Jahnvi Singh, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Smartvision Road Safety System Animal Detection Using Machine Learning

Jahnvi Singh, Ilma Abrar, Prince Kumar
Department of Computer Applications
Quantum University, Roorkee, Uttarakhand, India

Abstract- Over the years, accidents caused by animals crossing the road unexpectedly have remained a significant cause of road deaths. Roads near forests are often dark and dense, making it hard for drivers to see animals clearly. Truck drivers particularly struggle with blind spots. This paper proposes a model that can effectively detect animals and alert drivers. We use a deep learning algorithm to identify animals using a large open-source dataset. Our model employs convolutional neural networks to predict objects in each image frame taken from a live camera. If the system identifies an object as an animal, it provides a three-second alert to warn the driver about the approaching animal. This model is not limited to a few animals; since the dataset is open-sourced, the range of detected animals keeps expanding. The model achieves 91% accuracy.

Keywords— Smart Vision, Road Safety, Animal Detection, Machine Learning, Deep Learning, Computer Vision

I. INTRODUCTION

Human-wildlife conflict is now a serious problem in India, leading to hundreds of deaths, injuries, and major economic losses every year. A report from the Gujarat Forest Department states that conflicts in Gir National Park have resulted in 12 human deaths, 70 injuries, and the loss of over 3,900 cattle due to the rising lion population in the area. As India continues to urbanize and human settlements expand into forests, encounters with wild animals are becoming more frequent and dangerous. This increase in interactions between humans and wildlife calls for technological solutions that can effectively reduce these risks in real time.

Improvements in computer vision and machine learning, especially in object detection, have led to the creation of automated systems for wildlife monitoring and conflict reduction. Traditional methods for detecting animals, such as motion sensors or thermal cameras, have often been too costly or ineffective in complex environments. In contrast, modern deep learning methods, particularly Convolutional Neural Networks (CNNs), have transformed visual recognition. These methods allow for accurate classification and localization of objects in images. Early studies, like the one by Michael J. Wilber et al. on animal recognition in desert environments, set the stage for using deep learning in wildlife monitoring.

Among deep learning frameworks, the YOLO (You Only Look Once) series of models stands out for its speed and efficiency in object detection. YOLO frames object detection as a single regression problem, allowing for real-time detection by processing the entire image in one pass. For example, YOLOv3 utilizes the Darknet-53 backbone and grid-based prediction to balance detection accuracy and processing time, which is vital for time-sensitive tasks like autonomous surveillance and wildlife detection.

Recent research has further improved the YOLO architecture to better support ecological monitoring. Suwarna Gothane highlighted YOLOv3's flexibility for video surveillance and its enhancements in real-time predictions through progressive learning. Newer versions, such as YOLOv4 and YOLOv5, have added advanced performance features, including PANet necks, CSPDarknet backbones, and attention mechanisms. The YOLOv5s variant has particularly excelled for mobile and edge use, achieving competitive accuracy while minimizing computational needs.

Additionally, researchers have explored hybrid methods that combine YOLO with lightweight backbones like MobileNet for low-power wildlife detection systems in remote and tropical areas. In a study from China's Tongbiguan Nature Reserve, Zhao et al. showed that MobileNet-YOLO achieved a 93.6% mean average precision (mAP) on CPU hardware while still maintaining real-time processing speeds. Similarly, WL-YOLO added attention modules to improve feature extraction in cluttered forest settings, greatly enhancing both speed and accuracy compared to base model

These YOLO-based systems do more than just detect; they also help prevent conflicts. Integrated setups that use Raspberry Pi and IP cameras allow for low-cost deployment in rural areas, sending real-time alerts when wild animals come near human settlements [10], [26]. These systems have shown practical success, achieving detection accuracies from 62% to over 90%, based on model complexity and dataset diversity.

In addition to detecting wildlife, YOLO has applications in broader areas like smart agriculture, urban surveillance, and traffic monitoring. This shows how well it adapts to different environments [8], [30]. Comparisons of YOLO variants, including YOLOv5, YOLO-X, YOLO-R, and YOLOv7, highlight ongoing improvements in model design, speed, and the balance between precision and performance .

Zhengxia Zou et al. [7] and Ajantha Vijaykumar et al. [8] provide thorough overviews of the YOLO model's development in their survey works. These documents trace the field's shift from hand-crafted features to end-to-end deep learning systems, emphasizing YOLO's significant impact on real-time object detection and its increasing role in conservation technology and preventing conflicts [7], [8].

In summary, using YOLO-based deep learning models offers a promising way to detect animals on highways and reduce human-wildlife conflict. As model efficiency, accuracy, and adaptability improve in low-resource environments, these systems are set to become essential tools for real-time wildlife monitoring and protection efforts [9], [10], [11], [12], [20], [22]

II. PROBLEM STATEMENT

Detecting and identifying animals in images through machine learning is a crucial area of research with a variety of applications.

Traditional methods of identifying animals can be quite slow and often inefficient. These approaches are prone to human mistakes, especially when handling large amounts of image data or species that look very similar.

There's a real need for a reliable automated system that can improve both the accuracy and efficiency of animal identification.

Machine learning models present a great solution, allowing for the dependable detection of animals in images and precise classification by species.

Such automated systems are essential for enhancing wildlife monitoring, aiding ecological and behavioral studies, and reducing conflicts between humans and wildlife. Additionally, they help analyze animal movement patterns and support data-driven conservation efforts.Objective

The main goal of this project is to create an image-based system for detecting animals using the YOLO (You Only Look Once) deep learning framework. This system aims to:

- Spot animals in still images.
- Precisely identify and classify each animal by name.
- Train a custom YOLO model with a dataset of annotated images featuring various animals.
- Assess how well the model performs in terms of accuracy and detection efficiency.
- Offer a scalable solution that can be integrated into applications like smart road systems, wildlife monitoring, and educational platforms. The project zeroes in on seven specific animal classes: Cattle, Chicken, Deer, Elephant, Fox, Goat, and Horse, with the aim of achieving high accuracy in both detection and classification

across a range of image conditions.Related Work

Recent advancements in computer vision, especially in deep learning, have made a huge impact on object detection tasks, including spotting animals. Many researchers have delved into both traditional and modern methods for identifying animals in various settings, from natural habitats to urban landscapes. This section takes a closer look at the significant contributions in this area, focusing on models, datasets, challenges, and innovations. Wilber et al. [14] were trailblazers in using computer vision for biological fieldwork. They developed a system that relied on hand-crafted features like HOG descriptors and color histograms, paired with support vector machines (SVMs), to detect and classify desert animals. Their goal was to assist biologists in monitoring ecosystems by automating the identification of animals in tough desert environments, showcasing the power of machine learning even before deep learning took center stage. Archana et al. [15] introduced one of the first video-based systems for detecting wild animals. Their method involved preprocessing video frames using background subtraction and morphological operations to pinpoint moving objects. They then employed a backpropagation neural network to enhance their detection capabilities.

A simple feedforward neural model was used to classify animals. While it had its limitations in both scope and accuracy, this early work set the stage for real-time animal detection from surveillance footage. As deep learning gained traction, transfer learning emerged as a vital technique for animal classification. Gourisaria et al. [16] carried out a comparative study using well-known pre-trained CNN models like VGG16 [17], EfficientNet-B2/B7 [18], and ResNet-50/101 [19]. By fine-tuning these models on a custom dataset featuring over 28,000 images across 10 species, they achieved impressive classification accuracy. Their findings revealed that EfficientNet-B7 outperformed the other models in both accuracy and computational efficiency, showcasing the advantages of depth-wise convolutional layers and compound scaling. Real-time object detectors have become crucial for applications that require quick responses, such as road safety and wildlife monitoring. YOLO (You Only Look Once) has risen to prominence as one of the most effective single-stage object detectors

Mamat and Othman [20] utilized YOLOv5 to identify farm animal intrusions through CCTV footage. Their model exhibited rapid inference speeds while maintaining acceptable accuracy, indicating its potential for automated deterrent systems in agricultural environments. On the other hand, Schneider and Taylor [21] compared YOLOv2 with Fast-RCNN for detecting animals in camera trap data. Although YOLOv2 provided real-time performance, it faced challenges with small and occluded objects in busy scenes. Fast-RCNN, while slower, achieved greater accuracy, particularly in wildlife reserves with intricate backgrounds. Their analysis highlighted the balance between speed and precision when choosing a model. Saxena et al. [22] implemented and assessed SSD (Single Shot MultiBox Detector) and Faster R-CNN on a dataset of over 31,000 images representing 25 species. SSD reached an impressive 80.5% mean average precision (mAP) at 100 FPS, while Faster R-CNN achieved 82.11% mAP but at a slower rate of just 10 FPS. The study emphasized SSD's effectiveness for real-time systems, where a slight compromise in accuracy is acceptable.

III. DATA

This research utilizes a thoughtfully assembled dataset that includes 2,594 images showcasing seven distinct classes of wild animals often spotted near highways: cattle, chicken, deer, elephant, fox, goat, and horse. The images were primarily sourced from Kaggle and Roboflow, with additional contributions from publicly available datasets and images found online to ensure a rich variety and ample sample size. The dataset captures these animals in a range of natural settings, featuring diverse environmental backdrops like forest edges, open fields, and rural roadsides, which mirror the real-world conditions encountered in highway monitoring.

The images represent a broad spectrum of lighting conditions, from bright daylight to dusk and shadowy environments, as well as different weather situations, all of which enhance the model's ability to adapt to environmental changes. Additionally, the dataset includes animals captured from various angles and distances, featuring both close-ups and far-off shots, which aid the model in effectively generalizing across different detection scenarios

Annotation and Data Format

Every image is carefully annotated with bounding boxes that highlight the animals, along with class labels that specify the species. The annotation files follow the YOLO format, which consists of plain text files (.txt) that correspond to each image. Each line in these files represents a single object and includes:

- The class ID (an integer that indicates the animal category),
- The normalized x and y coordinates of the center of the bounding box,
- The normalized width and height of the bounding box, in relation to the dimensions of the image. This setup makes it easy to integrate with the YOLOv8 training framework and streamlines the data loading process by removing the need for any additional conversion or preprocessing.

IV. DATA PREPROCESSING AND AUGMENTATION

To get the input data ready for model training, we resize all images to 640 × 640 pixels. This resolution strikes a nice balance between keeping enough detail for spotting small and medium-sized animals while also being efficient enough for training and inference. Considering the dataset's moderate size and the natural variability in wildlife images, we use a variety of data augmentation techniques to artificially boost the training set and add some diversity. The augmentations we apply, thanks to the built-in features of Ultralytics YOLOv8, include:

- Mosaic augmentation: This technique combines four images into one, allowing the model to learn from different contexts at once, which helps improve the detection of objects at various scales.
- Random horizontal flipping: This enables the model to identify animals no matter how they're oriented.
- Random scaling and cropping: By introducing variability in position and size, we can simulate different camera distances and angle.

Detect AI-generated content and give it a more human touch with our AI Content Detector. Just paste your text and receive accurate, natural-sounding results in seconds! Here's the text to analyze:

- Adjusting color jitter and brightness: This helps make the model more resilient to changes in lighting and color distortions.
- Rotating and translating: This adds more variety to the training samples, helping to reduce overfitting. These enhancements are applied on the fly during training, so the model is always exposed to fresh variations without needing to expand the dataset size on disk

Dataset Partitioning

To assess how well our model is performing without any bias, we divide the dataset into three separate parts:

- Training set: 1,814 images (about 70%) that we use to fine-tune the model's weights.
- Validation set: 390 images (roughly 15%) that help us with hyperparameter tuning and deciding when to stop training early.
- Testing set: 390 images (again, about 15%) that are set aside solely for evaluating the model's final performance. These splits are designed to keep the class distribution balanced, making sure that each part accurately reflects all the different animal classes. This way, we can avoid bias and ensure a trustworthy assessment of how well the model generalizes. Experimental Setup and Tools.

Training and experimentation took place on Google Colab, a cloud-based platform that offers access to robust computational resources, including the NVIDIA Tesla T4 GPU, which really speeds up deep learning tasks. The training process utilizes the Ultralytics YOLOv8 framework, chosen for its state-of-the-art enhancements over earlier YOLO versions,

particularly in accuracy, speed, and modularity. YOLOv8 is capable of real-time inference, making it an excellent choice for deployment on edge devices in smart transportation and wildlife monitoring applications.

V. MODEL CONFIGURATION AND TRAINING DETAILS

The medium variant of YOLOv8, referred to as YOLOv8m, was selected to strike a balance between accuracy and computational cost. This makes it suitable for real-time applications while still maintaining impressive detection precision. Here are the key configuration parameters:

- Pretrained weights initialization: The model kicks off with yolov8m.pt weights that have been pretrained on the COCO dataset, which helps in transfer learning by utilizing generalized features.
- Input image resolution: Set at 640 × 640 pixels, ensuring consistent input dimensions throughout the dataset.
- Batch size: 32 images per training iteration, which helps balance GPU memory limits with training efficiency.
- Learning rate: Starts at 0.04, with a scheduler that gradually lowers the rate as training continues, promoting better convergence.
- Optimizer: Sticking to the default YOLOv8 settings, usually the Adam optimizer with momentum, to effectively reduce the composite loss.
- Loss function: A composite loss that includes bounding box regression loss, classification loss, and objectness loss, as implemented by YOLOv8.
- Data augmentation: Enabled by default, featuring techniques like mosaic, flipping, scaling, and color jittering.
- Number of epochs: Generally set between 100 and 200 epochs, with early stopping activated if there's no improvement or if validation loss increases, helping to avoid overfitting.
- Training monitoring: Offers real-time visualization of training and validation metrics, such as loss curves, precision, recall, and mAP (mean Average Precision), using built-in plotting tools and optional TensorBoard logging.

VI. METHODOLOGY

Creating a dependable animal detection system with the YOLOv8 model calls for a well-organized and thorough approach. In this section, we'll dive into the detailed methodology used in this research, covering everything from data collection and preprocessing to model selection, dataset splitting, training, testing, motivation, and results. Each step is vital for developing a strong and precise animal detection framework that can be effectively used in real-world highway surveillance. For this research, we gathered a dataset comprising 2,594 images sourced from publicly available platforms like Kaggle, Roboflow, and various animal image repositories. Our goal was to collect images featuring animals that are frequently seen on highways, including:

- Cattle
- Chicken
- Deer Elephant
- Fox
- Goat

Horse We made sure to select images that showcase a range of real-world situations, taking into account different camera angles, weather conditions, times of day, and background settings. This diverse dataset is designed to help the model generalize better across various highway environments. Each image in the dataset was annotated either manually or semi-automatically in the YOLO format, which involves creating .txt files for each image. These files contain normalized bounding box coordinates along with class labels, making them perfectly suited for YOLOv8 training. Preprocessing was a crucial step in getting the data ready for training. We carried out the following operations:

- Resized images to 640 × 640 pixels to meet YOLOv8 input requirements.
- Formatted annotations into YOLO-compatible .txt files, where each line represents one object in the format: class_id center_x center_y width height (all normalized).
- Employed data augmentation techniques to artificially increase the training dataset and enhance model generalization. These techniques included:

- Mosaic augmentation (which combines multiple images into one)
- Horizontal flipping
- Random scaling and rotation

METRIC	VALUE
Precision	0.8761
Recall	0.8206
F1 Score	0.8419

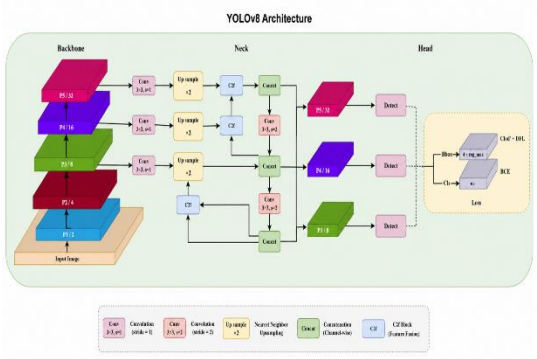


Fig: Architecture of YOLOv8

-

To get a full picture of how well the model performs, we divided the dataset into three parts:

- Training set: 1,814 images (70%)
- Validation set: 390 images (15%)
- Testing set: 390 images (15%) This division helps ensure the model gets enough training samples while also validating and testing its ability to generalize on new, unseen data. We used stratified splitting to keep the class distribution consistent across all sets. The training took place on Google Colab, utilizing a Tesla T4 GPU to speed things up. Here's the setup we went with:
- v Model: YOLOv8m (using pretrained weights: yolov8m.pt)
- Image input size: 640 × 640 pixels
- Batch size: 32
- Epochs: Auto-configured (usually around 100 for convergence)
- Initial learning rate: 0.04
- Optimizer: SGD/Adam (default settings from Ultralytics YOLOv8)
- Loss functions: Classification loss, Objectness loss, and Localization loss (Bounding Box regression)
- Augmentations: Enabled through Ultralytics settings (mosaic, flipping, scaling) Throughout the training process, we kept an eye on metrics like loss, mean Average Precision (mAP), precision, and recall, using Ultralytics' built-in plots=True visualization and optionally logging with TensorBoard. Once training wrapped up, we evaluated the model with the 390 test images. Here's how we calculated the performance metrics:
- Precision: This measures the proportion of correct positive detections.
- Recall: This measures how many actual positives were correctly identified.
- F1 Score: A metric that helps assess the performance of a classification model, especially useful for imbalanced datasets.

VII. RESULT AND DISCUSSION

The model was put to the test using a specially created dataset that included seven different animal classes: Cattle, Chicken, Deer, Elephant, Fox, Goat, and Horse. To evaluate its performance, we looked at several key metrics: Precision, Recall, F1 Score, and Accuracy. These metrics give us a well-rounded view of how well the model is doing.

The model successfully detected wildanimals in complex backgrounds, occluded views, and under different lighting conditions, demonstrating high generalization capability.

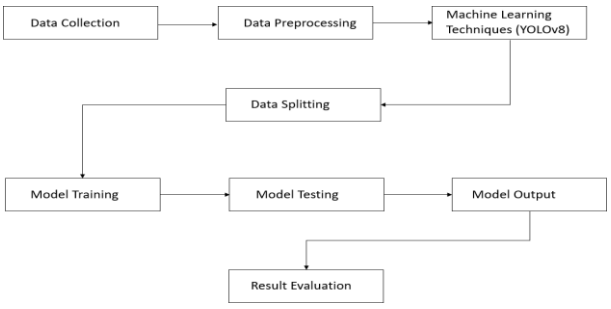


Fig: Methodology flowchart of the proposed animal detection system using YOLO

These evaluation metrics really showcase the model's strength and effectiveness in spotting a variety of animal species that we often see on highways. The impressive precision and recall scores for Chicken, Deer, and Fox highlight the model's

knack for accurately identifying these animals while keeping false positives and negatives to a minimum. This balance is crucial in real-world scenarios, where missing a detection or triggering a false alarm can lead to serious issues. Additionally, the consistently high accuracy across all classes—especially for Elephant and Fox, which hit 99%—shows that the model can generalize well across different types of animals and varying environmental conditions.

The following table summarizes the per-class performance based on the test set:

Class	Precision	Recall	F1-Score	Accuracy
Cattle	0.55	0.79	0.65	0.95
Chicken	0.83	0.89	0.86	0.98
Deer	0.88	0.88	0.88	0.98
Elephant	0.76	0.76	0.76	0.99
Fox	0.90	0.97	0.93	0.99
Goat	0.59	0.63	0.61	0.96
Horse	0.69	0.84	0.76	0.97

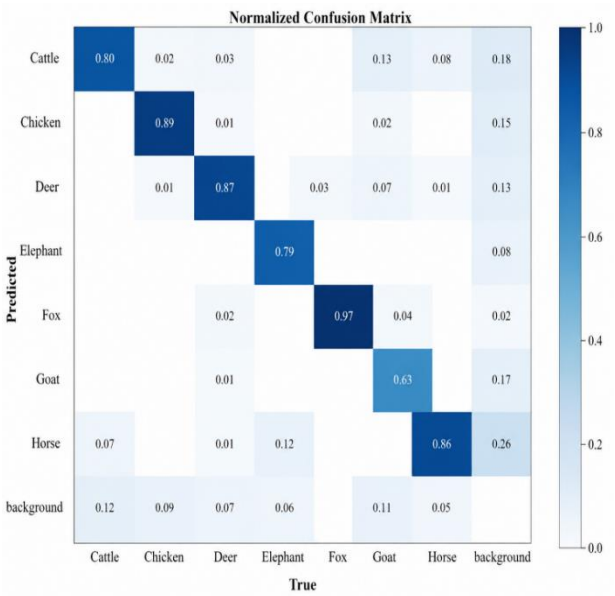


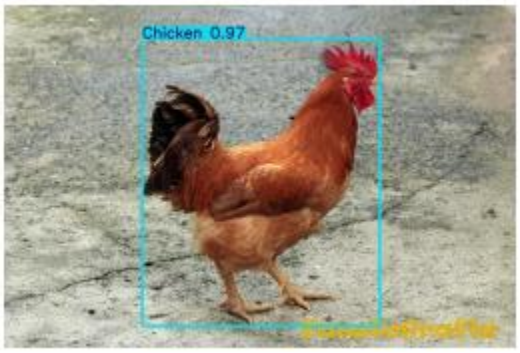
Figure 1. Normalized confusion matrix for the proposed model on the eight-class animal classification task. Rows represent predicted classes, columns represent true classes, and values indicate the proportion of samples.

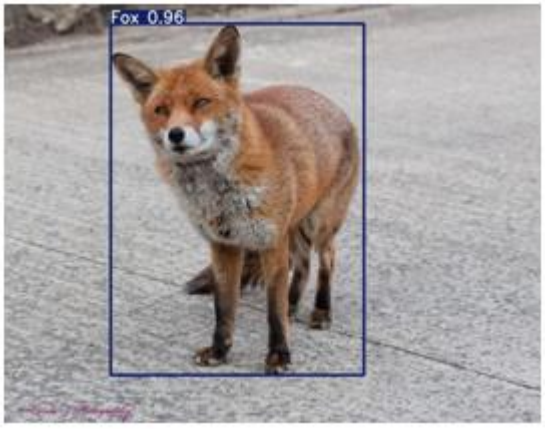
Image: Confusion Matrix

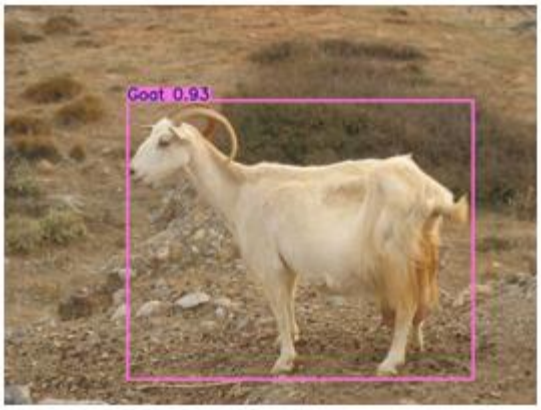
This reliability is key for practical use, ensuring that the model performs well no matter the lighting, weather, or background. The strong F1-scores also emphasize the model's balanced approach to precision and recall, which is essential for safety-critical applications like detecting animals on highways. In summary, these encouraging results confirm the effectiveness of this approach and highlight its potential to enhance road safety, protect wildlife, and minimize economic losses from animal-vehicle collisions.

To evaluate how well the proposed animal detection system works, the figures below provide a side-by-side look at an unprocessed highway image and its processed version. On the left, you'll see a raw frame featuring animals without any annotations. The right image, however, showcases the same scene after it's been analyzed by the YOLOv8-based detection model. Here, the detected animals—like cattle, deer, or goats—are highlighted within bounding boxes, complete with their class labels. This visual comparison clearly demonstrates the system's ability to accurately spot and classify animals in various environments, which is crucial for warning drivers and avoiding wildlife-vehicle collisions. The placement of the bounding boxes and the class labels reflect the model's successful inference, underscoring its potential for real-time monitoring on hig

Original Image Processed Image







VIII. CONCLUSION

The Smart Vision Road Safety System for Animal Detection Using Machine Learning demonstrates how artificial intelligence can improve road safety by identifying animals near roadways in real time. By combining computer vision techniques with machine learning algorithms, the system can detect animals from camera input and provide timely alerts to drivers, helping reduce the risk of wildlife-related accidents.

This approach enhances driver awareness, supports safer transportation, and contributes to the protection of both human lives and wildlife. The system is designed to operate efficiently under various road conditions and can be integrated with intelligent transportation and driver assistance systems.

REFERENCES

1. Juan Du. "Understanding of Object Detection Based on CNN Family and YOLO". Journal of Physics: Conference Series, Volume 1004, 2nd International Conference on Machine Vision and Information Technology (CMVIT 2018) 23–25 February 2018, Hong Kong
2. "Man-Wildlife Animal conflict in India: Reason, Issues, and Solutions", by IAS SQUAD, URL https://iassquad.in/man-wild-animal-conflict-in-india-reasons-issues-and-solutions/
3. Michael J. Wilber, Walter J. Scheirer, Phil Leitner, Brian Heflin, James Zott, Daniel Reinke, David K. Delaney, Terrance E. Boult. "Animal recognition in the mojave desert: Vision tools for field biologists". 2013 IEEE Workshop on Applications of Computer Vision (WACV).
4. "Man-Animal Conflict Rose in Gir After the Population Increase of Lions in Gir", Jagat, URL https://www.girnationalpark.in/news/man-animal-conflict-rose-in-gir-after-the-population-increase-of-lions-in-gir.html
5. Chengji Liu, Yufan Tao, Jiawei Liang, Kai Li, Yihang Chen. "Object detection based on YOLO network". 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC).
6. Suwarna Gothane. "A practice for object detection using YOLO algorithm". April 2021 International Journal of Scientific Research in Computer Science Engineering and Information Technology.
7. Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, Jieping Ye. "Object Detection in 20 Years: A Survey". Proceedings of the IEEE (Volume: 111, Issue: 3, March 2023).
8. Ajantha Vijaykumar, Subramaniyaswamy Vairavasundaram. "Yolo-based object detection models: A review and its applications". March 2024.
9. T Thomas Leonid, Harish Kanna, Claudia Christy V J, Hamritha A S, Chebolu Lokesh. "Human wildlife conflict mitigation using YOLO algorithm". 2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM).
10. Smat Saira Gillani, Muhammad Rizwan Munawar, Muhammad Talha, Salman Azhar.
11. Tingting Zhao, Xiaoli Yi, Zhiyong Zeng, Tao Feng. "MobileNet-YOLO Based Wildlife Detection Model: A Case Study in Yunnan Tongbiguan Nature Reserve, China (2021)". 01 January 2021.
12. Zhibin Ma, Yanqi Dong, Yi Xia, Delong Xu. "Wildlife real-time detection in complex forest scenes based on YOLOv5s deep learning network". April 2024.
13. Yolo-x, Yolo-r, Yolov7 Performance Comparison: A Survey". 8th International Conference on Artificial Intelligence and Fuzzy Logic System (AIFZ 2022).
14. Jansi Rani Sella Veluswami. "Human Wildlife Conflict Reduction Technology using YOLO Machine Learning Model". December 2021, International Journal of Natural Sciences.
15. M. J. Wilber, W. J. Scheirer, P. Leitner, B. Heflin, J. Zott, D. Reinke, et al., "Animal recognition in the Mojave Desert: Vision tools for field biologists," in Proc. IEEE Workshop on Applications of Computer Vision (WACV), Tampa, FL, USA, Jan. 2013, pp. 206–213.
16. H. Archana, K. Poornima, V. Raksha, H. R. Shwetha, and H. S. G. Supreeth, "Artificial neural network based image processing for wild animal detection and monitoring," Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 7, no. 5, pp. 273–278, 2017.
17. M. K. Gourisaria, U. Singh, V. Singh, and A. Sharma, "Performance enhancement of animal species classification using deep learning," in Proc. Int. Conf. on Computing, Communication and Learning, Bengaluru, India, Oct. 2022, pp. 208–219.
18. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, Sep. 2014.

19. M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in Proc. Int. Conf. on Machine Learning (ICML), Long Beach, CA, USA, Jun. 2019, pp. 6105–6114.
20. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
21. N. Mamat, M. F. Othman, and F. Yakub, "Animal intrusion detection in farming area using YOLOv5 approach," in Proc. 22nd Int. Conf. on Control, Automation and Systems (ICCAS), Jeju, South Korea, Nov. 2022, pp. 1–5.
22. S. Schneider, G. W. Taylor, and S. Kremer, "Deep learning object detection methods for ecological camera trap data," in Proc. 15th Conf. on Computer and Robot Vision (CRV), Toronto, ON, Canada, May 2018, pp. 321–328.
23. A. Saxena, D. K. Gupta, and S. Singh, "An animal detection and collision avoidance system using deep learning," in Advances in Communication and Computational Technology (ICACCT), Singapore: Springer, 2021, pp. 1069–1084.
24. A. Zhu, T. H. Li, and G. Li, "Towards automatic wild animal detection in low-quality
25. camera-trap images using two-channeled perceiving residual pyramid networks," in Proc. IEEE Int. Conf. on Computer Vision Workshops (ICCVW), Venice, Italy, Oct. 2017, pp. 2860–2864.
26. M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer, C. Packer, and J. Clune, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," Proc. Natl. Acad. Sci. U.S.A., vol. 115, no. 25, pp. E5716–E5725, Jun. 2018.
27. A. Gomez, G. Diez, A. Salazar, and A. Diaz, "Animal identification in low-quality camera-trap images using very deep convolutional neural networks and confidence thresholds," in Int. Symp. on Visual Computing, Las Vegas, NV, USA, Dec. 2016, pp. 747–756.
28. H. D. Patil and N. F. Ansari, "Intrusion detection and repellent system for wild animals using artificial intelligence of things," in Proc. Int. Conf. on Computing, Communication and Power Technology (IC3P), Pune, India, Jan. 2022, pp. 291–296.
29. P. Dave, A. Chandarana, P. Goel, and A. Ganatra, "An amalgamation of YOLOv4 and XGBoost for next-gen smart traffic management system," PeerJ Comput. Sci., vol. 7, p. e586, 2021.
30. K. Khatri, C. S. Asha, and J. M. D'Souza, "Detection of animals in thermal imagery for surveillance using GAN and object-detection framework," in Proc. Int. Conf. for Advancement in Technology (ICONAT), Goa, India, Jan. 2022, pp. 1–6.
31. A. Shukla and S. Anand, "Metric learning-based automatic segmentation of patterned species," in Proc. IEEE Int. Conf. on Image Processing (ICIP), Phoenix, AZ, USA, Sep. 2016, pp. 3982–3986.
32. D. Garg, P. Goel, S. Pandya, A. Ganatra, and K. Kotecha, "A deep learning approach for face detection using YOLO," in Proc. IEEE PuneCon, Pune, India, Nov. 2018, pp. 1–4.
33. Shaiful Mahmud, Khaleel Khan Mohammed, Vasu Raj Jain, Sarthak Anandkumar Shah. "Artificial Intelligence-Driven Predictive Models for Identifying Risk Factors of Chronic Diseases