Methods of Augmentation and Analysis of Deep Learning Based Stray Animal Recognition Architecture for Mishap Prevention

Vinayak B. Sutar[1], Kshama V. Kulhalli[2]
[1] *DKTE's Textile & Engineering Institute, Rajwada, Ichalkaranji, Maharashtra, India*
[2] *Dr. D.Y. Patil College of Engineering & Technology, Kolhapur, India*

***Abstract*—According to reports from the world bank, 64.13% of India's population resides in rural areas. As a result, domestic and stray animals such as cows and buffaloes are frequently seen on or near roadways. It is crucial for drivers or intelligent vehicular systems to detect these stray animals to regulate speed. Each year, hundreds of people and stray animals are injured or killed in vehicle-animal collisions, both during the day and at night. Records from the veterinary department show a nearly 23% increase in such incidents over the last six years. This paper explores a deep learning-based system for detecting stray animals to alert drivers. Deep learning algorithms generally require large datasets for training, testing, and validation. Given the limited availability of public datasets, various data augmentation techniques could be used. For augmentation, we applied techniques such as horizontal flipping, color space transformation, rotation, shear, zoom, intensity transformation, and resizing. The model's performance was evaluated using the training and validation of both datasets over different epochs and batch sizes. The model achieved a true positive rate of 80% to 85%, with 92.5% accuracy on the non-augmented dataset and 91% accuracy on the augmented dataset when tested on real-world camera footage.**

***Index Terms*—Data Augmentation, Deep Learning, Stray Animal Recognition, Yolo.**

I. INTRODUCTION

The proposed research pertains to road safety engineering, specifically focusing on an intelligent vehicular system that can detect and recognize cattle crossing roads or appearing in front of vehicles. It is frequently observed that cattle sit or stand in the middle of roads, making them visible to drivers for speed control. However, problems arise when drivers are unable to see the cattle during both daytime and nighttime driving, leading to collisions. These incidents are a critical road safety issue. The research aims to mitigate accidents involving vehicles and domestic animals by introducing a cattle detection model. This model addresses two types of collisions: direct and indirect. In a direct collision, the vehicle hits the cattle head-on, potentially causing the animal to be thrown to the roadside depending on the speed and movement. The type of vehicle also plays a significant role in the severity of the collision; in some cases, cattle can hit the windshield or cause the vehicle to overturn. In an indirect collision, while attempting to avoid hitting the cattle, the vehicle may collide with another vehicle, resulting in dangerous accidents that can harm both cattle and vehicle occupants. This paper focuses on cattle detection and recognition using the latest single-shot object detector, YOLOv5. The low-cost, computer vision-based automatic detection system is designed to help prevent accidents between cattle and vehicles on the road.

II. LITERATURE SURVEY

Over the past decade, numerous research studies have focused on detecting and monitoring wildlife using camera traps. (Zhang, Z., et al. 2016) developed a feature description method where a cross-frame temporal patch verification technique distinguishes between actual animals and background patches. This method, compared to Faster R-CNN, showed a 4.5% improvement in detection accuracy. Another animal detection model, created by (Verma, G.K. and Gupta, P. 2018), employed self-learned Deep Convolutional Neural Network (DCNN) features, with classification performed using machine learning algorithms like SVM, KNN, and ensemble trees. Their experiments were conducted on a standard camera trap dataset. In Brazil, researchers proposed a road-based animal detection system using cameras and a KNN learning

model. (Saxena, A., et al. 2020) introduced animal detection using SSD and R-CNN neural network architectures, achieving a mean average precision (mAP) of 80.5% and 82.11%, respectively. (Sharma, S. and Shah, D. 2016) focused on animal detection on Indian roadways, using a HOG feature descriptor applied in computer vision and image processing. With this feature and a boosted cascade classifier system, they achieved 82.5% accuracy in detecting animals. (Trnovszky, T., et al. 2017) proposed a Convolutional Neural Network (CNN) for classifying four specific animal classes and compared their method with established image recognition techniques such as PCA, LDA, LBPH, and SVM.

Hung Nguyen et al. developed an automated wildlife monitoring system utilizing DCNN for the identification and recognition of wild animals, achieving 90.5% accuracy for animals in South-Central Victoria, Australia. (Yang, Z., et al. 2023) surveyed automated data augmentation technology in perspective of image classification. Authors proposed AutoDA model with three key components such as a search space, a search algorithm and an evaluation function. Authors also discussed the pros and cons of the proposed model. In our research, we conducted experiments using a customized object detection model for cattle within the Google Colab Jupyter Notebook environment, which provides an Nvidia Tesla K80 GPU with 12GB of memory. A total of over 23,000 non-augmented and augmented image datasets of Indian cattle were utilized for training and testing. This dataset comprises various animals, including cows, bulls, calves, and buffaloes, collected primarily from on the roadways and the platforms such as Kaggle, Alamy, Shutterstock, Indiatimes, Indian Express, and India Today.

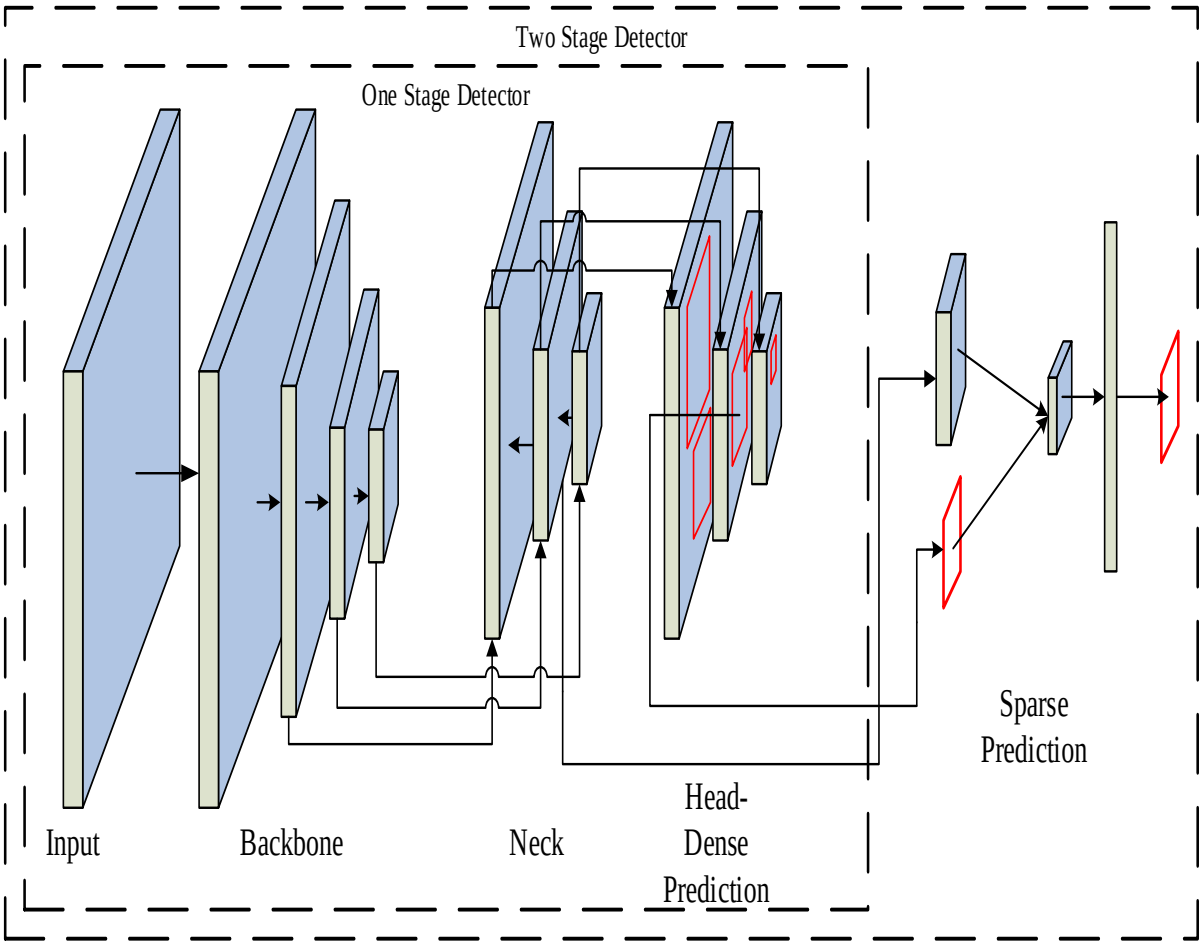

Fig. 1: Object Detection Process in YOLO Network

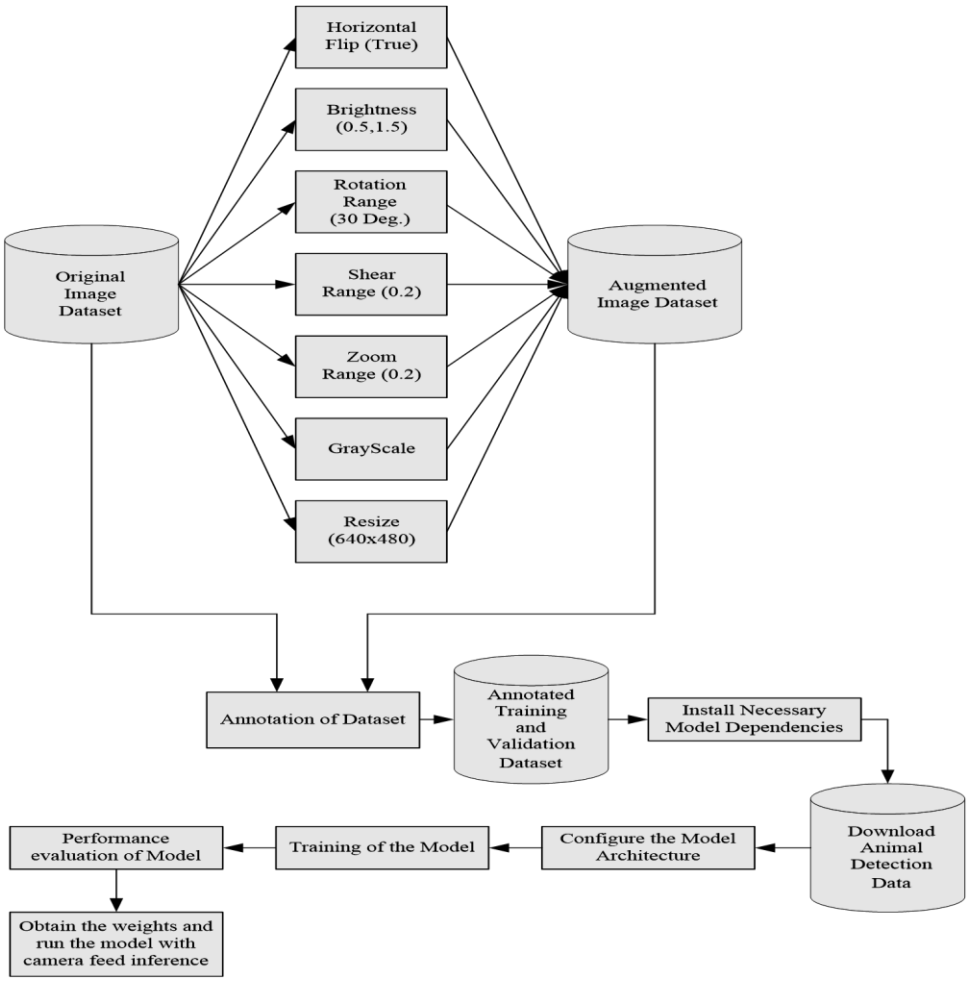

Fig. 2: Nine Stage Deep Learning Network with Image Augmentation Steps

III. COLLECTION OF DATASETS FOR TRAINING

While dealing with the challenges of limited training data for deep learning models, we have often faced a unique dilemma. In such situations, we came across the concept of image augmentation. Image augmentation is an effective strategy for expanding a dataset by creating various transformed versions of the original images. A common practice is to augment images and store them in a numpy array or a specified folder. We must admit that we relied on this traditional approach until we discovered the ImageDataGenerator class. The Keras ImageDataGenerator is an invaluable tool that facilitates real-time image augmentation during model training ('Keras ImageDataGenerator and data augmentation' 2019). This feature allows for the application of random transformations to each training image as it is input into the model, enhancing its robustness while conserving memory resources. Image augmentation involves applying different transformations to original images, resulting in multiple modified versions of the same image. Transformations like shifting, rotating, or flipping introduce unique characteristics to each copy as shown in Fig. 3. Importantly, these slight variations do not change the original image's target class; instead, they provide alternative representations of real-world objects. Consequently, image augmentation is a widely used technique in developing deep learning models. The Keras ImageDataGenerator class offers a convenient and efficient way to augment images, featuring a broad range of techniques, including standardization, rotation, shifts, flips, brightness adjustments, and more. One of the key benefits of using the Keras

ImageDataGenerator is its ability to perform real-time data augmentation. This means it generates augmented images on the fly while the model is training, ensuring that the model encounters new image variations with each epoch, thus mitigating the risk of overfitting that could result from repeatedly using the same original images. Additionally, ImageDataGenerator helps reduce memory usage. Without this class, all images would be loaded at once, but when using ImageDataGenerator, images are loaded in batches, leading to substantial memory savings. The second stage of model is annotating the training and validation dataset which is given below.

(A)

(B)

(C)

(D)

(E)

(F)

Fig. 3: Image Augmentation: (A) Image Rotate, (B) Image Flip, (C) Image Enhancement, (D) Image Zoom, (E) Image Shear, (F) Image Grayscale

IV. DATASET ANNOTATION PROCESS

In deep learning and object recognition architectures, it is essential to oversee the learning process through annotation of bounding box. At this process, all samples in the dataset used for training are annotated. Since the research work focuses on a cattle recognition model, each animal instance in the sample is annotated with label corresponding to the class of cattle, along with the coordinates of bounding box [x1, y1] and [x2, y2].

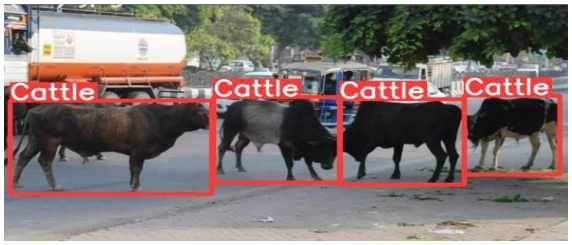

Fig. 4: Labeled Image

```
0 0.888643 0.546225 0.166667 0.334454
0 0.698378 0.574776 0.216814 0.358926
0 0.481563 0.566619 0.219764 0.358926
0 0.192478 0.597889 0.349558 0.394275
```

Fig. 5: Labeled Text File

Fig. 4 displays the annotated image from batch processing. Here the label "cattle" within the bounding box indicates the class of object, accompanied by four annotated bounding boxes around the cattle. In Fig. 5, the first number '0'

denotes the class number of object, along with the corresponding XY coordinates for the cattle in the image.

V. ANNOTATED TRAINING AND VALIDATION DATASET

All the original and augmented annotated images in the dataset are organized into separate training and validation subdirectories, with their corresponding labels compiled in these folders as well. The final train and validation directories, which include the labelled images and their respective text files, are now prepared for model building.

VI. INSTALLING THE ARCHITECTURAL DEPENDENCIES

In the research work, the object recognition architecture is built on YOLO version 5. At this stage, we used the repository of YOLO and deployed the required dependencies, which establishes the programming environment needed for training and executing our cattle detection model. Below are the commands used to clone the repository and install the required dependencies.

VII. DOWNLOADING THE DATASET

At this stage, the dataset of annotated samples and corresponding labelled text files are downloaded into the Colab notebook. The architecture now needs to access the custom animal dataset for training within this Colab environment. The path for custom dataset declared by utilizing commands given in a YAML file.

```
# Training and Validation dataset
train: ../images/train/  # training image samples
val: .. /images/val/  # validation image samples
test:  # testing image samples (optional)
# Classes
nc: 1  # Number of classes
names: [ 'Cattle' ]  # Name of class
```

The first command indicates the directory path containing all the training images, while the second command specifies the path for the validation. The third command is optional and can be used for testing images. The total number of classes and their titles are listed under the classes section. In current research, the class name ‘cattle’ to be detected by the architecture, thus it is labeled as '1' and the class name is set to ‘Cattle’.

VIII. CONFIGURATION OF ARCHITECTURE

The YOLO version 5 model offers various sizes such as v5 small, v5 medium, and v5 large. In this stage, the architecture is designed for the fastest variant, small YOLOv5, by considering yolov5s.yaml file. However, this part can be skipped, allowing us to proceed directly to the next training phase.

IX. TRAINING OF ARCHITECTURE

At this stage, the architecture will be trained using custom animal dataset. Prior to initiating the training process, the following options need to be configured before to start the training.

```
!python train.py --img 640 --batch 3 --epochs 100 --data custom_data.yaml --weights yolov5s.pt
```

Where,

img: input image size
batch: batch size
epochs: the number of training epochs.
data: the path to our yaml file
cfg: model configuration
weights: a custom path to weights
cache: cache images for faster training

After configuring all the options and file paths for our custom dataset, primarily the model was trained using the non-augmented limited dataset and further trained using augmented datasets. The training was conducted across various epoch values such as 60, 100, 150 and so on. Ultimately, the weights were acquired by achieving a higher mean average precision of validation. Following the training, the

Dataset Type	Non-augmented			Non-augmented + Augmented		
Batch Value	2	2	3	7	10	16
Epoch Value	60	150	150	60	100	100
Precision	0.64	0.87	0.84	0.90	0.90	0.91
Recall	0.66	0.76	0.73	0.82	0.81	0.82
mAP 0.5	0.63	0.83	0.78	0.90	0.89	0.89
Accuracy	0.92	0.91	0.91	0.91	0.905	0.905
Sensitivity	0.85	0.82	0.82	0.82	0.81	0.81
F Measure	0.91	0.90	0.90	0.90	0.90	0.90

performance of architecture was assessed separately for both the non-augmented and augmented models.
Table 1: Training performance of Model

Fig. 6 displays the output image samples of the trained architecture. The trained architecture has successfully predicted all cattle in the image samples, with the prediction percentages indicated above the bounding boxes for each cow, bull, and other animals.

Fig. 6: Test Batch Prediction Results

X. RESULTS AND PERFORMANCE EVALUATION

The proposed deep learning model for identifying stray animals achieved detection accuracies ranging from 90.5% to 92%. The model underwent training, validation, and testing using publicly available datasets and various image augmentation techniques. Evaluation of the model trained with a natural dataset revealed detection accuracies between 91% and 92%, with a peak precision of 87%, a recall rate of 76%, and a mean Average Precision (mAP) of 83% at an Intersection over Union (IoU) threshold of 0.5 as shown in table 1.

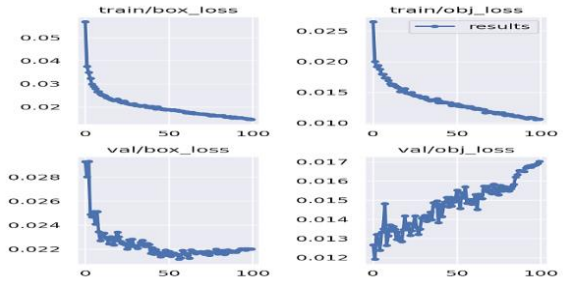

Fig. 7: Plots of Box Loss, Objectness

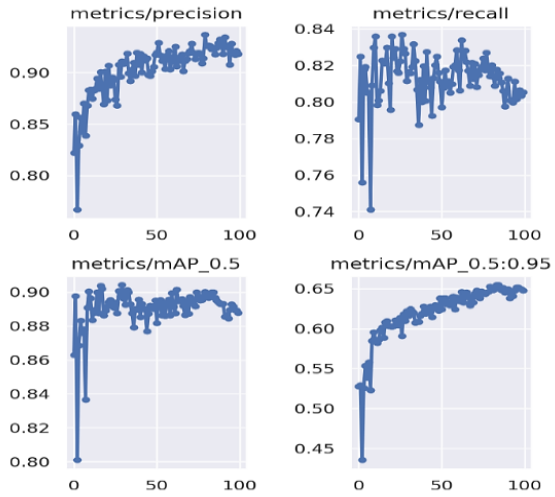

Fig. 8: Plots of Precision, Recall and Mean Average Precision Loss

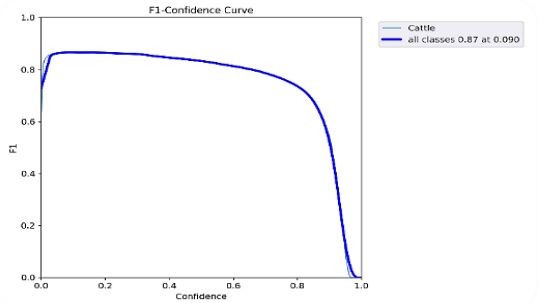

Fig. 9: F1 - Confidence Plot

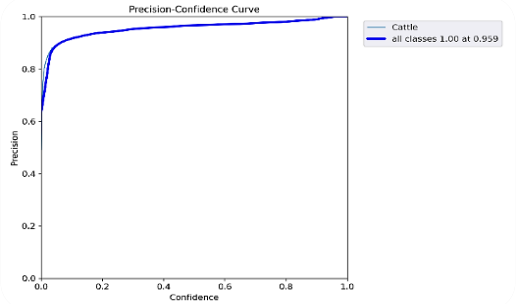

Fig. 10: Precision - Confidence Plot

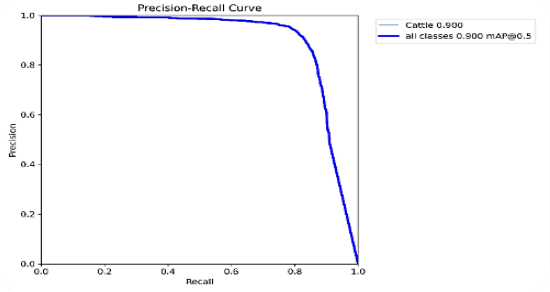

Fig. 11: Precision - Recall Plot

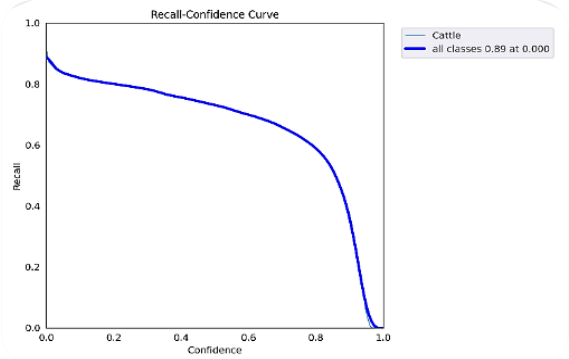

Fig. 12: Recall - Confidence Plot

Moreover, when the model was trained using both natural and augmented datasets, it maintained similar high accuracy levels, achieving a maximum precision of 91%, a recall rate of 82%, and a mAP@0.5 of 90%. The plots of performance parameters obtained at highest epoch value are showcased in following figures. As a result, the proposed deep learning animal detection model architecture, trained on this combined dataset, was tested using real-world camera feeds and sample videos, consistently achieving a confidence score of over 91%. The training and validation plots for box loss and objectness loss are shown in Fig. 7, while the precision, recall, and mAP plots are presented in Figure 8. Additionally, the F1 measure versus confidence curve is illustrated in Fig. 9. Sample image frames from the trained stray animal detection model are displayed in Fig. 13. The model successfully identified stray animals on roadways under challenging conditions, including varying lighting and distances, while maintaining high confidence scores.

Fig. 13: Frames from the Tested Video Input

XI. CONCLUSION

The system has been designed and tested by training the deep learning model with non-augmented dataset and augmented dataset. Table 2 presents a comparison of the performance parameters for the model trained using a non-augmented dataset versus the model trained with both non-augmented and augmented datasets. The model trained on the non-augmented dataset exhibited higher detection accuracy about 92% as compared to the model that utilized both datasets, but it performed poorly in case of precision @ 87%, recall@76%, and mAP@83%. Conversely, the model trained with both non-augmented and augmented datasets achieved better precision@91%, recall@82%, and mean average precision 90%@0.5 rates compared to the model trained solely on the non-augmented dataset. Overall, the model demonstrated acceptable detection accuracy for cattle detection.

Performance Parameters	Model with Non-Augmented Dataset	Model with Non-Augmented and Augmented Dataset
Detection Accuracy	91%-92%	90.5%-91%
Highest Precision	87%	91%
Recall Rate	76%	82%
mean Average Precision	83%	90%

Table 2: Performance Comparison of Trained Model

REFERENCES

[1] Zhang, Z., et al. (2016) 'Animal detection from highly cluttered natural scenes using spatiotemporal object region proposals and patch verification', IEEE Transactions on Multimedia, 18(10), pp. 2079–2092.

[2] Verma, G.K. and Gupta, P. (2018) 'Wild animal detection from highly cluttered images using deep convolutional neural network', International Journal of Computational Intelligence and Applications, 17(4), p. 1850021.

[3] Saxena, A., et al. (2020) 'An animal detection and collision avoidance system using deep learning', Lecture Notes in Electrical Engineering, pp. 1069–1084.

[4] Sharma, S. and Shah, D. (2016) 'Real-time automatic obstacle detection and alert system for driver assistance on Indian roads', Indonesian Journal of Electrical Engineering and Computer Science, 1(3), p. 635. Available at: https://doi.org/10.11591/ijeecs.v1.i3.pp635-646 (Accessed: 5 October 2019).

[5] Trnovszky, T., et al. (2017) 'Animal recognition system based on convolutional neural network', Advances in Electrical and Electronic Engineering, 15(3). Available at: https://doi.org/10.15598/aeee.v15i3.2202 (Accessed: 22 November 2019).

[6] Kaur, M. and Randhawa, R. (2022) 'Animal detection: Techniques, challenges and future scope', International Journal of Innovative Technology and Exploring Engineering (IJITEE). Available at: www.ijitee.org/portfolio-item/a4974119119/ (Accessed: 24 March 2025).

[7] Antônio, W.H.S., et al. (2019) 'A proposal of an animal detection system using machine learning', Applied Artificial Intelligence, 33(13), pp. 1093–1106. Available at: https://doi.org/10.1080/08839514.2019.1673993.

[8] Yang, Z., et al. (2023) 'A survey of automated data augmentation algorithms for deep learning-based image classification tasks', Knowledge and Information Systems, 65(7), pp. 2805–2861. Available at: https://doi.org/10.1007/s10115-023-01853-2 (Accessed: 12 July 2023).

[9] Sibanda, V., et al. (2019) 'Design of an animal detection system for motor vehicle drivers', Procedia CIRP, 84, pp. 755–760. Available at: https://doi.org/10.1016/j.procir.2019.04.175.

[10] Wilkins, D.C., et al. (2019) 'Animal-vehicle collisions in Texas: How to protect travelers and animals on roadways', Accident Analysis & Prevention, 131, pp. 157–170. Available at: https://doi.org/10.1016/j.aap.2019.05.030 (Accessed: 9 May 2020).

[11] Keerthana, S. and Mary Shyla, E. (2018) 'A literature survey on wild animal detection using various data mining techniques', International Journal of Research and Analytical Reviews, 5(3), pp. 616–622.

[12] Nguyen, H., et al. (2017) 'Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring', 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA). Available at: https://doi.org/10.1109/dsaa.2017.31.

[13] Jakobsson, L., et al. (2015) 'Large animal crashes: The significance and challenges', 2015 International Research Council on the Biomechanics of Injury, IRCOBI 2015, Lyon, France, 9-11 September 2015, pp. 302–314. Available at: research.chalmers.se/en/publication/247535 (Accessed: 24 March 2025).

[14] Singh, S., et al. (2018) 'Improving animal-human cohabitation with machine learning in fiber-wireless networks', Journal of Sensor and Actuator Networks, 7(3), p. 35.

[15] Saleh, K., et al. (2018) 'Effective vehicle-based kangaroo detection for collision warning systems using region-based convolutional networks', Sensors, 18(6), p. 1913. Available at: https://doi.org/10.3390/s18061913 (Accessed: 18 September 2019).

[16] Olorunshola, O.E., et al. (2023) 'A comparative study of YOLOv5 and YOLOv7 object detection algorithms', Journal of Computing and Social Informatics, 2(1), pp. 1–12. Available at: https://doi.org/10.33736/jcsi.5070.2023.

[17] Guo, S., et al. (2022) 'Research on mask-wearing detection algorithm based on improved YOLOv5', Sensors, 22(13), p. 4933. Available at: https://doi.org/10.3390/s22134933 (Accessed: 2 July 2022).

[18] 'Keras ImageDataGenerator and data augmentation' (2019) PyImageSearch. Available at: pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/.