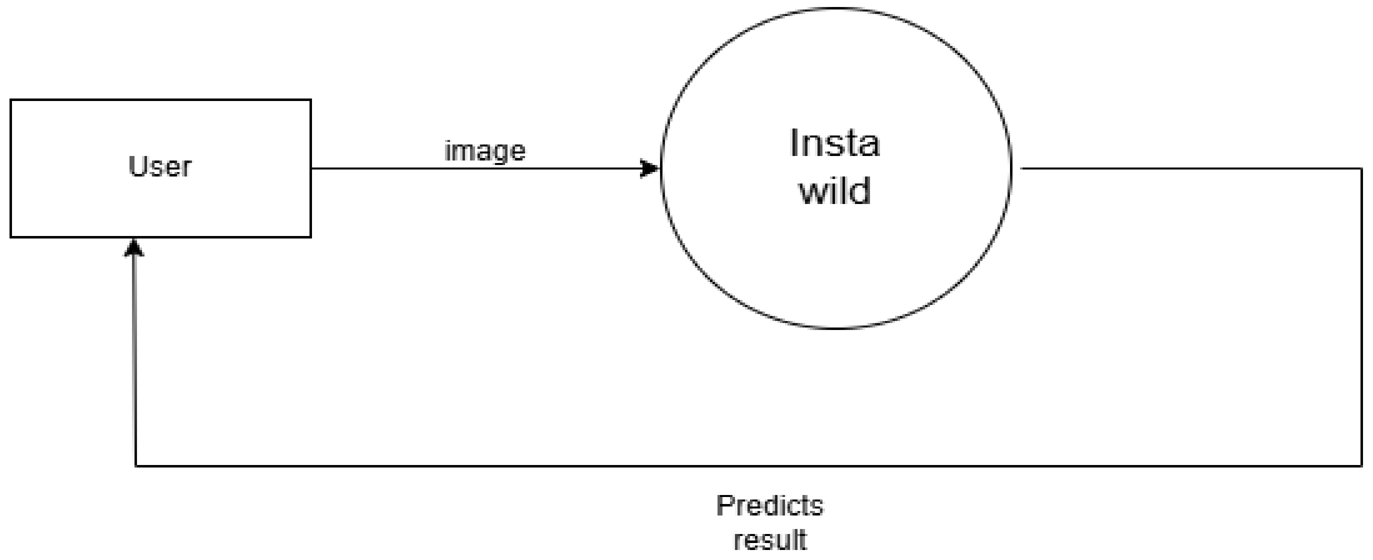
NON-FUNCTIONALITIES

- **Usability:** The application is easy to use and includes a user-friendly web-based interface built with Flask. The system provides clear instructions and error messages to assist users in navigating and resolving issues.
- **Compatibility :** The application is designed to run on multiple operating systems, including Windows and Ubuntu, with Python and necessary libraries (e.g., OpenCV, PyTorch) installed.
- **Performace Matrix:** The performance of the system will be evaluated using accuracy, precision, recall, F1-score, processing speed, and robustness. These metrics ensure reliable animal detection, counting, and behavior recognition, even under challenging conditions like occlusions, varying lighting, and environmental changes.

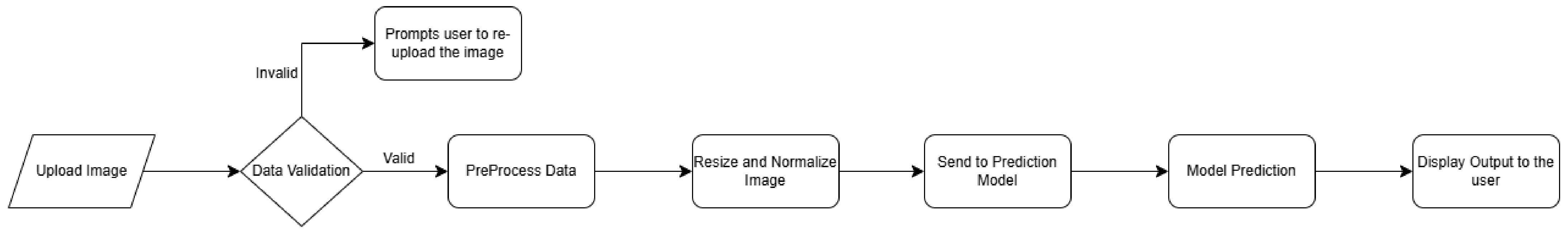
ARCHITECTURE DIAGRAM



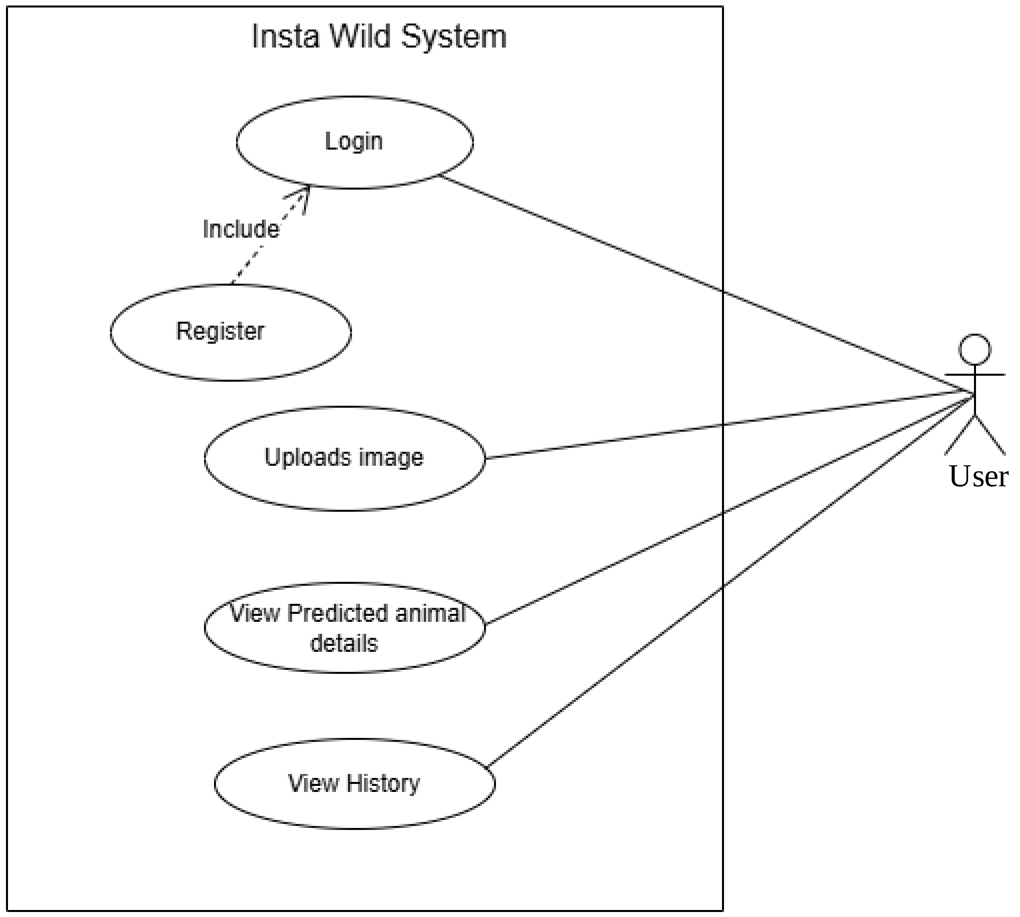
CONTEXT DIAGRAM



PROCESS FLOW



USE CASE DIAGRAM



DETAILED METHODOLOGY

Data Collection:

Gathering images from online sources (like Kaggle, web scraping, roboflow) and real-world wildlife camera footage.

Labeling data, using roboflow (semi-supervised) manually and using automated annotation tools, to create bounding boxes around animals.

Data Preprocessing: Cleaning & Organizing the Dataset:

Removing corrupt images to avoid errors during training.

Resizing images to 224X224 pixels for consistency.

Normalizing pixel values(0-1) to help the model learn effectively.

Spliting data into train(70%), validation(20%) and test(10%) sets for a balanced learning process.

DETAILED METHODOLOGY

Model Building:

YOLOV8

The YOLOv8 model training process begins by importing the yolov8l.pt model, the dataset is defined using data.yaml, specifying train, validation, and test paths, along with 3 classes: Deer, Elephant, andZebra. The model is trained for epochs controls how long the model trains with a batch size (optimized for CPU) and an image size of 416x416 pixels to balance speed and performance. Training results, including the best model weights (best.pt), logs, and accuracy metrics, are stored in runs/animal detector/. Since training runs on CPU (device="cpu"). The model applies data augmentation and backpropagation to improve accuracy, with the best-performing model saved automatically. Post-training, the model can be evaluated using validation metrics (mAP, precision, recall) and used for real-time animal detection.

DETAILED METHODOLOGY

Prediction:

The trained model is then tested on new images:

YOLOv8 processes images, drawing bounding boxes around detected animals.

Animal count is tracked for each detected species.

Results are displayed on images, with labels and confidence scores.

TOOLS AND TECHNOLOGIES

HARDWARE REQUIREMENTS

- **Processor:** Intel i5/i7 or AMD Ryzen 5/7
- **RAM:** Minimum 16 GB
- **Storage:** SSD with at least 512 GB

TOOLS AND TECHNOLOGIES

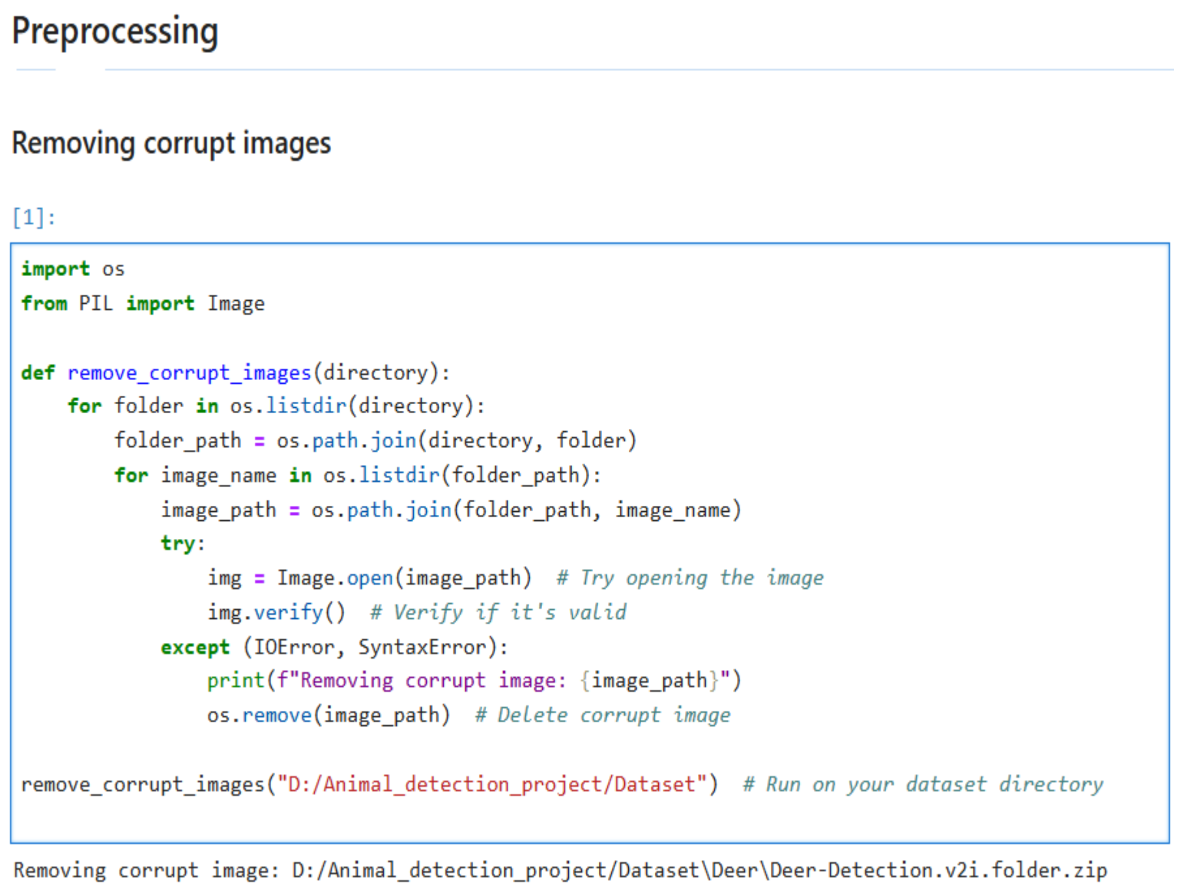
SOFTWARE REQUIREMENTS

- **Computer Vision**: OpenCV
- **Object Detection:** YOLO (You Only Look Once) YOLOv7 and YOLOv8, PyTorch
- **Programming Language:** Python 3.10
- **User Interface (UI):** ReactJS 19.0.0
- **Database:** MongoDB 8.0.4
- **Labeling :** roboflow

APPLICATIONS IN REAL WORLD

- **Wild-life Tracking:** Enables conservationists to track deer, elephants, and zebras in national parks and wildlife reserves.
- **Population Estimation:** Helps in counting animal populations to analyze their distribution and movement patterns.
- **Anti-Poaching Measures:** Identifies illegal human activities to prevent poaching of elephants and zebras.
- **Elephant Crop Raiding Prevention:** Detects elephants entering farmlands and triggers deterrents (lights, sounds, drones) to prevent crop destruction.
- **Deer Damage Control:** Identifies deer in agricultural areas to minimize crop losses due to grazing.

IMPLEMENTATION



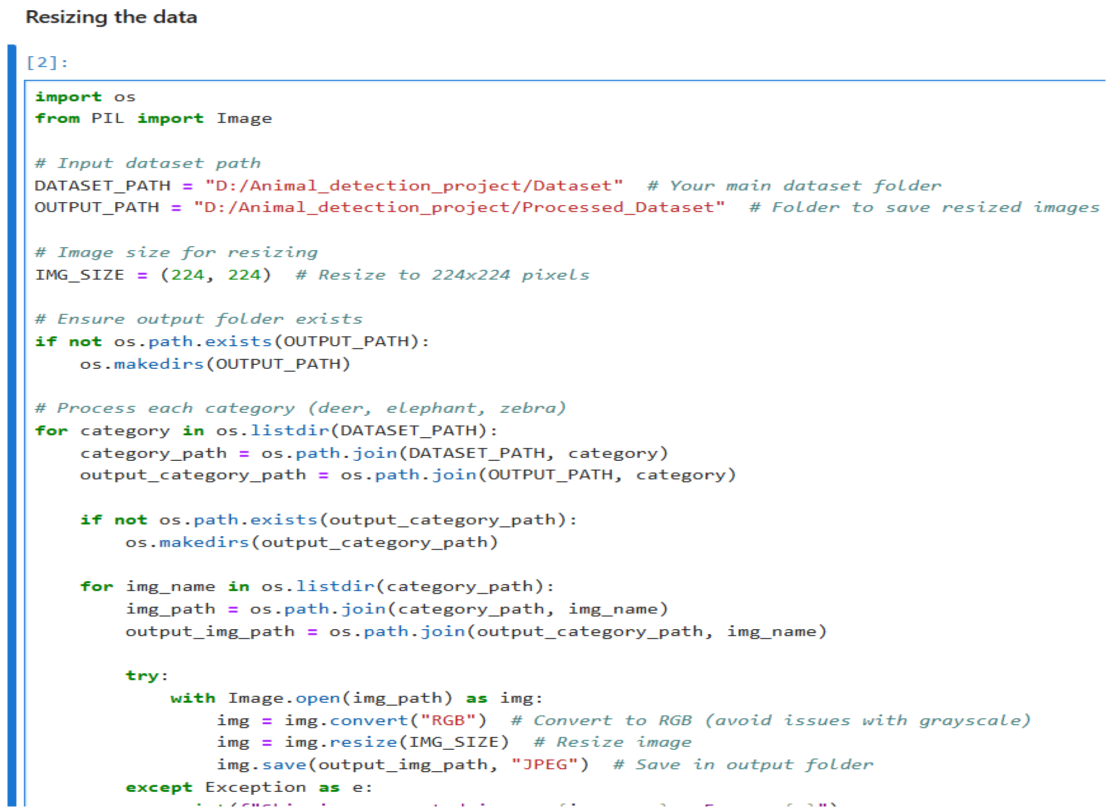
Preprocessing

Removing corrupt images

[1]:

```python
import os
from PIL import Image

def remove_corrupt_images(directory):
    for folder in os.listdir(directory):
        folder_path = os.path.join(directory, folder)
        for image_name in os.listdir(folder_path):
            image_path = os.path.join(folder_path, image_name)
            try:
                img = Image.open(image_path)  # Try opening the image
                img.verify()  # Verify if it's valid
            except (IOError, SyntaxError):
                print(f"Removing corrupt image: {image_path}")
                os.remove(image_path)  # Delete corrupt image

remove_corrupt_images("D:/Animal_detection_project/Dataset")  # Run on your dataset directory
```

```
Removing corrupt image: D:/Animal_detection_project/Dataset\Deer\Deer-Detection.v2i.folder.zip
```

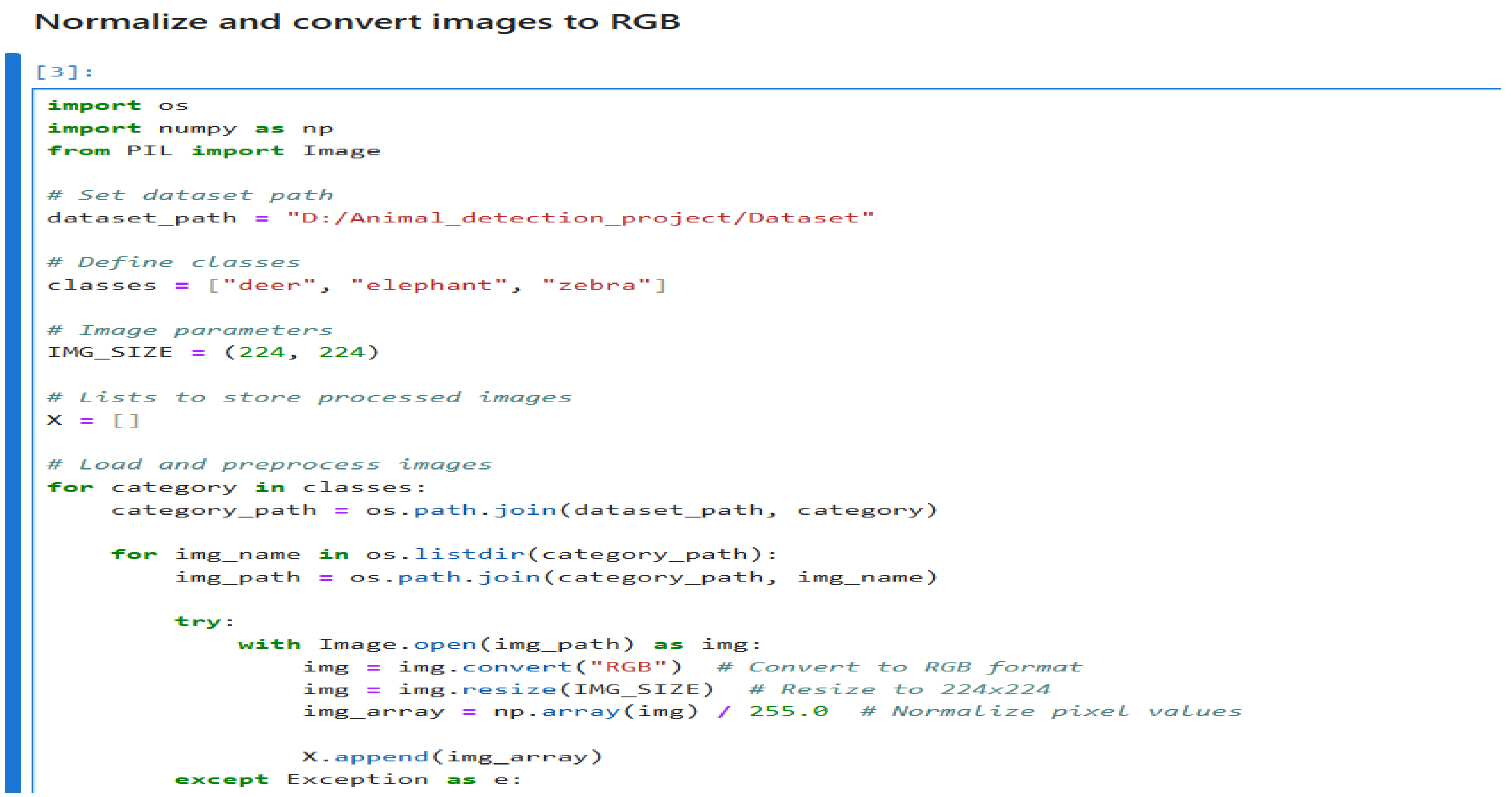


Resizing the data

[2]:

```python
import os
from PIL import Image

# Input dataset path
DATASET_PATH = "D:/Animal_detection_project/Dataset"  # Your main dataset folder
OUTPUT_PATH = "D:/Animal_detection_project/Processed_Dataset"  # Folder to save resized images

# Image size for resizing
IMG_SIZE = (224, 224)  # Resize to 224x224 pixels

# Ensure output folder exists
if not os.path.exists(OUTPUT_PATH):
    os.makedirs(OUTPUT_PATH)

# Process each category (deer, elephant, zebra)
for category in os.listdir(DATASET_PATH):
    category_path = os.path.join(DATASET_PATH, category)
    output_category_path = os.path.join(OUTPUT_PATH, category)

    if not os.path.exists(output_category_path):
        os.makedirs(output_category_path)

    for img_name in os.listdir(category_path):
        img_path = os.path.join(category_path, img_name)
        output_img_path = os.path.join(output_category_path, img_name)

        try:
            with Image.open(img_path) as img:
                img = img.convert("RGB")  # Convert to RGB (avoid issues with grayscale)
                img = img.resize(IMG_SIZE)  # Resize image
                img.save(output_img_path, "JPEG")  # Save in output folder
        except Exception as e:
```

IMPLEMENTATION

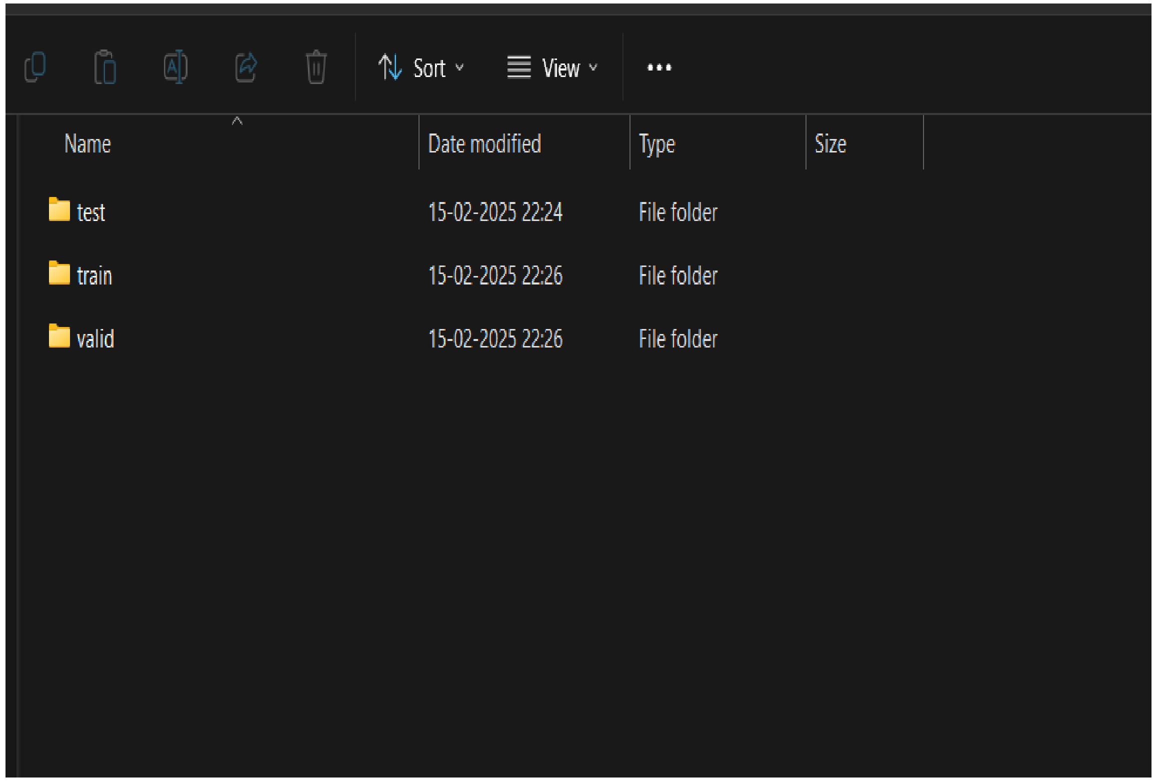


Normalize and convert images to RGB

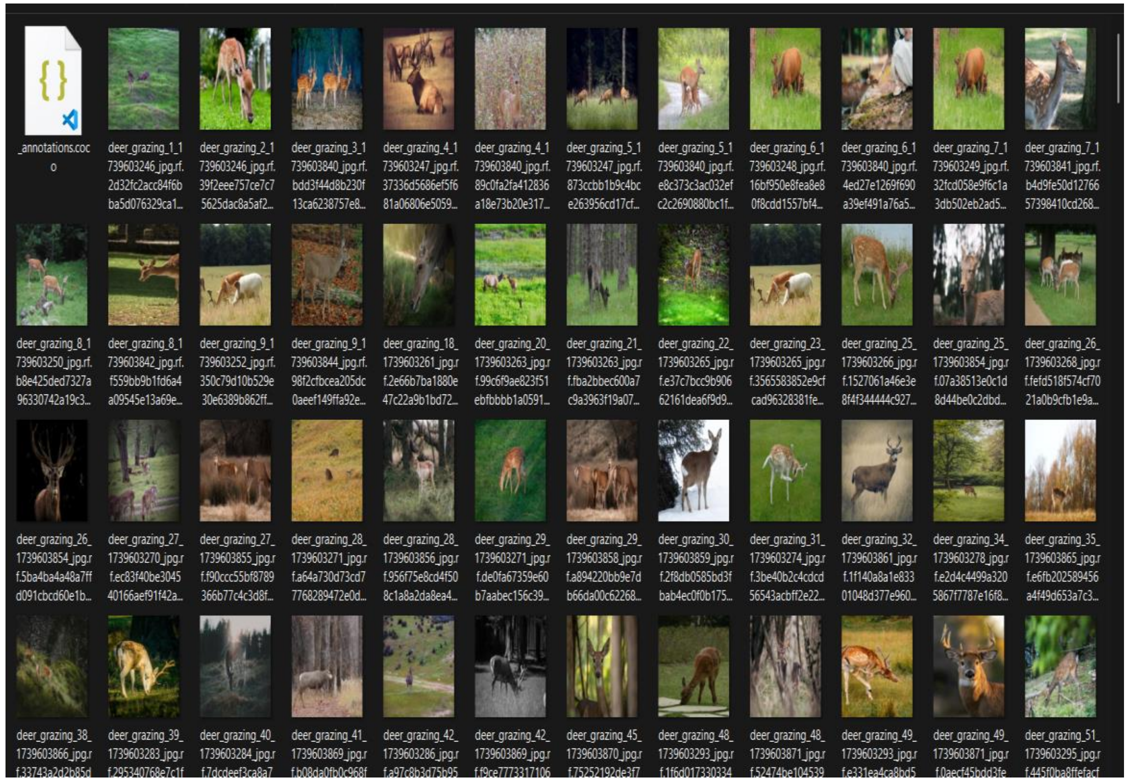
[3]:

```
import os
import numpy as np
from PIL import Image

# Set dataset path
dataset_path = "D:/Animal_detection_project/Dataset"

# Define classes
classes = ["deer", "elephant", "zebra"]

# Image parameters
IMG_SIZE = (224, 224)

# Lists to store processed images
X = []

# Load and preprocess images
for category in classes:
    category_path = os.path.join(dataset_path, category)

    for img_name in os.listdir(category_path):
        img_path = os.path.join(category_path, img_name)

        try:
            with Image.open(img_path) as img:
                img = img.convert("RGB")  # Convert to RGB format
                img = img.resize(IMG_SIZE)  # Resize to 224x224
                img_array = np.array(img) / 255.0  # Normalize pixel values

                X.append(img_array)
        except Exception as e:
```

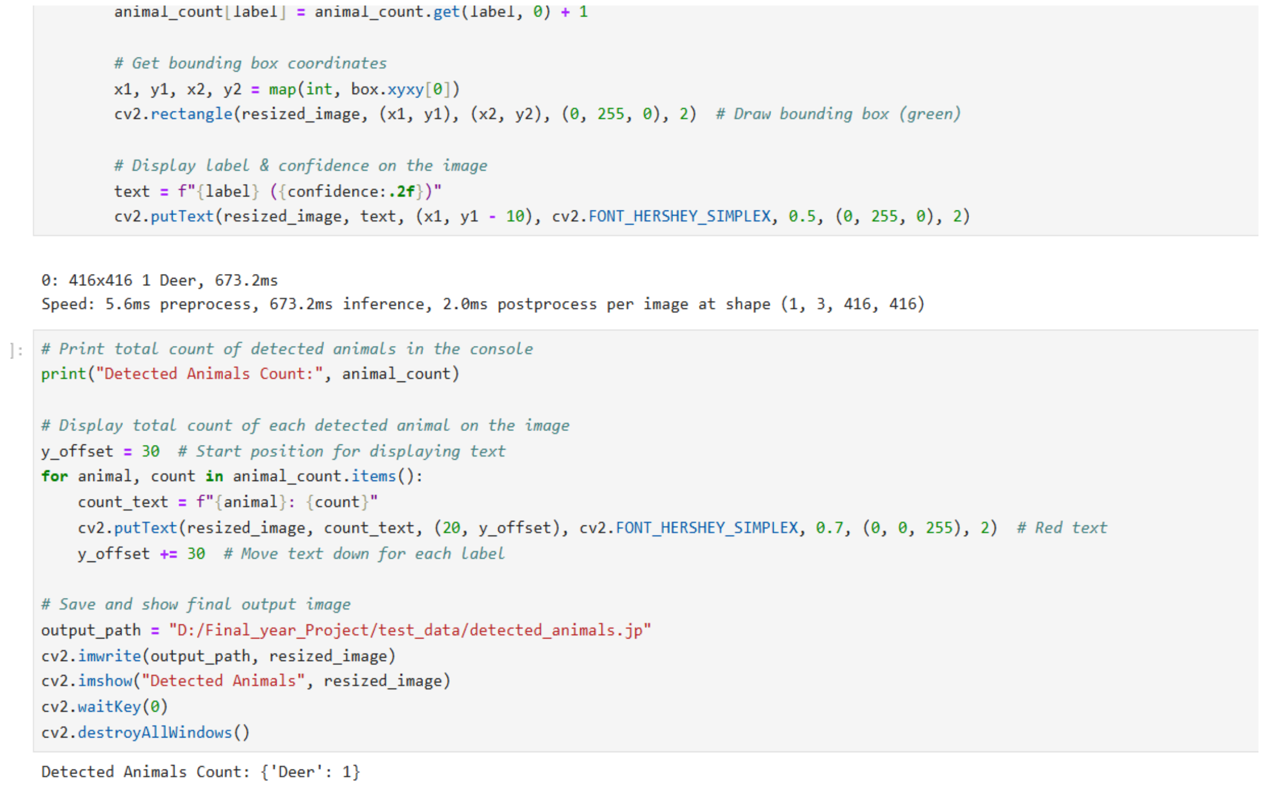
IMPLEMENTATION



Name	Date modified	Type	Size
test	15-02-2025 22:24	File folder	
train	15-02-2025 22:26	File folder	
valid	15-02-2025 22:26	File folder	


_annotations.coco
deer_grazing_1_1739603246_jpg.rf.2d32fc2acc84f6bba5d076329ca1...
deer_grazing_2_1739603246_jpg.rf.39f2eee757ce7c75625dac8a5af2...
deer_grazing_3_1739603840_jpg.rf.bdd3f44d8b230f13ca6238757e8...
deer_grazing_4_1739603247_jpg.rf.37336d5686ef5f681a06806e5059...
deer_grazing_4_1739603840_jpg.rf.89c0fa2fa412836a18e73b20e317...
deer_grazing_5_1739603247_jpg.rf.873ccbb1b9c4bce263956cd17cf...
deer_grazing_5_1739603840_jpg.rf.e8c373c3ac032efc2c2690880bc1f...
deer_grazing_6_1739603248_jpg.rf.16bf950e8fea8e80f8cdd1557bf4...
deer_grazing_6_1739603840_jpg.rf.4ed27e1269f690a39ef491a76a5...
deer_grazing_7_1739603249_jpg.rf.32fcd058e9f6c1a3db502eb2ad5...
deer_grazing_7_1739603841_jpg.rf.b4d9fe50d1276657398410cd268...
deer_grazing_8_1739603250_jpg.rf.b8e425ded7327a96330742a19c3...
deer_grazing_8_1739603842_jpg.rf.f559bb9b1fd6a4a09545e13a69e...
deer_grazing_9_1739603252_jpg.rf.350c79d10b529e30e6389b862ff...
deer_grazing_9_1739603844_jpg.rf.98f2cfbcea205dc0aeef149ffa92e...
deer_grazing_18_1739603261_jpg.rf.2e66b7ba1880e47c22a9b1bd72...
deer_grazing_20_1739603263_jpg.rf.99c6f9ae823f51ebfbbbb1a0591...
deer_grazing_21_1739603263_jpg.rf.fba2bbec600a7c9a3963f19a07...
deer_grazing_22_1739603265_jpg.rf.e37c7bcc9b90662161dea6f9d9...
deer_grazing_23_1739603265_jpg.rf.3565583852e9cfcad96328381fe...
deer_grazing_25_1739603266_jpg.rf.1527061a46e3e8f4f344444c927...
deer_grazing_25_1739603854_jpg.rf.07a38513e0c1d8d44be0c2dbd...
deer_grazing_26_1739603268_jpg.rf.fefd518f574cf7021a0b9cfb1e9a...
deer_grazing_26_1739603854_jpg.rf.5ba4ba4a48a7ffd091cbcd60e1b...
deer_grazing_27_1739603270_jpg.rf.ec83f40be304540166aef91f42a...
deer_grazing_27_1739603855_jpg.rf.f90ccc55bf8789366b77c4c3d8f...
deer_grazing_28_1739603271_jpg.rf.a64a730d73cd77768289472e0d...
deer_grazing_28_1739603856_jpg.rf.956f75e8cd4f508c1a8a2da8ea4...
deer_grazing_29_1739603271_jpg.rf.de0fa67359e60b7aabec156c39...
deer_grazing_29_1739603858_jpg.rf.a894220bb9e7db66da00c62268...
deer_grazing_30_1739603859_jpg.rf.2f8db0585bd3fbab4ec0f0b175...
deer_grazing_31_1739603274_jpg.rf.3be40b2c4cdcd56543acbff2e22...
deer_grazing_32_1739603861_jpg.rf.1f140a8a1e83301048d377e960...
deer_grazing_34_1739603278_jpg.rf.e2d4c4499a3205867f7787e16f8...
deer_grazing_35_1739603865_jpg.rf.e6fb202589456a4f49d653a7c3...
deer_grazing_38_1739603866_jpg.rf.33743a2d2b85d
deer_grazing_39_1739603283_jpg.rf.295340768e7c1f
deer_grazing_40_1739603284_jpg.rf.7dcdeef3ca8a7
deer_grazing_41_1739603869_jpg.rf.b08da0fb0c968f
deer_grazing_42_1739603286_jpg.rf.a97c8b3d75b95
deer_grazing_42_1739603869_jpg.rf.f9ce7773317106
deer_grazing_45_1739603870_jpg.rf.75252192de3f7
deer_grazing_48_1739603293_jpg.rf.1f6d017330334
deer_grazing_48_1739603871_jpg.rf.52474be104539
deer_grazing_49_1739603293_jpg.rf.e331ea4ca8bd5
deer_grazing_49_1739603871_jpg.rf.0aecf45bdd3fe
deer_grazing_51_1739603295_jpg.rf.445f0ba8ffefacf

IMPLEMENTATION

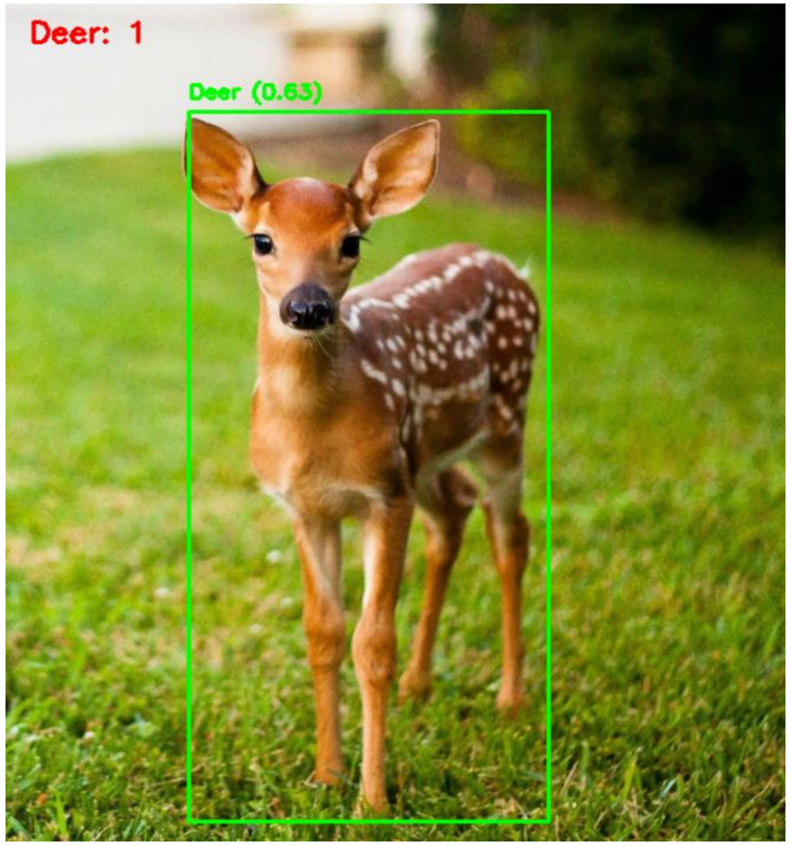


```
        animal_count[label] = animal_count.get(label, 0) + 1

        # Get bounding box coordinates
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        cv2.rectangle(resized_image, (x1, y1), (x2, y2), (0, 255, 0), 2)  # Draw bounding box (green)

        # Display label & confidence on the image
        text = f"{label} ({confidence:.2f})"
        cv2.putText(resized_image, text, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
```

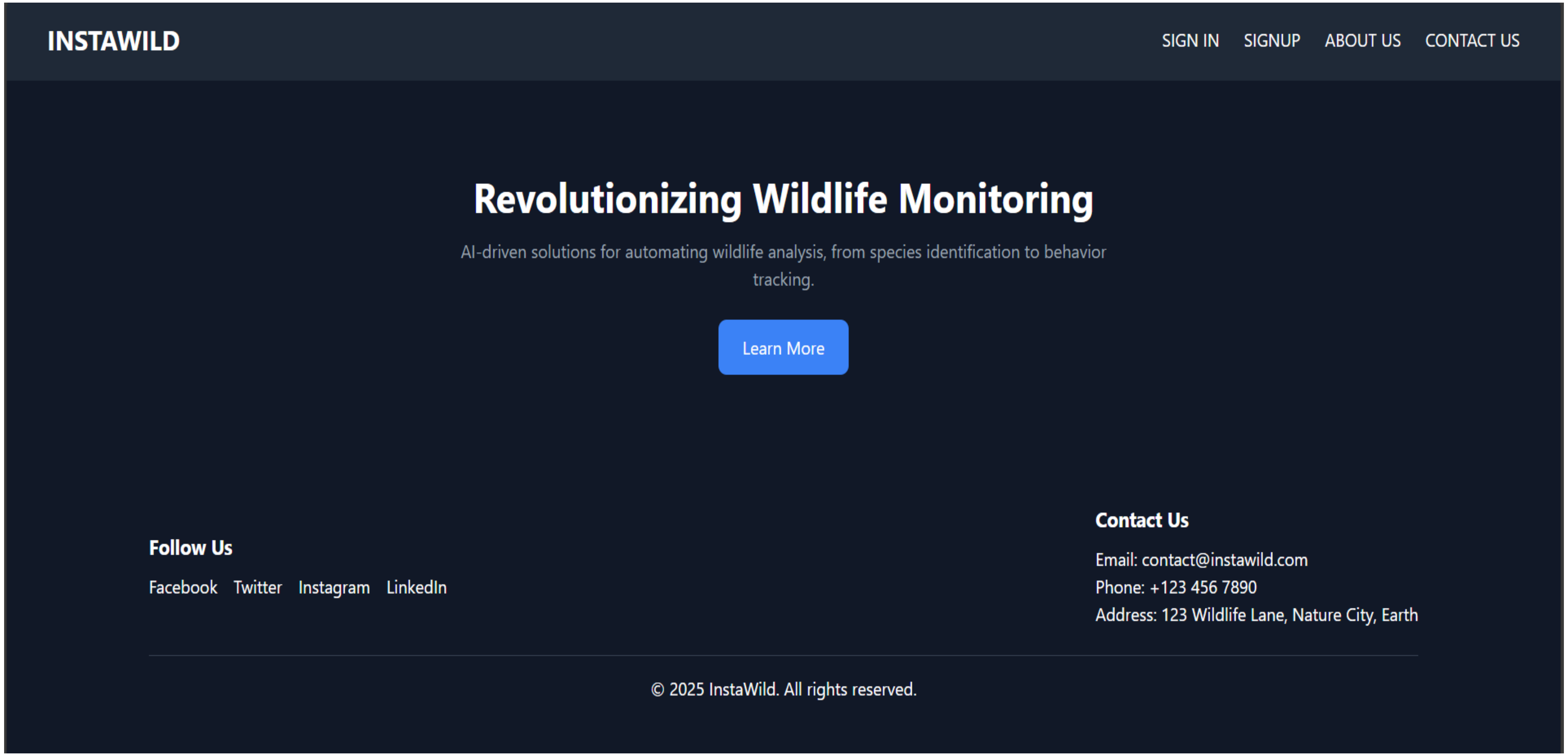
```
0: 416x416 1 Deer, 673.2ms
Speed: 5.6ms preprocess, 673.2ms inference, 2.0ms postprocess per image at shape (1, 3, 416, 416)
```

```
]: # Print total count of detected animals in the console
print("Detected Animals Count:", animal_count)

# Display total count of each detected animal on the image
y_offset = 30  # Start position for displaying text
for animal, count in animal_count.items():
    count_text = f"{animal}: {count}"
    cv2.putText(resized_image, count_text, (20, y_offset), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)  # Red text
    y_offset += 30  # Move text down for each label

# Save and show final output image
output_path = "D:/Final_year_Project/test_data/detected_animals.jp"
cv2.imwrite(output_path, resized_image)
cv2.imshow("Detected Animals", resized_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Detected Animals Count: {'Deer': 1}
```



IMPLEMENTATION


INSTAWILD

SIGN IN SIGNUP ABOUT US CONTACT US

Revolutionizing Wildlife Monitoring

AI-driven solutions for automating wildlife analysis, from species identification to behavior tracking.

Learn More

Follow Us

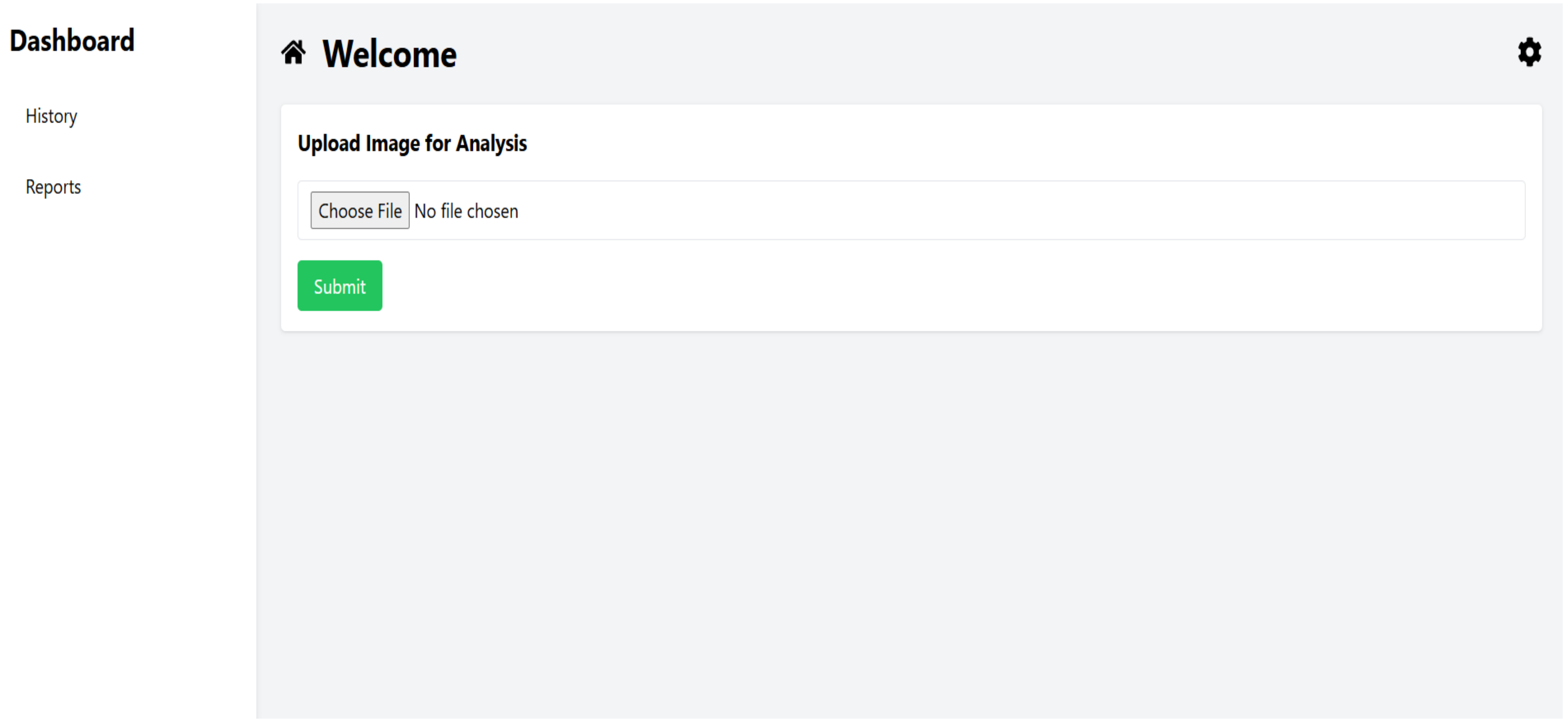
Facebook Twitter Instagram LinkedIn

Contact Us

Email: contact@instawild.com
Phone: +123 456 7890
Address: 123 Wildlife Lane, Nature City, Earth

© 2025 InstaWild. All rights reserved.

IMPLEMENTATION


Dashboard

History

Reports

Welcome

Upload Image for Analysis

Choose File No file chosen

Submit

REFERENCES

- **Websites:**

https://www.kaggle.com

https://www.tensorflow.org

https://github.com/ultralytics/yolov5

REFERENCES

➢ **Research Papers:**

1. " Mohammad Sadegh Norouzzadeh, Dan Morris, Sara Beery, Neel Joshi, Nebojsa Jojic, Jeff Clune, “A Deep Active Learning System for Species Identification and Counting in Camera Trap Images“14 October 2020, **https://doi.org/10.1111/2041-210X.13504**”
2. “Shuang Li, Haiyan Zhang, Fu Xu, Intelligent Detection Method for Wildlife Based on Deep Learning, December 7, 2023, *Sensors* .“
3. “Anne K. Schütz, Helen Louton, Mareike Fischer, Carolina Probst, Jörn M. Gethmann, Franz J. Conraths, Timo Homeier-Bachmann, Automated Detection and Counting of Wild Boar in Camera Trap Images, *Animals*, May 8, 2024 “

REFERENCES

4. “ Pandiselvi R, Jeyaprabhu J, Jerome Immanuel Jebaraj J, Muthupandi L, AI-Driven Wildlife Behavior Monitoring Using Computer Vision, International Journal for Multidisciplinary Research, Volume 6, Issue 5, September-October 2024”

5. “ Xin Tian, Bomeng Li, Xiaodong Cheng, Xiangyang Shi, Target Detection and Cow Standing Behavior Recognition Based on YOLOv5 Algorithm, 2022 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS)”

THANK

YOU