Real-Time Wildlife Poaching Detection System Using YOLOv12 and Lightweight CNN Architecture

Harsha Naik, Laxmi Madiwalar, Amulya Chougale, Basalingappa Patil, and Sharada Shiragudikar

KLE Technological University, Hubballi, India
`01FE23BCS267@kletech.ac.in` `01FE23BCS275@kletech.ac.in`
`01FE24BCS416@kletech.ac.in` `01FE24BCS419@kletech.ac.in`
`sharada.shiragudikar@kletech.ac.in`

Abstract. Wildlife poaching continues to endanger biodiversity, motivating automated vision models that can detect both animals and humans in unconstrained forest imagery. In this work, a YOLOv12n detector is fine tuned on a Roboflow wildlife dataset containing 5,204 labelled images, split into 3,650/1,037/517 for training/validation/testing. The data set provides 38 object categories that span multiple animal species as well as human-related classes. In the held test set, the model achieves Precision= 0.790, Recall= 0.738, mAP@0.5= 0.796, and mAP@0.5:0.95= 0.615. The resulting `best.pt` checkpoint is approximately 5.5 MB, indicating that a compact YOLOv12 detector can provide a practical perception backbone for real-time multi-species monitoring and poaching-prevention pipelines on resource-constrained deployments.

Keywords: Poaching detection, YOLOv12, object detection, wildlife monitoring, deep learning, edge deployment

1 Introduction

Wildlife poaching threatens biodiversity, but manual patrols and fixed-camera monitoring provide limited coverage and delayed response(3)(8)(9)(11). Modern computer vision enables automated detection of wildlife and human activity in near real time. (1)(2)

Field deployment is constrained by limited compute, power, and connectivity. While Faster R-CNN achieves strong localization, it is often impractical on edge hardware; YOLO-style single-stage detectors improve throughput while retaining competitive accuracy. (2)(4)(3)(8)

A YOLOv12 based(14) wildlife-and-human detection pipeline has been developed and trained on a Roboflow dataset (5,204 images, 38 classes) and evaluated with precision, recall, and mAP plus diagnostic plots. The resulting model is compact ($\sim$5.5 MB) for real-time multi-class monitoring.

2 Related Work

Early object detection relied on hand-crafted features and rule-based pipelines that degrade under clutter, scale, and illumination variation, while deep learning enabled end-to-end feature learning with improved robustness. (2)

Two-stage detectors (e.g., R-CNN variants) improve localization via region proposals, but typically increase latency and compute, limiting real-time and edge deployment. (2) Single-stage detectors such as YOLO predict boxes and classes in one forward pass and are widely used when throughput is critical. (1)(4)

To support resource-constrained deployment, lightweight backbones (e.g., MobileNet) (5) and compression methods such as pruning, quantization, and knowledge distillation are used to reduce memory and runtime while maintaining accuracy.

Wildlife monitoring introduces additional challenges (occlusion, camouflage, viewpoint/scale variation, and class imbalance), motivating class-wise evaluation and diagnostic plots beyond aggregate metrics. (8)

Building on these directions,YOLOv12n was fined-tuned for multi-class wildlife-and-human detection using a reproducible training and evaluation pipeline.

3 Proposed Work

The proposed system implements an end-to-end pipeline that starts from a labelled wildlife dataset and produces a trained detector capable of localizing multiple wildlife species as well as human presence in unconstrained imagery. The workflow is designed to be reproducible and deployment-oriented, covering dataset acquisition, normalization into a consistent YOLO(14) directory layout, model training, and model evaluation using standard detection metrics and diagnostic artifacts.

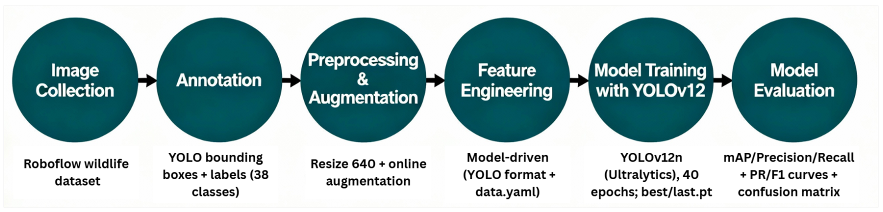
Image Collection
Annotation
Preprocessing & Augmentation
Feature Engineering
Model Training with YOLOv12
Model Evaluation
Roboflow wildlife dataset
YOLO bounding boxes + labels (38 classes)
Resize 640 + online augmentation
Model-driven (YOLO format + data.yaml)
YOLOv12n (Ultralytics), 40 epochs; best/last.pt
mAP/Precision/Recall + PR/F1 curves + confusion matrix

Fig. 1: Workflow for the implemented YOLOv12-based wildlife detection pipeline. Data processing is performed during dataset preparation and normalization, while training and evaluation are executed using the Ultralytics training loop and its diagnostic outputs.

For reproducibility, dataset download and restructuring are scripted, and the final dataset is organized in the standard YOLO format: `images/{train,val,test}` and `labels/{train,val,test}`. (14)The normalization preserved the provided split and produced 3,650/1,037/517 images for train/validation/test, respectively.

As shown in Fig. 1, the workflow is organized into six main stages.

3.1 Image Collection

A publicly available dataset is used that is Roboflow wildlife object-detection dataset containing 5,204 annotated images. (7) The dataset includes 38 labelled categories spanning multiple animal species and human-related classes, and it is distributed with YOLO-format bounding-box annotations. (14) A predefined split of 3,650/1,037/517 images for training/validation/testing is retained throughout to enable consistent evaluation.

3.2 Annotation

The dataset is sourced from Roboflow with pre-existing YOLO-format bounding-box annotations, where each image has a corresponding text file containing a class ID and normalized box coordinates. (14) The dataset's original class mapping is retained and reorganized the files into the standard YOLO directory layout (`images/{train,val,test}` and `labels/{train,val,test}`) after verifying image–label consistency. (14) The provided split is preserved (3,650 /1,037/ 517 for train /val /test) to avoid leakage and maintain comparable evaluation. During processing 5,204 images and 5,206 label files have been observed; unmatched labels are ignored via basename pairing during normalization.

Table 1: Dataset split summary

Split	Images	Labels
Train	3650	3652
Validation	1037	1037
Test	517	517
Total	5204	5206

3.3 Preprocessing and Augmentation

Training uses the Ultralytics YOLO pipeline with a fixed input size of 640×640; aspect ratio is preserved via letterbox-style resizing and the default Ultralytics preprocessing.

To improve robustness to illumination changes, viewpoint variation, and partial occlusion,the standard YOLO augmentations have been enabled (flip $p = 0.5$, HSV jitter: `hsv_h`=0.015, `hsv_s`=0.7, `hsv_v`=0.4, `translate`=0.1, `erasing`=0.4) and lightweight Albumentations perturbations (Blur, MedianBlur, ToGray, CLAHE; $p = 0.01$ each).

3.4 Feature Engineering

Feature representations are learned end-to-end by YOLOv12 (backbone/neck) from RGB inputs; no hand-crafted descriptors or external features are added. (14) For reporting and optional downstream logic only, simple post-inference box attributes (centroid, aspect ratio, and area)have been computed from predicted (x_c, y_c, w, h), which do not affect training. (14)

3.5 Model training with YOLOv12

A YOLOv12n detector using the Ultralytics training pipeline has been trained on the prepared dataset, with on-the-fly augmentation enabled. Training was run for 40 epochs with input resolution 640×640 and batch size 16, and the checkpoint `best.pt` was selected based on validation performance. (14)

Although `lr0`=0.01 was specified, Ultralytics `optimizer=auto` overrode manual hyperparameters and selected AdamW with an effective learning rate of $\approx 2.38 \times 10^{-4}$ and momentum 0.9 in the run. Training progress was monitored using loss curves and validation metrics (precision, recall, and mAP@0.5). (14)

YOLOv12(14) is a single-stage detector that predicts bounding-box coordinates and class confidences in a single forward pass. Ultralytics optimizes a multi-term detection objective that jointly balances localization accuracy and classification quality, with Distribution Focal Loss (DFL) used for improved bounding-box refinement. (14)

To capture this training objective concisely, The optimization is as follows:

$$L_{\text{total}} = L_{\text{loc}} + L_{\text{cls}} + L_{\text{dfl}} \tag{1}$$

where L_{loc} represents an IoU-based localization loss (CIoU in Ultralytics), L_{cls} is the classification loss, and L_{dfl} is the Distribution Focal Loss term that refines localization.

3.6 Model Evaluation

The final detector (`best.pt`), chosen using validation performance, was then assessed only on the untouched test partition (517 images) to estimate generalization. The usual detection metrics have been reported:precision, recall, mAP@0.5, and mAP@[0.5:0.95] (mean AP averaged over IoU thresholds from 0.50 to 0.95). (14) On the test split, the model obtained precision 0.790, recall 0.738, mAP@0.5 0.796, and mAP@[0.5:0.95] 0.615.

To diagnose failure modes beyond aggregate scores, inspected class-wise results and Ultralytics diagnostics were inspected, including confusion matrices and confidence-sweep curves (precision–recall and F1). (14) For qualitative verification and reproducibility, per-image predictions to a JSON artifact were exported (`evaluation/testmetrics/predictions.json`) and reviewed the generated visualizations for typical success and failure cases. (14)

Training configuration (for reproducibility): Training used the Ultralytics YOLO pipeline for 40 epochs with input size 640×640, batch size 16, augmentation enabled, and early stopping with patience 20. Although `lr0=0.01` was specified, `optimizer=auto` overrides manual hyperparameters and selected AdamW with an effective learning rate of $\approx 2.38 \times 10^{-4}$ and momentum 0.9 in the run.

Deployment note: All reported results correspond to the trained detector checkpoint as evaluated in the Ultralytics pipeline; additional deployment optimizations (e.g., pruning or INT8 quantization) were not applied to the evaluated model and are left for future work. (14)

4 Results

The section reports quantitative performance and diagnostic artifacts for YOLOv12-based wildlife detector trained with the Ultralytics pipeline. Following has been analyzed: (i) training dynamics over 40 epochs via Ultralytics loss/metric logs, (ii) class-wise behaviour and confusion patterns, (iii) confidence-sweep behaviour using precision–recall and F1 curves, and (iv) held-out test-set performance using standard detection metrics.

4.1 Training Dynamics and Convergence

Figure 2 summarises the evolution of the three optimization components reported by Ultralytics (box loss, classification loss, and DFL loss) across the run. Training ran to completion for all 40 epochs (*no early-stop termination occurred*) and produced the standard Ultralytics checkpoints (`best.pt` and `last.pt`). As expected for detector optimization, most gains occur early, followed by smaller incremental improvements as localization and classification refine. Small short-term fluctuations"especially in early epochs"are typical under mini-batch optimization with online augmentation and do not, by themselves, indicate instability. (14)

4.2 Performance Metrics

The YOLOv12 detector was evaluated on a held-out test split (517 images, 682 instances; 38 classes), achieving Precision = 0.790, Recall = 0.738, mAP@0.5 = 0.796, and mAP@0.5–0.95 = 0.615. (14) To keep evaluation interpretable at 38 classes, compact diagnostics have been reported: a normalized confusion matrix and confidence-sweep curves (precision–recall and F1) for selecting the operating point.

4.3 Confusion Matrix Analysis

The normalized confusion matrix highlights systematic inter-class confusions and helps distinguish true class ambiguity from confidence-threshold effects.

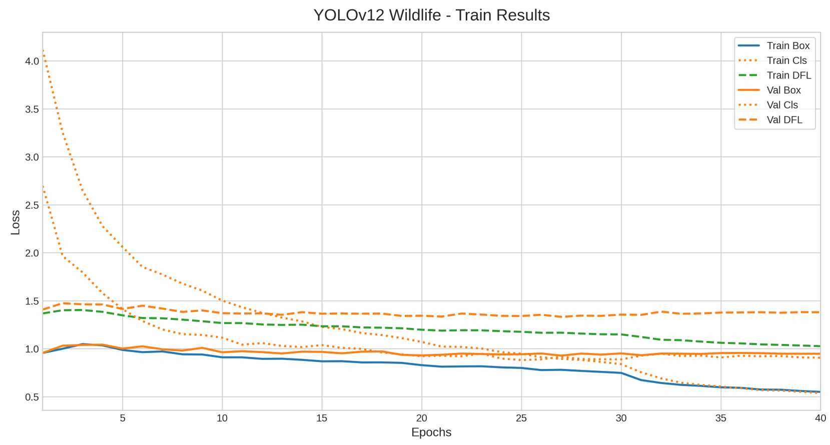

Fig. 2: Training and validation loss curves over 40 epochs (Box, Cls, and DFL) showing stable convergence.

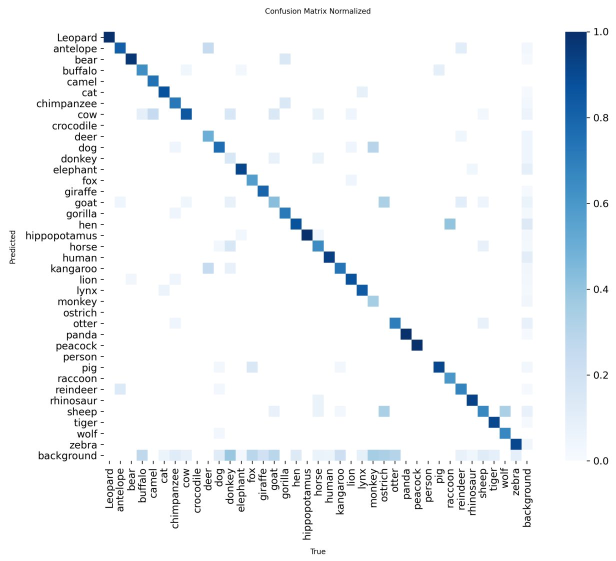

Fig. 3: Normalized confusion matrix for multi-class wildlife detection, showing per-class true positives (diagonal) and inter-class confusion (off-diagonal).

4.4 Precision–Recall Curve and Operating Point

Figure 4 shows the aggregated precision–recall behaviour across confidence thresholds. (14) Overall test performance is mAP@0.5 = 0.796. (14) For consistent inference, the operating confidence threshold is fixed at $\tau = 0.50$ for all reported outputs. (14)

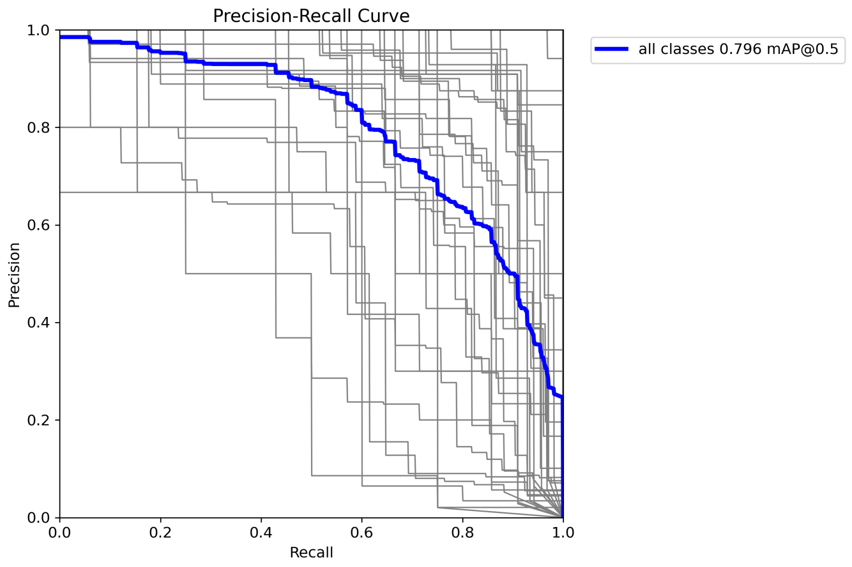

Fig. 4: Global precision–recall curve aggregated across all classes (test mAP@0.5 = 0.796) and the selected confidence operating threshold.

4.5 F1-Score Optimization

Figure 5 plots F1-score versus confidence threshold and indicates a peak around a mid-range τ. The value of $\tau = 0.50$ is retained to avoid post-hoc threshold tuning and to keep reporting consistent across splits. The sweep still highlights confidence thresholding as a simple deployment knob: lower τ typically increases recall, while higher τ typically increases precision.

4.6 Performance Metrics Over Epochs

Figure 6 tracks four key metrics:precision, recall, mAP@0.5, and mAP@0.5–0.95"over training epochs on the validation split. The curves increase rapidly during the early and mid training phase and gradually plateau after approximately 25–30 epochs, indicating convergence with diminishing gains from further training.

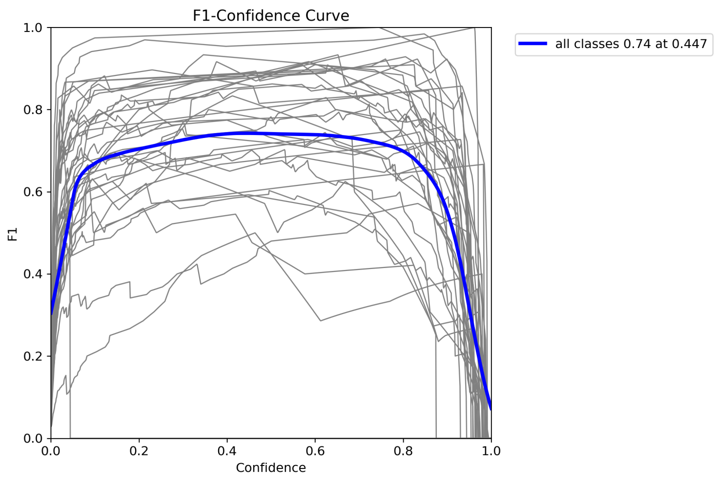

Fig. 5: F1-score versus confidence threshold, highlighting the threshold region that best balances precision and recall.

The final validation metrics obtained for the best checkpoint are precision = 0.825, recall = 0.689, mAP@0.5 = 0.772, and mAP@0.5–0.95 = 0.578.

4.7 Comparative Baseline Analysis

The test-set performance has been compared with a baseline detector used in experiments. (7) As shown in Table 2, the proposed model increases mAP@0.5 (0.769 $\rightarrow$ 0.796) and recall (0.673 $\rightarrow$ 0.738), with a moderate drop in precision (0.837 $\rightarrow$ 0.790). (7) Overall, the configuration prioritizes recall,reducing missed detections at the cost of more false positives, which is often acceptable in safety-critical monitoring.

Table 2: Test-set comparison against baseline configuration

Method	Precision	Recall	mAP@0.5
Baseline (reference)	0.837	0.673	0.769
Proposed (YOLOv12-based)	0.790	0.738	0.796

4.8 System Limitations and Mitigation

Although overall performance is encouraging, evaluation revealed limitations: occlusion (vegetation/overlap) reduces recall for some classes, and low-light”especially night-time infrared”remains challenging even after histogram equalization.

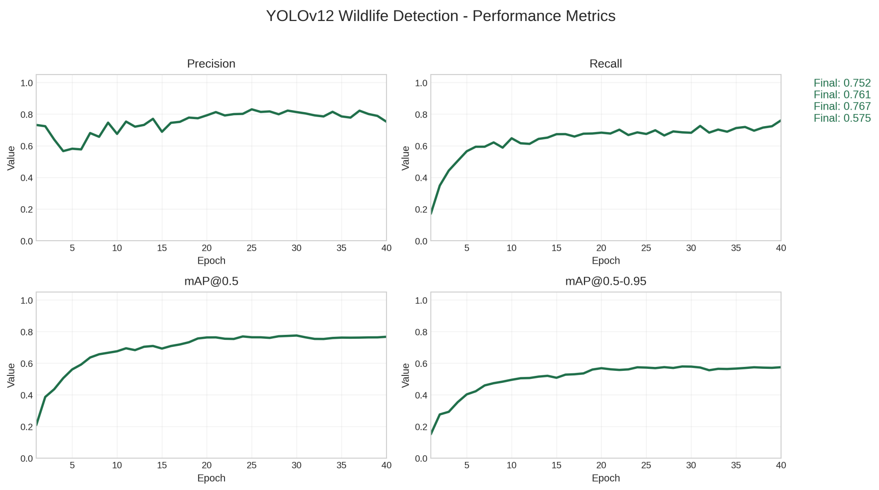

Fig. 6: Validation metrics over training epochs: precision, recall, mAP@0.5, and mAP@0.5–0.95, showing convergence by approximately epoch 30.

Table 3: System Limitations and Proposed Mitigation Strategies

Limitation	Cause	Mitigation
Occlusion	Vegetation, groups	Multi-frame voting, temporal integration
Low-light	Night IR imagery	Histogram equalization, specialized filters
Class imbalance	Limited rare species	Augmentation, weighted loss, synthetic data generation
False alarms	Wind, shadows	Temporal smoothing, ensemble methods

5 Conclusion

A real-time YOLOv12-based wildlife and human detector for 38 classes has been presented. On the held-out test split (517 images, 682 instances), the model achieved Precision = 0.790, Recall = 0.738, mAP@0.5 = 0.796, and mAP@0.5–0.95 = 0.615. The resulting checkpoint is lightweight (`best.pt` $\approx$ 5.51 MB), supporting practical edge deployment. Future work should improve robustness under hard conditions (low-light/night, motion blur, heavy occlusion, rare classes) and validate performance on longer continuous video streams to estimate real-world false-alarm and miss rates.

Bibliography

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, 2016.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in Proc. IEEE CVPR, pp. 580–587, 2014.

[3] J. Smith, A. Kumar, et al., “Unmanned Aerial Surveillance and Tracking System in Forest Areas for Poachers and Wildlife,” IEEE Trans. Industrial Informatics, vol. 18, no. 3, pp. 1950–1965, 2021.

[4] X. Wang, Z. Zhang, T. Zhao, et al., “Distribution Focal Loss for Dense Object Detection,” in Proc. Int'l Conf. on Computer Vision (ICCV), pp. 9408–9417, 2020.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv preprint arXiv:1704.04861, 2017.

[6] J. Robinson, D. Aiken, M. Brehm, and R. Van Etten, “Roboflow Universe: Open Public Computer Vision Datasets,” Roboflow, 2022. [Online]. Available: https://universe.roboflow.com

[7] Final Project Datasets Creation Training, “Animal & Human Detection Dataset,” Roboflow Universe, 2025. [Online]. Available: https://universe.roboflow.com/final-project-datasets-creation-training/animal-human-detection

[8] N. M. Schneider and H. Z. Tan, “Automated Large-Scale Animal Detection for Aerial Wildlife Surveying,” Remote Sensing, vol. 12, no. 5, pp. 891–908, 2020.

[9] S. J. K. Reddy and A. Kumar, “Poacher Detection using YOLO Algorithm,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 3, pp. 1–5, 2021.

[10] P. Gupta and R. Singh, “IoT-Enabled Real-Time Poaching Detection Using YOLOv5 and Edge Devices,” *SSRN*, 2025. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5358559

[11] A. Sharma *et al.*, “AI System to Protect Endangered Animal Population and Prevent Poaching Using YOLOv5 and LoRa,” *International Journal of Innovative Science and Research Technology*, vol. 8, no. 9, pp. 1–8, 2023.

[12] L. Wang *et al.*, “Intelligent Detection Method for Wildlife Based on Deep Learning,” *Sensors*, vol. 23, no. 24, 2023.

[13] S. Roy and M. Das, “WilDect-YOLO: An Efficient and Robust Computer Vision-Based Model for Endangered Wildlife Detection,” *Ecological Informatics*, vol. 60, 2020.

[14] Y. Tian, Q. Ye, and D. Doermann, “YOLOv12: Attention-Centric Real-Time Object Detectors,” arXiv preprint arXiv:2502.12524, 2025. [Online]. Available: https://arxiv.org/abs/2502.12524

[15] S. K. Shiragudikar, G. Bharamagoudar, K. K. Manohara, *et al.*, “Insight Analysis of Deep Learning and a Conventional Standardized Evaluation System for Assessing Rice Crop’s Susceptibility to Salt Stress during the Seedling Stage,” *SN Computer Science*, vol. 4, p. 262, 2023.

[16] S. Y. Malathi, G. R. Bharamagoudar, and S. K. Shiragudikar, “Diagnosing and Grading Knee Osteoarthritis from X-ray Images Using Deep Neural Angular Extreme Learning Machine,” *Proceedings of the Indian National Science Academy*, vol. 91, pp. 95–108, 2025.

[17] S. K. Shiragudikar, G. Bharamagoudar, M. K. K., M. S. Y., and G. S. Totad, “Predicting Salinity Resistance of Rice at the Seedling Stage: An Evaluation of Transfer Learning Methods,” in *Intelligent Systems in Computing and Communication (ISCComm 2023)*, CCIS, vol. 2231, Springer, Cham, 2025.

[18] S. Malathi, G. Bharamagoudar, S. K. Shiragudikar, and G. S. Totad, “Predictive Models for the Early Diagnosis and Prognosis of Knee Osteoarthritis Using Deep Learning Techniques,” in *Intelligent Systems in Computing and Communication (ISCComm 2023)*, CCIS, vol. 2231, Springer, Cham, 2025.

[19] S. K. Shiragudikar and G. Bharamagoudar, “Enhancing Rice Crop Resilience: Leveraging Image Processing Techniques in Deep Learning Models to Predict Salinity Stress of Rice during the Seedling Stage,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 14s, pp. 116–124, 2024.