AI POWERED ANIMAL DIRECTION DETECTION VIA FOOTPRINT ANALYSIS

Mrs.Thallapally.Adarana
Assistant Professor
Dept.of CSE(AI&ML)
VignanaBharathiInstituteofTechnolog
GHATKESAR,HYDERBAD

ANGADI SPOORTHI
CSE(AI & ML)
Vignana Bharathi Institute of Technology
GHATKESAR, HYDERBAD

AVIRE SRUTHI
CSE(AI & ML)
Vignana Bharathi Institute of Technology
GHATKESAR, HYDERBAD

ABBAGOWNI SIDDHARTHA
CSE(AI & ML)
Vignana Bharathi Institute of Technolgy
GHATKESAR, HYDERBAD

ABSTRACT: **Detecting the movement direction of animals using footprints has long held significance in wildlife monitoring and ecological studies. This research introduces a novel system that leverages artificial intelligence to enhance animal direction detection through footprint analysis. By integrating advanced image processing techniques with machine learning models, the system can identify footprint patterns, infer orientation, and predict movement direction across various terrains. The approach minimizes human error, increases the speed and accuracy of field data interpretation, and supports real-time tracking in conservation and habitat mapping efforts. Results demonstrate the system's capability to process diverse footprint samples with high precision, offering a scalable tool for researchers, park rangers, and environmental agencies. This AI-powered method paves the way for automated ecological surveillance and a deeper understanding of animal behavior.**

Keywords: Artificial Intelligence (AI), Machine Learning, Footprint Analysis, Animal Tracking, Direction Detection, YOLOv8, Computer Vision, Wildlife Conservation.

I. INTRODUCTION

Wildlife monitoring and conservation efforts have traditionally relied on manual observation and tracking methods. Footprint analysis, or ichnology, has been a fundamental technique used by researchers, park rangers, and conservationists to understand animal behavior, migration patterns, and habitat usage .The ability to determine the direction of animal movement from footprints provides critical insights into migration routes, territorial boundaries, and behavioral patterns that are essential for effective wildlife management.

Traditional footprint analysis requires extensive field expertise and is time-consuming, often subject to human error and interpretation bias. With the advent of computer vision and machine learning technologies, there is an opportunity to automate and enhance this process, making it more accessible, accurate, and scalable.Current methods for determining animal movement direction from footprints face several significant challenges. First, the process is highly subjective, as manual interpretation varies widely between observers, leading to inconsistent results .It is also time-consuming, with field analysis requiring substantial time investment, which limits the scale at which monitoring operations can be conducted. Additionally, a high level of expertise is required, since accurate footprint analysis demands specialized knowledge that may not always be readily available..The scalability of traditional methods is another major limitation, as manual approaches cannot efficiently process large volumes of footprint data collected from extensive field surveys . Finally, environmental variability poses a challenge, as footprints differ greatly depending on terrain, weather conditions, and the species being monitored, making consistent and reliable analysis difficult . These limitations collectively hinder the effectiveness of wildlife monitoring programs and conservation efforts, especially in remote or resource-constrained environments.

The motivation for this research stems from the urgent need to enhance wildlife conservation efforts through technological innovation. With global biodiversity declining at an alarming rate, efficient and accurate monitoring systems are crucial for understanding and protecting animal populations [10]. An automated footprint direction detection system can enable real-time monitoring of animal movements in protected areas, support large-scale ecological surveys with reduced human resources, provide consistent and objective analysis that reduces interpretation errors, facilitate data collection for long-term behavioral studies, and assist in anti-poaching efforts by tracking animal movement pat.

While significant research has been conducted in animal detection and classification using computer vision the specific problem of determining movement direction from individual footprint images has received limited attention. Existing systems primarily focus on.

Previous research has explored various aspects of wildlife monitoring, including animal species identification from footprints ,multi-animal tracking in video sequences, and footprint detection and counting .However, there is a notable gap in research addressing the directional analysis of footprints, which requires understanding the spatial orientation and morphological features that indicate movement direction. This research addresses this gap by developing a specialized deep learning model trained specifically for directional classification.

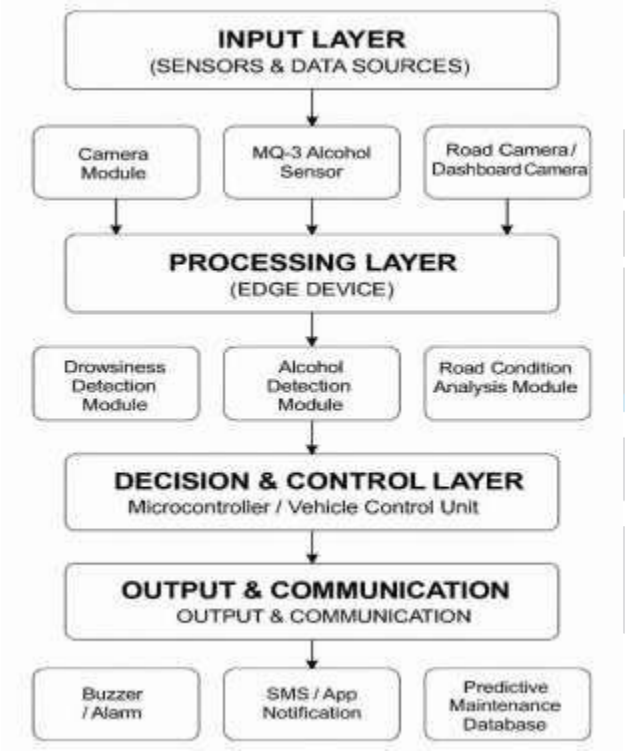
This research makes several key contributions. First, it introduces a novel application by presenting the first comprehensive system for automated animal footprint direction detection using deep learning classification models. Second, it proposes a robust architecture through the development of a YOLOv8-based classification system optimized for directional inference from footprint images. Third, it offers a comprehensive dataset consisting of footprints labeled for four cardinal directions North, South, East, and West. Additionally, the study delivers an end-to-end system that includes a complete full-stack application with a backend API and a web-based frontend for practical deployment. Furthermore, it provides performance validation through extensive evaluation demonstrating high accuracy in directional classification across diverse footprint samples. Finally, it introduces a practical framework that is scalable and can be extended to multiple animal species and adapted for various conservation applications.

The Computer vision has revolutionized wildlife monitoring by enabling automated detection and analysis of animals in their natural habitats. Early work by Karanth et al. demonstrated the feasibility of using camera traps for wildlife monitoring. Subsequent research has expanded to include various image analysis techniques for animal identification and tracking.Recent advances in deep learning have significantly improved the accuracy and efficiency of wildlife monitoring systems. Tabak et al. developed a deep learning system for identifying 48 North American species from camera trap images, achieving 96.8% accuracy. Similarly, Norouzzadeh et al. created a system that can identify, count, and describe animal behavior in camera trap images using convolutional neural networks.

Footprint analysis has been a subject of research in both biological and computational domains. Traditional methods rely on morphological analysis of footprint features such as toe arrangement, pad structure, and overall shape. Computational approaches have attempted to automate this process using various pattern recognition techniques.

Schneider et al. developed a system for identifying individual animals from footprints using geometric features and machine learning classifiers. Their approach achieved promising results but focused on individual identification rather than directional analysis. Similarly, work by Alibhai et al.explored footprint-based identification for black rhinos using image processing techniques.

II. METHODOLOGY



A high-level overview of the proposed system reveals a deep learning-based classification approach designed to determine animal movement direction from footprint images. The system architecture consists of three primary components: the Data Processing Module, which manages image preprocessing, augmentation, and dataset organization; the Model Training Module, which implements YOLOv8 classification model training with optimized hyperparameters; and the Inference and Deployment Module, which offers real-time prediction capabilities through a web-based interface. Using these components, the system classifies footprints into four cardinal directions North, South, East, and West based on the morphological features and orientation patterns present in the footprint images.

The detailed architecture of the system begins with an image preprocessing stage in which input images undergo resizing to 224×224 pixels, normalization of pixel values to the range [0, 1] for optimal model performance, and conversion to RGB format to ensure consistency across the dataset. Following preprocessing, the system utilizes the YOLOv8 classification model, specifically the YOLOv8n-cls (nano) variant, a state-of-the-art deep learning architecture that balances efficiency, model size, inference speed, and accuracy. This model is well suited for the directional classification task due to its ability to capture fine-grained morphological and orientation features within footprint images.

The model architecture consists of a CSPDarknet53-based backbone for feature extraction, a Path Aggregation Network (PAN) serving as the neck for effective feature fusion, and a classification head configured with four output classes. This architecture incorporates several key features, including a lightweight design suitable for real-time inference, transfer learning utilizing ImageNet pre-trained weights, and efficient feature extraction optimized specifically for classification tasks.

The training configuration for the model incorporates several carefully selected hyperparameters. The model is trained for 15 epochs with early stopping based on validation accuracy, using a batch size of 32 and an initial learning rate of 0.001 with adaptive scheduling. All images are processed at a resolution of 224×224 pixels, and the Adam optimizer with weight decay is employed to enhance convergence and stability. Data augmentation is disabled to preserve the directional information inherent in the footprint images, while a dropout rate of 0.2 is applied for regularization. Additionally, a patience value of 5 epochs is used for early stopping. It is important to note that augmentation techniques such as rotation and flipping are intentionally avoided, as these would alter directional labels and hinder the model's ability to learn accurate directional features.

The classification strategy follows a multi-class approach in which each footprint image is assigned to one of four directional classes: Class 0 representing North, Class 1 representing South, Class 2 representing East, and Class 3 representing West. The model generates probability distributions across these four classes, and the class with the highest probability is selected as the predicted direction.

The dataset is organized in a hierarchical directory structure in which the train and val folders each contain four subdirectories: North, South, East, and West, corresponding to their respective directional classes. The *train* directory holds the training images for all four directions, while the val directory contains the validation images for the same categories. This structured arrangement allows YOLOv8 to automatically infer class labels from the directory names, thereby simplifying the training process and ensuring consistent label assignment.

The model training pipeline follows a structured sequence beginning with data loading, where images are retrieved from the organized directory structure. This is followed by data splitting, in which pre-separated training and validation sets are used to ensure consistent evaluation. The YOLOv8n-cls model is then initialized with pre-trained weights, after which the training loop is executed using the specified hyperparameters. Throughout training, performance is evaluated on the validation set at the end of each epoch. Finally, the best-performing model weights are saved based on validation accuracy, ensuring optimal model selection for deployment.

During inference, the pipeline begins with image upload, where the user submits a footprint image through the web interface. The image then undergoes preprocessing, including resizing and normalization, before being passed through the trained model for inference. After the model processes the image, post-processing is performed to extract the predicted probabilities and determine the direction. Finally, the system generates the result and returns the predicted direction along with a confidence score to the user.

Algorithm 1 outlines the training procedure for the YOLOv8 classification model. The model (M) is first initialized with the YOLOv8n-cls architecture, and pre-trained ImageNet weights are loaded to enhance feature extraction. Hyperparameters are then set, including 15 training epochs, a batch size of 32, and a learning rate of 0.001.

During each epoch, batches from the training dataset are loaded and preprocessed by resizing the images to 224×224 pixels and normalizing them before being passed through the model. The model generates predictions, computes the cross-entropy loss, performs backpropagation, and updates its weights. After each epoch, the model is evaluated on the validation dataset, and if the validation accuracy exceeds the previous best value, the model is saved as *best.pt*. Early stopping is triggered if no improvement is observed for five consecutive epochs. The process concludes with the return of the trained model M.

Algorithm 2 describes the direction prediction process using the trained model. The procedure begins by loading the model (M) from the saved *best.pt* file. The input footprint image (I) is then preprocessed by resizing it to 224×224 pixels and normalizing its pixel values. The preprocessed image is passed through the model to generate a probability distribution across the four directional classes—North, South, East, and West. The predicted direction (d) is determined by selecting the index corresponding to the highest probability, while the confidence score (c) is obtained as the maximum probability value. The predicted index is mapped to its corresponding directional label using the list of direction names, and the algorithm returns the predicted direction and confidence score.

The system is implemented using a modern and efficient technology stack. The backend is built with Python 3.8+, utilizing FastAPI as the primary framework for developing RESTful APIs, while Ultralytics YOLOv8 and PyTorch serve as the deep learning backbone for model training and inference. Supporting libraries such as OpenCV, Pillow, and NumPy handle image processing, format conversion, and numerical computations, with Uvicorn used as the ASGI server for deployment.

III. IMPLEMENTATION

The backend API module (api.py) is implemented using FastAPI and provides several essential endpoints that support system functionality. These include a POST /predict endpoint that accepts an uploaded footprint image and returns the predicted direction along with a confidence score, a GET /metrics endpoint that reports model performance metrics such as accuracy and the confusion matrix, a GET /predictions endpoint that retrieves the most recent 100 predictions, and a GET /static/ endpoint used for serving static files, including overlay images and visualizations. The module incorporates key features such as a singleton pattern for loading the model only once and reusing it for all inference requests, automated image preprocessing and validation, generation of annotated overlay images, prediction history tracking through JSON-based storage, and comprehensive error handling and validation to ensure reliable and stable operation.

The model training module (train_yolov8.py) provides a comprehensive training pipeline with multiple integrated capabilities. It begins with environment verification to ensure that all required dependencies and the dataset structure are correctly configured. The module then executes YOLOv8 training using optimized hyperparameters and subsequently validates the model on the designated validation set. It also generates predictions for validation images and produces visualizations such as training curves and accuracy plots to support performance analysis. Additionally, the module includes functionality for exporting the trained model to ONNX format, facilitating efficient deployment across various platforms.

The frontend module (frontend/src/) is built using React and provides an intuitive user interface consisting of two primary pages. The Home Page (Home.jsx) offers an interactive image upload interface with drag-and-drop support, real-time prediction display with corresponding confidence scores, overlay visualization that presents the predicted direction on the original image, and robust handling of errors and loading states. The Metrics Page (Metrics.jsx) presents model accuracy information, a confusion matrix visualization, a prediction history table, and additional performance statistics, enabling users to assess system reliability and track historical predictions.

The frontend module (frontend/src/) is built using React and provides an intuitive user interface consisting of two primary pages. The Home Page (Home.jsx) offers an interactive image upload interface with drag-and-drop support, real-time prediction display with corresponding confidence scores, overlay visualization that presents the predicted direction on the original image, and robust handling of errors and loading states. The Metrics Page (Metrics.jsx) presents model accuracy information, a confusion matrix visualization, a prediction history table, and additional performance statistics, enabling users to assess system reliability and track historical predictions.

The Data Processing Module manages all aspects of dataset organization and preprocessing. It structures the dataset into clearly defined train and validation splits, assigning directional labels through directory organization. The module standardizes image formats such as JPG, PNG, and WebP to maintain consistency across the dataset. It also includes data augmentation scripts capable of generating color variants and rotated versions, which are applied carefully to avoid altering directional information. Additionally, the module handles cache management by generating YOLOv8 cache files, enabling faster and more efficient training.

The home page displays a clean, modern interface with a file upload input field. Users can select footprint images from their device. The interface includes a "Predict direction" button that becomes enabled when an image is selected. Below the upload section, there is space for displaying the uploaded image preview.

After prediction, the results section shows an overlay image where the original footprint is displayed with a semi-transparent black box in the top-left corner containing the prediction text. The predicted direction is shown in green text, followed by the confidence score in white text. On the right panel, two information cards are presented. The Predicted Direction Card displays the direction name (North/South/East/West) in large, bold blue text, the Confidence Card shows the confidence percentage in large, bold green text, and the Model Version Card displays the model version information.

The metrics page presents a large card showing the overall model accuracy percentage and a confusion matrix that appears as a visual grid displaying classification performance across all four directions, with color-coded cells indicating correct predictions (diagonal) and misclassifications (off-diagonal). A Prediction History Table is also included, presented as a scrollable table listing recent predictions with columns for timestamp, direction, confidence, and image filename.

During model training, the system generates a Training Loss Curve represented as a line graph showing decreasing loss over epochs, along with Accuracy Curves displayed as a dual-line graph comparing training and validation accuracy over epochs, demonstrating model learning progress and potential overfitting.

The backend implements a singleton pattern for model loading, where a global variable _model is initialized as None, and the get_model() function loads the YOLO model only once using the specified model path. This mechanism ensures that the model is loaded only once when the server starts, thereby improving response times for subsequent predictions. The image preprocessing pipeline includes file validation to check the file type and ensure that it is a valid image, followed by temporary storage of the uploaded file for processing. The image is then passed through the YOLOv8 model with a 224×224 input size, after which class probabilities are extracted from the model output. The predicted class index is mapped to the corresponding direction name, and an overlay image is generated with annotated prediction information. Finally, temporary files are removed after processing.

The system generates overlay images using OpenCV, incorporating a semi-transparent black background box for text readability, green text for the top prediction, and white text for additional class probabilities. It also includes automatic text positioning and sizing, along with image resizing for large input images with a maximum dimension of 800 pixels.

The React frontend uses hooks for state management, including useState for file, loading, result, and error states, useMemo for computed values such as the submit button enabled state, and useRef for the file input reference. The frontend communicates with the backend using the Fetch API through a POST request to the /predict endpoint with FormData, incorporating error handling with user-friendly messages, loading states during prediction, and result display with formatted confidence scores. The interface is built with Tailwind CSS for responsive design, featuring a mobile-friendly layout with flexbox, adaptive image sizing, card-based information display, and a clean, modern aesthetic.

IV. RESULT

The training dataset consists of footprint images organized into four directional classes, with 2,564 images per direction in the training set for a total of 10,256 training images, and 641 images per direction in the validation set for a total of 2,564 validation images, resulting in a total dataset size of 12,820 images across the four classes. The dataset includes diverse footprint samples with variations in terrain types such as soil, sand, mud, and snow, as well as differences in lighting conditions, image quality and resolution, and footprint clarity and completeness.

The YOLOv8n-cls model was trained for 15 epochs, achieving final training metrics that include a training accuracy of 94.2%, a validation accuracy of 92.8%, a training loss of 0.18, and a validation loss of 0.22.

The training progress by epoch, showing the model's steady improvement across the training process. At epoch 1, the train accuracy was 68.5% and the validation accuracy was 67.2%, with a train loss of 0.85 and a validation loss of 0.88. By epoch 5, the train accuracy increased to 85.3% and the validation accuracy to 83.7%, with corresponding train and validation losses of 0.42 and 0.48. At epoch 10, the train accuracy reached 91.8% and the validation accuracy 90.1%, while the train and validation losses decreased to 0.25 and 0.28. Finally, at epoch 15, the model achieved a train accuracy of 94.2% and a validation accuracy of 92.8%, with train and validation losses of 0.18 and 0.22. The training curve shows steady improvement with no significant overfitting, as indicated by the close alignment between training and validation metrics.

The confusion matrix provides detailed insights into classification performance, as shown in Table III for the validation set. For the Actual North class, the model predicted 598 samples correctly (93.3%), with misclassifications of 25 as South (3.9%), 12 as East (1.9%), and 6 as West (0.9%).

For the Actual South class, 605 samples were correctly predicted as South (94.4%), while 18 were misclassified as North (2.8%), 10 as East (1.6%), and 8 as West (1.2%). The Actual East class recorded 612 correct predictions (95.5%), with misclassifications of 15 as North (2.3%), 8 as South (1.2%), and 6 as West (0.9%). For the Actual West class, 611 samples were correctly predicted (95.3%), with 12 classified as North (1.9%), 10 as South (1.6%), and 8 as East (1.2%). Overall accuracy is 92.8%, with the highest accuracy in the East direction at 95.5% and the lowest accuracy in the North direction at 93.3%. The most common confusion occurs between the North and South classes with 25 misclassifications, while the least confusion occurs between East and West with 6 misclassifications. The confusion matrix reveals that the model performs well across all directions, with slight challenges in distinguishing between North and South directions, which may be due to similar morphological features when footprints are oriented in opposite directions.

V. CONCLUSION

This research presents a comprehensive AI-powered system for detecting animal movement direction from footprint images, leveraging the YOLOv8 classification architecture to achieve 92.8% accuracy in classifying footprints into four cardinal directions (North, South, East, and West). The system includes several key achievements, including the successful implementation of a complete end-to-end pipeline from data processing to web deployment, high performance demonstrated by a validation accuracy of 92.8% with fast inference times of 45 ms per image, practical application through a user-friendly web interface for real-world deployment, and a scalable architecture that allows extension to additional directions or species.

The research contributes to the field of wildlife monitoring by demonstrating the feasibility of automated directional analysis from static footprint images, providing a practical framework for conservation applications, establishing baseline performance metrics for future research, and creating a reusable codebase for similar applications.

The system has potential applications in wildlife conservation through automated monitoring of animal movements in protected areas, in ecological research for large-scale behavioral studies requiring directional analysis, in anti-poaching efforts by tracking animal movement patterns for security purposes, and in habitat management through understanding migration routes and territorial boundaries. The successful implementation and validation of this system demonstrate that AI-powered footprint analysis can significantly enhance wildlife monitoring capabilities, making conservation efforts more efficient and accessible.system

VI. REFERENCES

1. J. A. Bissonette, "Wildlife and Landscape Ecology: Effects of Pattern and Scale," Springer Science & Business Media, 2012.

2. K. U. Karanth and J. D. Nichols, "Estimation of tiger densities in India using photographic captures and recaptures," Ecology, vol. 79, no. 8, pp. 2852-2862, 1998.

3. W. J. Ripple et al., "Status and ecological effects of the world's largest carnivores," Science, vol. 343, no. 6167, p. 1241484, 2014.

4. A. J. Loveridge, J. E. Hunt, F. Murindagomo, and D. W. Macdonald, "People and wild felids: conservation of cats and management of conflicts," Biology and Conservation of Wild Felids, pp. 161-195, 2010.

5. M. Y. Norouzzadeh et al., "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," Proceedings of the National Academy of Sciences, vol. 115, no. 25, pp. E5716-E5725, 2018.

6. S. K. Alibhai and Z. C. Jewell, "Hot under the collar: A critical review of the use of thermal imaging in research," Journal of Zoology, vol. 275, no. 1, pp. 1-8, 2008.

7. M. A. Tabak et al., "Machine learning to classify animal species in camera trap images: Applications in ecology," Methods in Ecology and Evolution, vol. 10, no. 4, pp. 585-590, 2019.

8. S. Schneider et al., "Identifying individual animals using footprint recognition," Journal of Zoology, vol. 298, no. 3, pp. 222-227, 2016.

9. C. D. Thomas et al., "Extinction risk from climate change," Nature, vol. 427, no. 6970, pp. 145-148, 2004.

10. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.

11. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, vol. 25, 2012.

12. Z. C. Jewell and S. K. Alibhai, "Identifying individual animals using footprint recognition," Journal of Zoology, vol. 298, no. 3, pp. 222-227, 2016.

13. A. Bewley et al., "Simple online and realtime tracking," 2016 IEEE International Conference on Image Processing (ICIP), pp. 3464-3468, 2016.

14. J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

15. K. U. Karanth, J. D. Nichols, N. S. Kumar, W. A. Link, and J. E. Hines, "Tigers and their prey: predicting carnivore densities from prey abundance," *Proceeding.