Creating Alert Messages Based on Wild Animal Activity Detection Using Hybrid Deep Neural Networks

Dr. Shah S. N.[1], Prof. S. A. Kokare[2], Mr. Vikas Mandlik[3], Ms. Swapnali Lambhate[4], Mr. Rushikesh Gaikwad[5]

[1]*HOD, Dept. of Computer Engineering, SSPM's Sharadchandra Pawar College of Engineering and Technology, Someshwarnagar, Baramati, India*

[2]*Project Guide, Dept. of Computer Engineering SSPM's Sharadchandra Pawar College of Engineering and Technology, Someshwarnagar, Baramati, India*

[3,4,5]*Dept. of Computer Engineering SSPM's Sharadchandra Pawar College of Engineering and Technology, Someshwarnagar, Baramati, India Roll No.: COA437*

***Abstract*—Wild animal movement near human settlements, agricultural fields, forest boundaries, highways, and railway areas has become a serious concern due to increasing human-wildlife conflict. In many rural and forest-border regions, sudden entry of wild animals such as tigers, elephants, leopards, bears, lions, and other species can cause crop damage, property loss, livestock attacks, road accidents, and life-threatening situations for humans. Traditional monitoring methods such as manual patrolling, camera traps, watch towers, and delayed reporting systems are not always reliable because they require continuous human observation and often fail to provide real-time warnings. This paper presents an intelligent system for creating alert messages based on wild animal activity detection using Hybrid Deep Neural Networks. The proposed system uses deep learning-based object detection to identify wild animals from live camera input or video streams. A YOLOv8-based detection model is used for fast and accurate animal detection, while OpenCV is used for real-time video frame processing. After detecting wild animal activity, the system generates alert messages, activates an alarm, updates the monitoring status, and stores alert history for future analysis. The system is implemented using a Flask-based web dashboard that displays live video streaming, detection status, detected ani-mal name, alert count, alarm indication, and previous detection records. The proposed system reduces manual monitoring effort and provides quick warnings to farmers, forest officers, wildlife departments, security teams, and local authorities. This system can be used in forest-border villages, agricultural fields, wildlife-sensitive roads, railway zones, and protected areas to improve safety and reduce human-wildlife conflict.**

***Index Terms*—Wild Animal Detection, Deep Learning, YOLOv8, Hybrid Deep Neural Networks, Computer Vision, OpenCV, Flask, Alert Message Generation, Real-Time Monitoring, Human-Wildlife Conflict.**

I. INTRODUCTION

Wildlife is an important part of the ecosystem and plays a major role in maintaining ecological balance. However, the increasing interaction between humans and wild animals has created serious safety and environmental challenges. Due to deforestation, urban expansion, climate change, food scarcity, and reduction of natural habitats, wild animals often move towards villages, farms, roads, and human settlements. This movement can lead to crop damage, property loss, road accidents, livestock attacks, and sometimes serious injury or death.

In many forest-border areas, people still depend on traditional methods to identify wild animal movement. These methods include manual patrolling, physical barriers, watch towers, camera traps, and reports from local residents. Although these methods are useful, they are not sufficient for real-time detection and quick alert generation. Manual monitoring requires continuous human effort and may fail during night time, poor weather conditions, or in remote locations. Camera traps are also limited because they usually capture images for later analysis and may not provide instant warnings.

Human-wildlife conflict is not only dangerous for humans but also harmful to animals. When wild

animals enter hu-man settlements, they are often chased, injured, trapped, or killed due to fear and lack of proper response mechanisms. Therefore, early detection and timely alert generation are important for protecting both human life and wildlife. A real-time intelligent monitoring system can help authorities take preventive action before the situation becomes dangerous.

With the advancement of Artificial Intelligence, Deep Learning, and Computer Vision, real-time object detection has become more accurate and efficient. Deep Neural Networks are capable of learning visual patterns from images and videos. Object detection models such as YOLO, Faster R-CNN, and SSD can identify and locate objects in real time. Among these, YOLO-based models are widely used because of their high speed and good accuracy. YOLOv8 is one of the latest and efficient object detection models suitable for real-time monitoring applications.

The proposed project, "Creating Alert Messages Based on Wild Animal Activity Detection Using Hybrid Deep Neural Networks," focuses on detecting wild animal activity from live video streams and generating alert messages automatically. The system uses YOLOv8 for object detection, OpenCV for image and video processing, and Flask for web-based monitoring. When a wild animal is detected, the system displays danger status, activates an alarm sound, generates an alert message, and stores the detection record.

The main goal of this system is to provide early warning so that users can take preventive action before a harmful incident occurs. The system is useful for farmers, forest officers, wildlife departments, security teams, and local communities living near forest areas. By using real-time detection and alert generation, the proposed system helps reduce the risk of human-wildlife conflict and improves wildlife monitoring efficiency.

The rest of this paper is organized as follows: Section II presents the problem statement and objectives. Section III explains the literature survey. Section IV describes the proposed system architecture. Section V presents the method-ology. Section VI explains system requirements. Section VII discusses implementation details. Section VIII presents result analysis and testing. Section IX explains limitations and future scope, and Section X concludes the paper.

II. PROBLEM STATEMENT AND OBJECTIVES

A. Problem Statement

Wild animal activity near human-populated areas is difficult to monitor continuously using manual methods. In many cases, people receive information about animal movement only after damage has already occurred. Existing systems such as camera traps and manual patrolling are limited because they are slow, costly, and dependent on human observation. There is a need for an automated system that can detect wild animals in real time and generate alert messages instantly.

The problem addressed in this paper is to develop an intelligent wild animal detection and alert generation system that can identify wild animals from live video streams and notify users immediately through visual alerts, audio alarms, and stored alert records.

B. Objectives

The main objectives of the proposed system are as follows:

- To design a real-time wild animal detection system using deep learning.
- To use YOLOv8 for accurate and fast detection of wild animals from video frames.
- To generate alert messages when wild animal activity is detected.
- To activate an alarm sound for immediate warning.
- To display live monitoring status through a Flask-based web dashboard.
- To store alert history with detected animal name, times-tamp, and status.
- To reduce manual monitoring effort and improve safety in wildlife-sensitive areas.

III. LITERATURE SURVEY

Wild animal monitoring has been an important area of research in wildlife conservation, forest security, and human safety. Earlier systems mainly depended on manual tracking, camera traps, radio collars, and field surveys. These methods helped researchers understand animal behavior and movement patterns, but they were not suitable for real-time alert generation.

Camera trap systems are commonly used to capture images of animals in forest areas. These systems are useful for wildlife observation, but most of them store

images for later analysis. As a result, there is a delay between animal detection and human response. In critical situations, delayed information may increase the risk of accidents or damage.

Traditional image processing techniques used features such as color, shape, edges, texture, and motion to identify animals. However, these methods are less reliable in complex natural environments. Factors such as poor lighting, background noise, animal posture, distance from camera, and weather conditions can reduce detection accuracy.

Deep Learning has improved the performance of image classification and object detection systems. Convolutional Neural Networks can automatically extract important features from images without manual feature engineering. CNN-based models have been used for animal classification, species recognition, and wildlife monitoring. However, classification models usually identify only whether an animal is present or not, and may not provide exact location in the image.

Object detection models solve this limitation by identifying both the class and location of objects. Models such as Faster R-CNN, SSD, and YOLO are widely used for object detection tasks. Faster R-CNN provides good accuracy but may require more processing time. SSD provides faster detection but may be less accurate in some complex cases. YOLO is widely used for real-time detection because it processes the image in a single pass and provides fast output.

YOLOv8 provides improved accuracy, speed, and flexibility compared to earlier YOLO versions. It can detect multiple objects in a frame and provide bounding boxes, class labels, and confidence scores. This makes it suitable for wild animal detection from live video streams. When YOLOv8 is integrated with OpenCV, it can process continuous video frames and detect animals in real time.

Several researchers have also worked on alert generation systems for surveillance, security, and disaster management. These systems detect unusual events and notify users through alarms, messages, or dashboards. In wildlife monitoring, alert generation is very important because early alerts allow people to stay safe and inform authorities quickly.

Existing wildlife detection systems still face challenges such as false detection, lack of real-time alerting, limited datasets, poor performance in low-light conditions, and absence of user-friendly dashboards. Some systems only focus on detection and do not provide complete alert history or monitoring interfaces. The proposed system addresses these limitations by combining deep learning-based detection, real-time video streaming, alert message generation, alarm indication, and alert history storage.

IV. PROPOSED SYSTEM ARCHITECTURE

The proposed system architecture is designed to detect wild animal activity in real time and generate alert messages immediately. The architecture follows a modular structure so that each part of the system performs a specific task. The major modules include Input Layer, Preprocessing Layer, Detection Layer, Alert Generation Layer, Presentation Layer, and Data Storage Layer.

A. Input Layer

The Input Layer captures real-time visual data from a webcam, CCTV camera, mobile camera, or video input source. The captured video stream is continuously processed frame by frame. This layer is important because the overall accuracy of the system depends on the quality and clarity of input data. The camera is placed in a monitoring area such as a forest border, agricultural field, village boundary, highway, or wildlife-sensitive zone.

B. Preprocessing Layer

The Preprocessing Layer prepares the captured frames be-fore sending them to the detection model. In this stage, video frames are resized, cleaned, and converted into a suitable for-mat for model processing. OpenCV is used for reading frames, resizing images, converting color formats, and handling the video stream. Preprocessing helps reduce noise and improves the overall detection performance of the system.

C. Detection Layer

The Detection Layer is the core part of the proposed system. It uses a YOLOv8-based deep learning model to detect wild animals from the processed frames. The model identifies the animal class, confidence score, and bounding box location. Only wild animal classes such as tiger, lion, elephant, leopard, bear, zebra, giraffe, and similar animals are considered for alert

generation. Other unwanted classes such as humans or domestic animals are ignored to reduce false alerts.

D. Alert Generation Layer

The Alert Generation Layer processes the detection output. If a wild animal is detected with a valid confidence score, the system generates an alert message. The alert includes the detected animal name, detection time, danger status, and recommended safety action. An alarm sound is also activated to notify the user immediately. If no animal is detected for a specific time period, the alert status is automatically reset.

E. Presentation Layer

The Presentation Layer provides a web-based dashboard using Flask. The dashboard displays the live video stream, current detection status, detected animal name, alert message, alarm status, total detections, active alerts, and previous alert history. The user can monitor the system through a browser and take quick action when an alert is generated.

F. Data Storage Layer

The Data Storage Layer stores alert history in a JSON file or database. Each record contains details such as detected animal name, detection time, alert message, and alert status. This stored data helps in reviewing previous detections and analyzing wild animal movement patterns over time.

Overall, the proposed architecture provides a real-time, scalable, and efficient solution for wild animal activity detection and alert message generation. It reduces manual monitoring effort and helps users take preventive action before any harmful incident occurs.

V. METHODOLOGY

The methodology of the proposed system is divided into several stages, including data collection, data preprocessing, model selection, real-time detection, alert message generation, dashboard display, and system evaluation.

A. Data Collection

The first stage involves collecting images and videos of different wild animals. The dataset includes animal classes such as tiger, lion, elephant, leopard, bear, zebra, giraffe, and other wildlife species. Images may be collected from public datasets, wildlife camera sources, online repositories, or custom captured data. The dataset should contain different lighting conditions, backgrounds, camera angles, animal poses, and distances to improve detection performance.

B. Data Preprocessing

Data preprocessing is performed to improve the quality of training and detection. Images are resized according to the model input size. Duplicate, blurred, or incorrect images are removed from the dataset. Annotation is performed by drawing bounding boxes around animals and assigning correct class labels. Data augmentation techniques such as rotation, flipping, brightness adjustment, scaling, and cropping can be used to improve model generalization.

C. Model Selection

YOLOv8 is selected as the main object detection model because it provides fast and accurate real-time detection. YOLOv8 is suitable for live monitoring because it can process video frames with low delay. A pre-trained YOLOv8 model can be used initially, and later it can be trained or fine-tuned using a custom wild animal dataset for improved accuracy.

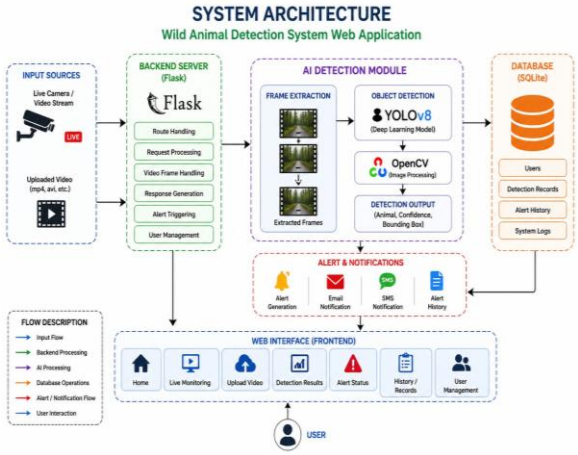

Fig. 1. Proposed System Architecture

D. Real-Time Detection

During real-time monitoring, the camera continuously captures video frames. Each frame is passed to the YOLOv8 detection model. The model analyzes the frame and returns the detected class name, confidence score, and bounding box coordinates. If the detected class belongs to the wild animal category, the system highlights the animal with a bounding box and displays the detected animal name.

E. Alert Message Generation

When wild animal activity is detected, the system creates an alert message automatically. The alert message includes important information such as detected animal name, detection time, danger status, and recommended safety action. For example, if a tiger is detected, the system may generate a message such as: "Danger Alert: Tiger detected near the monitoring area. Please take immediate safety action."

F. Alarm and Notification

The system activates an alarm sound when a wild animal is detected. The alarm continues while the animal is present in the video frame. If the animal disappears from the frame, the alarm stops after a short delay. This prevents sudden false stopping of alerts and gives users enough time to respond. In future versions, the system can be integrated with WhatsApp, SMS, email, or mobile app notifications.

G. Dashboard and Alert History

The Flask dashboard shows the live video stream, current safety status, detected animal name, total detections, active alerts, handled alerts, and alert history. The alert history stores previous detection records with timestamps. This helps users review animal activity and maintain a record of past incidents.

TABLE I Hardware Requirements

Component	Minimum Requirement
Processor	Intel Core i5 or above
RAM	8 GB or above
Storage	10 GB free space
Camera	Webcam / CCTV Camera
GPU	Optional for faster processing
Network	Required for remote alert support

H. System Evaluation

The system is evaluated based on detection accuracy, response time, alert correctness, and dashboard performance. Testing is performed using sample images, recorded videos, and live webcam input. The main goal is to ensure that the system detects wild animals correctly and generates alerts in real time with minimum delay.

VI. SYSTEM REQUIREMENTS

The proposed system requires both hardware and software components to perform real-time detection and alert generation efficiently.

A. Hardware Requirements

B. Software Requirements

VII. IMPLEMENTATION DETAILS

The proposed system is implemented using Python-based technologies. The backend uses Flask to handle routing, video streaming, and alert status management. OpenCV captures frames from the camera and passes them to the YOLOv8 model for detection. The YOLOv8 model processes each frame and returns detection results.

TABLE II Software Requirements

Software	Purpose
Python	Main programming language
YOLOv8	Object detection model
OpenCV	Image and video processing
Flask	Web application framework
HTML, CSS, JavaScript	Frontend dashboard design
JSON / Database	Alert history storage
Overleaf / LaTeX	Documentation and report writing

The system checks whether the detected object belongs to the predefined wild animal class list. If the detected class matches the wild animal category, the system sets the alert status to danger. The dashboard then displays the detected animal name and activates the alarm. If no animal is detected for a fixed time interval, the system automatically resets the status to safe.

The frontend dashboard contains different sections such as live camera feed, safety status, detection information, alert count, and previous alert records. The dashboard helps users understand the current situation clearly. It also improves us-ability because users do not need to check terminal output or raw model logs.

A. Working Algorithm

The working algorithm of the proposed system is as follows:

1) Start the Flask web server.
2) Initialize the camera or video input source.
3) Load the YOLOv8 detection model.
4) Capture video frames continuously.
5) Preprocess each frame using OpenCV.

6) Pass the frame to the YOLOv8 model.
7) Check detected class names and confidence scores.
8) If a wild animal is detected, generate an alert message.
9) Activate alarm and update dashboard status.
10) Store alert details in alert history.
11) If no animal is detected for a specific time, reset status to safe.
12) Continue monitoring until the system is stopped.

B. Alert Message Format

The alert message generated by the system contains the following information:

- Alert type
- Detected animal name
- Detection time
- Danger status
- Recommended action
- Alert record status

A sample alert message is shown below:

Danger Alert: Tiger detected near the monitoring area at 07:45 PM. Please stay away from the area and inform the concerned authority immediately.

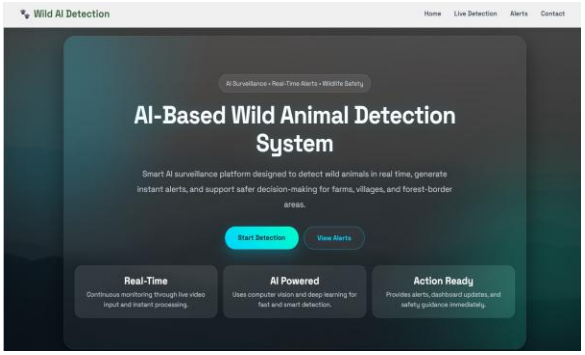

Fig. 2. Real-Time Wild Animal Detection Home page

VIII. RESULT AND DISCUSSION

The proposed system was tested using live camera input and sample wildlife images. The YOLOv8 model successfully detected wild animals from video frames and displayed bounding boxes around detected animals. When a wild animal was detected, the system changed the monitoring status from safe to danger and generated an alert message.

The Flask dashboard displayed the live video stream along with the current detection status. The alarm module worked when animal activity was detected and stopped automatically when the animal was no longer visible. The alert history successfully stored previous detection records with animal name, detection time, and alert status.

The system showed effective performance in real-time monitoring conditions. It was able to process video frames continuously and generate alerts with minimum delay. The system also reduced manual monitoring effort by automatically identifying wild animal activity and informing users through visual and audio alerts.

A. Detection Output

The detection output includes the following details:

- Detected animal name
- Bounding box around the animal
- Confidence score
- Detection time
- Alert status
- Recommended safety message

B. Alert Output

The system generates alert messages such as:

Danger Alert: Wild animal detected in the monitoring area. Please stay away from the area and inform the concerned authority immediately.

C. Result Images

The following figures show the practical output screens of the proposed wild animal detection and alert generation system.

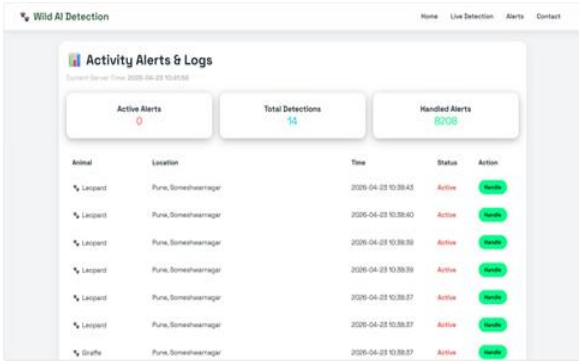

Fig. 3. Alert Message Generation Output Dashboard page

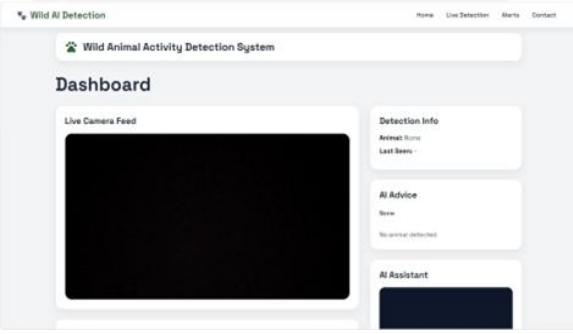

Fig. 4. Flask Dashboard Result Output

TABLE III Sample Test Cases

Test ID	Input	Expected Output	Status
TC01	Open dashboard	Dashboard loads successfully	Pass
TC02	Camera input	Live video stream displayed	Pass
TC03	Tiger image/video	Tiger detected and alert generated	Pass
TC04	No animal frame	Safe status displayed	Pass
TC05	Wild animal disappears	Alarm stops after delay	Pass
TC06	Multiple detections	Alert count updated	Pass
TC07	Detection event	Alert history stored	Pass

D. Sample Testing Table

E. Performance Observation

The system performed effectively for real-time animal detection and alert generation. Detection accuracy depends on image quality, camera angle, lighting condition, animal visibility, and model training. The system works better when the animal is clearly visible in the frame. False alerts can be reduced by using a custom-trained dataset and confidence threshold filtering.

The response time of the system is suitable for basic real-time monitoring. However, performance may vary depending on hardware capability. Systems with GPU support can process frames faster than CPU-only systems. The proposed system can be further improved by optimizing frame size, using custom-trained models, and deploying the application on edge devices.

IX. ADVANTAGES AND APPLICATIONS

A. Advantages

The proposed system provides several advantages over traditional monitoring methods:

- Provides real-time wild animal detection.
- Reduces dependency on manual observation.
- Generates instant alert messages.
- Supports live monitoring through a web dashboard.
- Stores alert history for future analysis.
- Can be deployed in farms, forest borders, and wildlife-sensitive zones.
- Helps reduce risk to humans and animals.

B. Applications

The proposed system can be used in various real-world applications:

- Forest boundary monitoring.
- Agricultural field protection.
- Village safety near wildlife areas.
- Highway and railway animal movement detection.
- Wildlife conservation and research.
- Smart surveillance systems.
- Protected area monitoring.

X. LIMITATIONS AND FUTURE SCOPE

A. Limitations

Although the proposed system provides real-time detection and alert generation, it has some limitations. Detection ac-curacy depends on camera quality, lighting conditions, model training, and visibility of the animal. In poor lighting or heavy rain, the system may not detect animals accurately. If the animal is far from the camera or partially hidden, detection confidence may decrease.

Another limitation is that a pretrained model may not detect all wild animal species accurately. For better performance, a custom dataset containing local wildlife species should be used for training. Internet connectivity may also be required for remote notifications such as SMS, WhatsApp, or email alerts.

B. Future Scope

In the future, the system can be improved by adding the following features:

- Training a custom YOLOv8 model on a large wildlife dataset.

- Adding night vision and thermal camera support.
- Integrating WhatsApp, SMS, and email alerts.
- Deploying the system on Raspberry Pi or edge devices.
- Adding GPS location-based alert messages.
- Developing a mobile application for remote monitoring.
- Adding cloud storage for alert records and analytics.
- Improving detection accuracy using advanced deep learning models.

XI. CONCLUSION

This paper presented a real-time wild animal activity detection and alert message generation system using Hybrid Deep Neural Networks. The proposed system uses YOLOv8, OpenCV, and Flask to detect wild animals from live video streams and generate instant alerts. The system helps reduce human-wildlife conflict by providing early warnings when wild animals enter monitored areas.

The system provides live monitoring, visual detection, alarm indication, alert message generation, and alert history storage. It is useful for forest departments, farmers, wildlife researchers, security teams, and local authorities. Compared to manual monitoring, the proposed system is faster, more reliable, and cost-effective.

The proposed system shows that deep learning and computer vision can be effectively used for wildlife monitoring and safety management. With further improvements such as custom dataset training, night vision support, mobile alerts, and edge deployment, the system can become a practical solution for real-world human-wildlife conflict reduction.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016.

[2] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," 2023. [Online]. Available: Ultralytics GitHub Repository

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[4] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[6] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016.

[10] R. Szeliski, *Computer Vision: Algorithms and Applications*. London, U.K.: Springer, 2010.

[11] A. Rosebrock, *Deep Learning for Computer Vision with Python*. PyImageSearch, 2017.

[12] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2001.

[13] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2005.

[14] M. Everingham *et al.*, "The Pascal Visual Object Classes Challenge: A Retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, 2015.

[15] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014.

[16] D. Norouzzadeh *et al.*, "Automatically Identifying, Counting, and Describing Wild Animals in Camera-Trap Images with Deep Learning," *Proc. Natl. Acad. Sci.*, vol. 115, no. 25, pp. E5716–E5725, 2018.

[17] M. A. Tabak *et al.*, "Machine Learning to Classify Animal Species in Camera Trap Images,"

Methods Ecol. Evol., vol. 10, no. 4, pp. 585–590, 2019.

[18] S. Schneider, G. W. Taylor, and S. Kremer, "Deep Learning Object Detection Methods for Ecological Camera Trap Data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2018.

[19] T. Nguyen *et al.*, "Animal Recognition and Identification with Deep Convolutional Neural Networks for Automated Wildlife Monitoring," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, 2017.

[20] M. Willi *et al.*, "Identifying Animal Species in Camera Trap Images using Deep Learning and Citizen Science," *Methods Ecol. Evol.*, vol. 10, no. 1, pp. 80–91, 2019.

[21] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, 2000.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[24] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.

[25] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019.