Specimen Detection Using Voice and Footprint Images in Deep Learning

Roopa B S

Chapter 1

INTRODUCTION

1.1 Domain

Artificial Intelligence (AI) is a rapidly evolving domain that enables machines to simulate human intelligence, including perception, decision-making, and learning. Within AI, deep learning—a subset of machine learning—has revolutionized pattern recognition tasks by leveraging neural networks to automatically extract and learn features from complex data. In the context of biodiversity research and wildlife monitoring, AI plays a crucial role in identifying species through non-invasive means. Our project harnesses deep learning techniques to identify bird and animal species based on two key biometric inputs: voice (audio) and footprint images. This dual-modal approach enhances accuracy and reliability, enabling automated, scalable, and efficient species recognition for conservation, ecological studies, and environmental monitoring.

1.2 Introduction

This document details the development of a Flask-based web application designed to identify bird and animal species using two distinct types of input: audio recordings and foot images. The application leverages the power of Convolutional Neural Networks (CNNs) to classify species based on uploaded audio spectrograms and foot images. The system aims to provide real-time prediction of species and display the classification result along with a confidence score through a user-friendly web interface. This project integrates various technologies, including Flask for the web server, librosa or matplotlib for audio processing, OpenCV for image preprocessing, and TensorFlow/Keras for implementing the deep learning models.

The identification of biological species has long been a crucial task in various scientific and practical domains. Traditionally relying on manual observation and expert knowledge, recent advancements in machine learning, particularly deep learning, have opened new avenues for automated species recognition. This report outlines the development of a Flask-based web application designed to identify bird and animal species using two distinct modalities: audio recordings and foot images. The primary objective of this project is to create a user-friendly system that leverages Convolutional Neural Networks (CNNs) to classify species based on uploaded audio spectrograms and foot images. This application aims to meet specific functional and technical requirements, including real-time prediction, a web-based interface for file uploads, and the utilization of pre-trained deep learning models. Furthermore, the project scope encompasses the creation of 25 distinct models for both audio and image classification, alongside the integration of bonus features such as data visualizations and scalability considerations. The structure of this report will follow the development lifecycle of the application, from high-level architecture to detailed implementation plans for each module, culminating in deployment considerations and potential future enhancements. The combination of audio and image analysis in a single application presents an opportunity to explore how different sensory inputs can contribute to species identification.

1.3 Existing System

The existing system is a Flask-based web application developed to identify bird and animal species through the analysis of audio recordings or footprint images. In this system, users upload high-quality audio recordings captured via microphones or footprint images taken with high-resolution cameras or pressure-sensitive mats. Once uploaded, the system performs preprocessing tasks—audio data undergoes noise reduction and is converted into spectrograms such as Mel-spectrograms or MFCCs, while footprint images are normalized, enhanced, and processed to remove background noise. The preprocessed data is then fed into deep learning models for feature extraction: audio inputs are analyzed using CNNs combined with LSTM or Transformer-based models like Wav2Vec 2.0 or DeepSpeech, and footprint images are processed using CNN architectures such as ResNet or EfficientNet to identify spatial features. To enhance prediction accuracy, the system incorporates multimodal fusion strategies, including early fusion (feature-level), late fusion (decision-level), or hybrid fusion methods. These fused features are passed through fully connected layers for final classification, which yields a predicted species or specimen ID along with a confidence score. Additionally, the system can optionally cross-verify results with a biometric database. Finally, it presents the predicted species along with detailed information sourced from a structured database, providing an efficient and user-friendly solution for wildlife species identification and research support.

Disadvantages

1. Data Quality and Availability

- Voice:
  - Susceptible to noise, accents, health conditions, and emotional state.
  - Requires high-quality recordings for reliable performance.
- Footprint:

- Needs clean, high-resolution images.
- Variability due to surface texture, shoe type, or walking style.

2. Dataset Limitations

- Lack of large, diverse, publicly available datasets, especially for footprint images.
- Hard to find matched voice and footprint data for the same subjects.

3. High Computational Cost

- Deep learning models for image and audio processing can be resource-intensive.
- Real-time detection systems require significant hardware.

4. Vulnerability to Spoofing or Forgery

- Voice can be mimicked or synthesized (deepfakes).
- Footprints can be faked using molds or printed patterns.

5. Ethical and Privacy Issues

- Collecting and storing biometric data raises legal and privacy concerns.
- Consent and data protection (GDPR, HIPAA) must be handled carefully.

6. Limited Generalization

- Models may perform poorly on unseen users or conditions not present in training data.
- Adaptation to environmental changes (e.g., noise, different walking surfaces) is challenging.

7. Sensor Dependence

- Accuracy heavily depends on the quality and type of sensors (microphones, cameras, pressure plates).

1.4Objective

1. Develop a Multimodal Detection System

- Design a system that uses both voice and footprint images to detect or identify specimens (humans or animals) more accurately than unimodal systems.

2. Enhance Detection Accuracy and Robustness

- Leverage deep learning to extract and fuse discriminative features from voice and footprint data for high-accuracy detection under varied environmental conditions.

3. Reduce False Positives/Negatives

- Combine biometric modalities to minimize errors in detection and increase reliability in real-world scenarios.

4. Ensure Real-Time or Near-Real-Time Processing

- Optimize the system architecture to allow for efficient, fast processing, suitable for real-time applications like surveillance or wildlife monitoring.

5. Improve Resistance to Spoofing and Tampering

- Utilize multimodal data to make the system more secure and difficult to deceive, as attackers would need to fake both voice and footprint data simultaneously.

6. Enable Scalability and Adaptability

- Create a system that can scale to large datasets and adapt to new specimens over time using continuous or incremental learning techniques.

7. Create a Dataset for Training and Evaluation

- Build or curate a comprehensive, labeled dataset of voice and footprint images for training and validating the system.

8. Support Diverse Application Scenarios

- Ensure the system is applicable to multiple domains, such as biometric authentication, forensic science, smart surveillance, and animal tracking.

1.5Proposed System

The proposed system is a Flask-based web application designed to identify bird and animal species through either audio recordings or images of footprints. Users can upload high-quality audio recorded via microphones or high-resolution footprint images captured using cameras or pressure-sensitive mats. Upon uploading, the system initiates a preprocessing phase, where audio undergoes noise reduction and is converted into spectrograms such as Mel-spectrograms or MFCCs. Meanwhile, footprint images are normalized, enhanced, and cleared of background noise to focus on relevant features. Following preprocessing, feature extraction is conducted using advanced deep learning models: audio inputs are processed through a CNN combined with an LSTM or Transformer-based architecture like Wav2Vec 2.0 or DeepSpeech, while footprint data is analyzed using powerful CNNs such as ResNet or EfficientNet to detect spatial patterns.

To improve prediction accuracy, the system employs a multimodal feature fusion strategy. This could involve early fusion, where features from both audio and footprint data are concatenated before classification; late fusion, where predictions from individual models are combined; or hybrid fusion, which integrates both feature and decision-level information. Once fused, these features are passed through fully connected layers for final classification, resulting in a predicted species or specimen ID. The system also outputs a confidence score indicating the reliability of the prediction and can optionally cross-check results against a biometric database to enhance verification. All predictions are supplemented with detailed species information retrieved from a structured data source. This application aims to deliver an efficient, user-friendly, and accurate tool for wildlife identification and research support.

Advantages

1. Increased Accuracy Through Multimodal Fusion

- Combining voice and footprint data reduces the chance of misidentification.
- Helps mitigate weaknesses in one modality using the strengths of the other.

2. Robustness to Environmental Variability

- If one input (e.g., voice) is noisy or distorted, the system can still perform using the footprint input.

3. Enhanced Security

- Harder to spoof both voice and footprint at the same time.
- More resistant to impersonation attacks compared to unimodal systems.

4. Scalability and Real-Time Capability

- Optimized deep learning models (e.g., MobileNet, quantized CNNs) can allow real-time performance on edge devices.

5. Wide Applicability

- Suitable for:
    - Human identification (biometric security, surveillance).
    - Wildlife monitoring (animal tracking via vocalization and paw prints).
    - Forensics and law enforcement.

6. Adaptive Learning

- With continual learning techniques, the system can adapt to new specimens or changing conditions over time.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

A literature survey is a systematic review of existing research and studies related to a specific field of interest. It plays a crucial role in understanding the current state of knowledge, methodologies, and tools used by previous researchers. In the context of our project—Bird and Animal Species Identification using Voice and Footprint Images in Deep Learning—the literature survey helps us explore how similar problems have been addressed. It covers studies involving audio signal processing, image recognition, and biometric species identification. The survey also examines various deep learning models such as CNNs and RNNs used in related applications. Through this, we identify limitations in current approaches, such as accuracy issues, lack of multi-modal data, or limited datasets. These insights help us refine our objectives and adopt improved methods. The literature survey also ensures that our work is relevant, novel, and contributes meaningfully to the field of AI in wildlife monitoring.

2.2 Survey Papers

1. TITLE: Birds Species Identification Using Deep Learning Model
YEAR: 2024
AUTHORS: Ms. K. Sherin M.E., Ms. A.R. Darshika Kelin M.E., Surendar Senthilvelan
Summary:"The model is trained on a preprocessed dataset containing various species like owls, robins, eagles, and unidentified birds. The system aims to enhance current identification methods by improving classification accuracy. It evaluates performance using a confusion matrix to ensure precise results. This approach supports broad and accurate species recognition in diverse environments.

2. TITLE: Design of Bird Sound Recognition Model Based on Lightweight
YEAR: 2022
AUTHORS : Fan Yang, Ying Jiang, and Yue Xu
Summary: proposed a lightweight bird sound recognition model using MobileNetV3 to improve efficiency and generalization. The model uses a large dataset of 264 bird species and enhances feature extraction through depth wise separable convolutions and a multi-scale feature fusion structure. It incorporates Pyramid Split Attention (PSA) and channel attention mechanisms for better spatial and channel information extraction. The model is optimized for performance while reducing complexity and computational cost.

3. TITLE: Bird Species Recognition using Deep Learning
YEAR: 2023
AUTHOR: Hari Kishan Kondaveeti and team
Summary: developed an Arduino-based bird species recognition system combining hardware and deep learning. It uses a PIR motion sensor and ESP-32 camera to capture bird images, which are uploaded to Google Drive. These images are then analyzed by a trained deep learning model to identify bird species. The system aids researchers and enthusiasts in recognizing birds in specific regions.

4. TITLE: AudioProtopnet: An Interpretable Deep Learning Model for Bird Sound Classification
YEAR: 2024
AUTHOR: René Heinrich and colleagues

Summary: proposed AudioProtoPNet, an interpretable deep learning model for multi-label bird sound classification. It uses a ConvNeXt backbone and prototype learning to identify bird vocalizations from spectrograms. Unlike black-box models, it provides clear explanations by comparing audio to learned prototypes. This approach supports both accurate classification and deeper insight for researchers and engineers.

5. TITLE: Foot Prints Image Based Animal Species Classification using PNN
YEAR: 2024
AUTHOR: Mrs.Simran Mansoori, Mr. Rajneesh Pachouri, Mr. Anurag Jain
Summary: In this research work, issues related in development of an effective and efficient animal detection, classification and tracking system are addressed. Successful attempts towards development of algorithmic models for animal segmentation, animal detection, classification and tracking are made.

6. TITLE: Identifying endangered species from footprints
YEAR: 2012
AUTHOR: Zoe Jewell, Sky K. Alibhai
Summary: A technique that analyzes images of animal footprints provides reliable data on endangered wildlife populations and individuals. To protect endangered species and understand extinction threats, we need effective monitoring techniques. Biologists have documented around one million species, which represent only 1–10% of all those on earth. Less quantified, however, is how human activity elevates local extinction rates, particularly in vulnerable endemic populations.

7. TITLE: Bird Species Identification from Audio Data
YEAR: 2023
AUTHOR: Ching Seh Wu, Sasanka Kosuru, Samaikya Tippareddy
Summary: Birds are an important indicator species for environmental changes, and identifying bird species can provide valuable insights into changes in their populations and habitats. This research focuses on identifying bird species using audio recordings from the BirdCLEF dataset on Kaggle. Using these audio recordings, we extracted 26 acoustic features like MFCC, spectral centroid, spectral bandwidth, etc.

8. Title: Audio-based Bird Species Identification with Deep Convolutional Neural Networks
YEAR: 2018
AUTHOR: Mario lasseck
Summary: A system for audio-based bird identification has proven to be particularly useful for biodiversity monitoring and education. It can help professionals working with bioa-coustic sounds by processing large

audio collections in a fast and automated way. Or help amateurs to identify birds in self-made recordings. The Life CLEF bird identification task challenges participants to identify different bird species in a large collection of audio recordings provided by Xeno-Canto.

9.TITLE: A Novel Bird Sound Recognition Method Based on Multifeature Fusion and a Transformer Encoder

YEAR: 2023

AUTHOR: Shaokai Zhang 1, Yuan Gao 1ORCID, Jianmin Cai 1, Hangxiao Yang 2, Qijun Zhao 2ORCID and Fan Pan

Summary: Birds play a vital role in the study of ecosystems and biodiversity. Accurate bird identification helps monitor biodiversity, understand the functions of ecosystems, and develop effective conservation strategies. However, previous bird sound recognition methods often relied on single features and overlooked the spatial information associated with these features, leading to low accuracy. Recognizing this gap, the present study proposed a bird sound recognition method that employs multiple convolutional neural-based networks and a transformer encoder to provide a reliable solution for identifying and classifying birds based on their unique sounds.

CHAPTER 3

REQUIREMENT SPECIFICATION

The Software Requirement Specification (SRS) for the project "Bird and Animal Species Identification Using Footprint Image and Audio in Deep Learning" outlines the objectives, features, and technical requirements of the system. The aim of the project is to develop a deep learning-based application that can accurately identify bird and animal species using either footprint images or audio recordings. Users will be able to upload footprint images or bird/animal sounds through a user-friendly interface. Once the data is uploaded, it will be preprocessed—images will undergo normalization and filtering, while audio recordings will be converted into spectrograms to extract visual features for analysis.

The system will utilize Convolutional Neural Networks (CNNs) as the core deep learning model for both image and audio-based classification. CNNs are effective in extracting spatial features from images and spectrograms, making them suitable for this dual-input identification task. After processing, the model will output the predicted species name along with a confidence score, helping users quickly and accurately identify wildlife based on limited data. The system will also include a feature to store and retrieve previous identification results.

3.1 Functional Requirements

This system is designed to accurately identify bird and animal species using image and audio inputs provided by users. It incorporates deep learning techniques to process, classify, and return results through an accessible web interface. The following are the core functional capabilities of the application:

Functional Requirement 1: Input Handling

Voice Input: The system must allow the user to record or upload voice samples in formats like WAV or MP3.

Footprint Image Input: Users should be able to upload clear images of footprints in standard image formats (JPG, PNG).

Validation: The system will check file types, sizes, and clarity to ensure usable data.

Functional Requirement 2: Image and Audio Preprocessing

Audio Preprocessing:

Noise reduction, normalization, and silence trimming.

Conversion to features such as MFCC (Mel-Frequency Cepstral Coefficients) or spectrograms.

Image Preprocessing:

Image resizing, grayscale conversion, and noise filtering.

Feature enhancement via techniques such as edge detection and thresholding to highlight footprint contours.

Functional Requirement 3: Specimen Classification

Feature Fusion: The system will combine voice and image features into a multimodal representation.

Deep Learning Model: A trained model (e.g., CNN + LSTM or Transformer-based) will process the features to identify the specimen.

Output: The system will return a predicted label (e.g., individual ID, species) with a confidence score.

Functional Requirement 4: User Interface (UI)

Input Interface: A simple UI for users to upload voice and image files.

Result Display: A section showing classification results, including the predicted label and confidence.

History/Logs: Optional feature for tracking past results or exporting data.

Feedback Option: Users can verify or correct the system's prediction to support model improvement

3.2 Non-Functional Requirement:

Non-functional requirements define how a system performs rather than what it does. They focus on aspects like speed, reliability, usability, security, scalability, and maintainability. These requirements ensure the system is efficient, user-friendly, and robust under various conditions. While they do not directly affect the system's core functionality, they significantly influence the user experience and overall system quality.

Non-Functional Requirement 1: Performance and Response Time

The system should deliver fast and responsive processing for an optimal user experience.

- Classification results should be returned within 5 seconds on average.
- Efficient handling of large audio/image files through optimized processing pipelines.
- Use of background processing or multithreading to prevent UI delays.
- Caching frequently used results to reduce redundant computations.

Non-Functional Requirement 2: Availability and Reliability

The application must remain accessible and function consistently with minimal interruptions.

- Maintain 99% uptime in live environments.
- Include automated recovery mechanisms for unexpected failures.
- Scheduled maintenance should be minimal and well-communicated.
- Implement system logs for error tracking and diagnostics.

Non-Functional Requirement 3: Usability and Interface Design

The interface should be intuitive, visually clear, and accessible to all types of users.

- Design a clean, minimal UI that supports ease of navigation.
- Ensure mobile responsiveness and cross-platform compatibility.
- Include visual feedback during operations (e.g., loading spinners).
- Provide help sections or tooltips for guiding users.

Non-Functional Requirement 4: Security and Data Privacy

The system must secure all user data and ensure privacy compliance.

- Use HTTPS for secure communication between client and server.
- Uploaded files should be processed temporarily and deleted unless permission is granted.
- Implement data encryption for sensitive storage if used.
- Follow data regulations such as GDPR or local data privacy standards.

Non-Functional Requirement 5: Scalability and Maintainability

The system should grow with demand and be easy to manage or enhance over time.

- Support horizontal and vertical scaling as user load increases.
- Design with modular architecture for ease of updates and debugging.
- Allow seamless integration of new features, models, or datasets.
- Ensure clear documentation for developers and maintainers.

Non-Functional Requirement 6: Compatibility and Portability

The system should work across different platforms and be deployable in varied environments.

- Ensure compatibility with common web browsers (Chrome, Firefox, Edge, Safari).
- The backend should support deployment on cloud or local servers.
- Use platform-independent technologies (e.g., Python, Docker) for portability.
- Provide installation/setup guides for different environments.

3.3 System Requirement

The external interface requirements for this system include a user-friendly web interface that allows seamless image and audio file uploads. It should integrate with cloud storage services for storing and retrieving data, such as Google Drive. The system must also interact with deep learning models via an API to process the inputs and return the classification results. Additionally, it should support common web browsers to ensure accessibility for users across different platforms.

Hardware requirement

- Processor: Intel i5 7500
- RAM: 4 GB & above
- Cache Memory: 512 KB
- Hard Disk: 50 GB

Software requirement

- Operating System: Windows 7 & above
- NLP-Technology: TextBlob
- Front-End: HTML, CSS, JavaScript, Python
- Back-End: MySQL

- Web Server: Flask

Introduction to SRS

A Software Requirement Specification (SRS) is a comprehensive document that outlines the complete behavior, features, and constraints of a software system. It serves as a formal agreement between stakeholders, including clients, developers, and project managers, defining what the software will do and how it is expected to perform. The SRS includes both functional requirements (specific functionalities the system must provide) and non-functional requirements (performance, usability, security, etc.). It also specifies system interfaces, data handling, and operational scenarios. By providing a clear and structured framework, the SRS minimizes misunderstandings and sets a solid foundation for design, development, testing, and maintenance. It plays a crucial role in project planning, ensuring the software meets user expectations and business goals. A well-prepared SRS helps prevent scope creep, reduces errors, and enhances communication throughout the development lifecycle.

Purpose

The purpose of a Software Requirement Specification (SRS) is to clearly document the functionalities, behavior, and constraints of a software system. It ensures that all stakeholders, including developers, clients, and testers, have a shared understanding of the system's expectations. The SRS guides the development process, helps prevent misunderstandings, and serves as a basis for system testing and validation. It also acts as a contract between stakeholders, establishing agreed-upon deliverables. Additionally, the SRS supports future maintenance and enhancements by providing a reference for updates. Ultimately, it ensures that the system meets user needs and project objectives.

Flask

Flask is a lightweight and flexible web framework for Python, designed to help developers build web applications quickly and easily. Unlike more heavyweight frameworks, Flask provides just the essentials, allowing for high customization and extensibility based on project needs. It features simple routing, where URLs are mapped to specific functions, and integrates seamlessly with databases, external APIs, and machine learning models. Flask supports secure communication with HTTPS, ensuring safe data transmission. It also includes tools for session management, making it easier to track user data across different requests. Its minimalistic nature and simplicity make it an ideal choice for small to medium-sized projects, like our species identification system.

Minimalistic and Lightweight: Flask is a lightweight framework that provides only the essential tools needed to build web applications. It doesn't come with unnecessary built-in features, allowing for flexibility in how you structure the project. This minimalism makes it ideal for projects like ours, where we need to focus on specific tasks, such as handling image and audio uploads, without added complexity. Flask's simplicity allows for rapid development and easy integration with other services, such as machine learning models. This results in a cleaner, faster application that's easy to maintain.

Routing: Flask's routing system is designed to map URLs to specific Python functions, making it easy to define and handle various user interactions. In our project, Flask's routes will be responsible for handling requests like uploading files (audio or images), triggering the classification process, and rendering the results. Each route can be linked to a specific function that processes the input and returns the appropriate response. This flexibility makes it simple to manage different aspects of the web application, from processing data to displaying results. Flask's dynamic routing ensures smooth navigation within the app.

Templating: Flask uses the Jinja2 templating engine to render dynamic HTML pages. Templating allows us to create reusable, dynamic content that can display results, such as predicted species names, confidence scores, and detailed information. The templates are written in HTML but can include dynamic elements like variables, loops, and conditions. This makes it easier to generate personalized content based on the user's interaction, ensuring a more engaging experience. Flask's templating engine helps streamline the front-end by separating logic from presentation, which is ideal for our user interface.

HTTPS Request Handling: Flask supports secure HTTPS communication, ensuring that data transmitted between the client and server is encrypted. In our project, this is crucial for handling sensitive user data, such as uploaded audio recordings or footprint images. By enabling HTTPS, we protect users' privacy and ensure that their data is securely transmitted during the species identification process. Flask makes it straightforward to configure SSL/TLS certificates to enforce HTTPS, promoting a secure connection. This added layer of security is vital, especially for applications dealing with user-submitted content.

Session Management: Flask provides built-in session management tools to track and store user-specific data across requests. This feature is useful for maintaining user progress, such as tracking uploaded files, processing states, or keeping temporary results. In our project, Flask sessions can store the uploaded data for predictions, allowing the user to receive results or perform additional queries without re-uploading the files. Flask's session handling also ensures that users don't lose their progress due to session timeouts or page reloads. This feature improves the overall user experience, ensuring smooth and continuous interaction with the application.

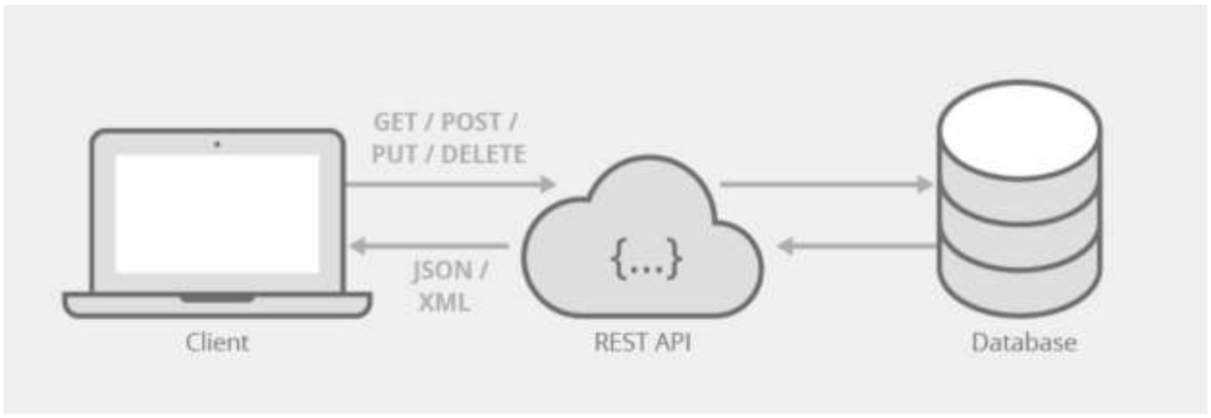


Fig 3.1: Software Architecture

Python

Python is a high-level, interpreted programming language known for its simplicity, readability, and clean syntax. It supports multiple programming paradigms, including object-oriented, procedural, and functional programming. Python comes with a vast standard library, making it easy to handle tasks like file operations, web development, and data processing. It is cross-platform and runs seamlessly on Windows, macOS, and Linux. Python is widely used in fields like data science, machine learning, web development, and automation. Its strong community support and rich ecosystem of libraries (like NumPy, pandas, TensorFlow, and Flask) enhance its versatility. Python's easy learning curve makes it ideal for both beginners and professionals.

Simplicity and Readability: Python is known for its clear, easy-to-read syntax that closely mirrors natural language, making it an excellent choice for beginners. It emphasizes readability and reduces the complexity of programming by using indentation instead of braces to define code blocks. This simplicity promotes clean and maintainable code. Developers can focus more on solving problems than on understanding complex syntax.

Extensive Standard Library: Python comes with a comprehensive standard library that includes modules for file I/O, web development, data manipulation, and more. These built-in libraries eliminate the need to reinvent the wheel for common tasks, allowing developers to quickly implement features. The library's extensive scope makes Python versatile and useful for a wide range of applications. It significantly speeds up the development process by providing reusable code.

Machine Learning Capabilities: Python has become the go-to language for machine learning, thanks to libraries like **TensorFlow**, **Keras**, **Scikit-learn**, and **PyTorch**. These tools offer powerful frameworks for data preprocessing, training, and model evaluation. Python's simple syntax and strong ecosystem make it easy to implement machine learning algorithms with minimal code. It also supports deep learning models, data analysis, and visualization seamlessly.

Libraries

TensorFlow is an open-source deep learning framework developed by Google that supports large-scale machine learning and numerical computation. In our project, TensorFlow served as the foundational backend for executing model training, evaluation, and prediction tasks. Its robustness and flexibility made it ideal for managing the complex operations required in processing both audio and image data. TensorFlow's compatibility with various deployment tools also enabled seamless integration into the Flask-based web application.

Keras is a high-level neural network API built on top of TensorFlow, designed for easy and fast prototyping of deep learning models. In this project, Keras was used to construct and train convolutional neural networks (CNNs) for classifying bird and animal species based on voice recordings and footprint images. Its user-friendly syntax and pre-built components allowed us to quickly implement and fine-tune models, improving development efficiency and overall model performance.

Librosa is a Python library for analyzing and processing audio signals. In this project, Librosa was used to extract features such as Mel spectrograms and MFCCs from bird and animal sounds, which are essential for training the audio classification model. Its powerful audio processing capabilities helped convert raw audio files into a format suitable for deep learning models.

NumPy is a fundamental Python library for numerical computing and array manipulation. It played a key role in handling and transforming large datasets, especially when preparing image and audio data for training and prediction. NumPy arrays were also used to store intermediate data such as spectrograms, image pixel values, and model outputs efficiently.

OpenCV (Open Source Computer Vision Library) is widely used for real-time image processing tasks. In this project, OpenCV was utilized for preprocessing footprint images, including resizing, grayscale conversion, thresholding, and normalization. These steps enhanced the clarity and consistency of visual inputs before feeding them into the CNN model.

Matplotlib is a Python plotting library used to create static, interactive, and animated visualizations. In our system, Matplotlib was used to generate and display spectrograms of the uploaded audio files, helping to visualize sound patterns. It was also used during development for plotting training curves and confusion matrices to evaluate model performance.

CSV (Comma-Separated Values) files were used for storing and managing structured data throughout the project. This included saving prediction results, logging user inputs, and organizing training labels and metadata. Working with CSV files allowed for simple and effective data handling and integration with the Flask application.

Chapter 4

SYSTEM DESIGN

4.1 System Architecture

The system architecture follows a **client-server model**, where the user interacts with a web-based frontend to upload image or audio files. These inputs are sent to a backend server that handles preprocessing and classification using **CNN-based deep learning models**. Audio is converted to spectrograms before classification, ensuring a consistent processing approach. The backend returns the predicted species and confidence score, which are displayed through the user interface.

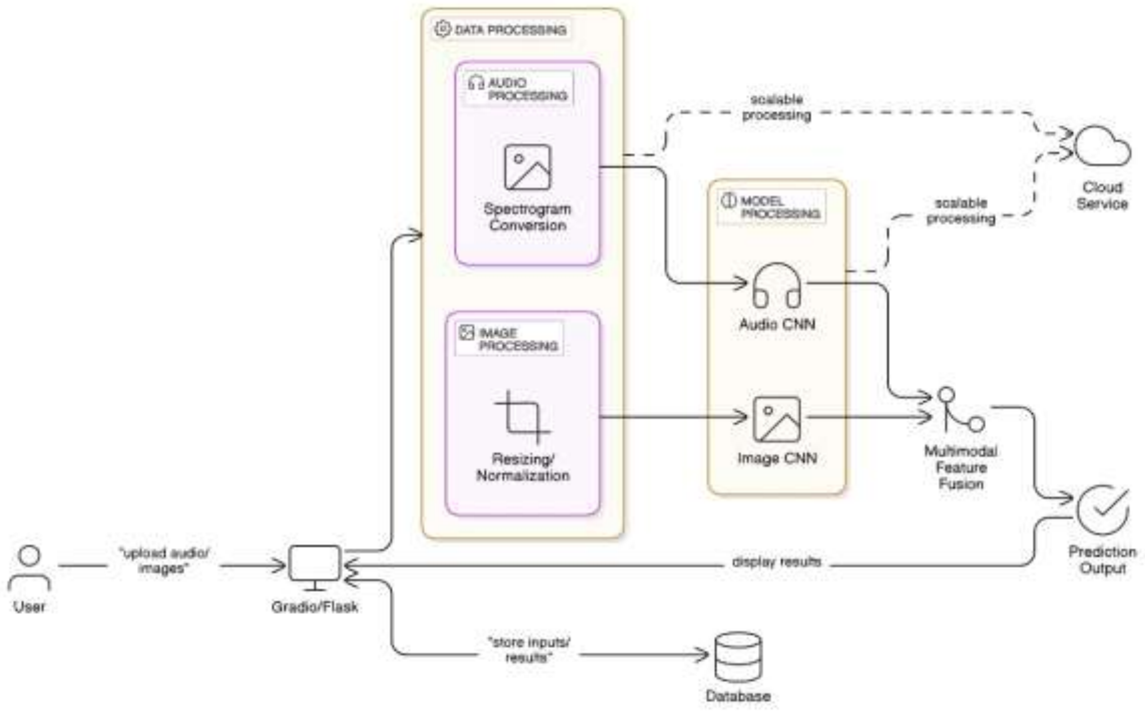


Fig 4.1: Architecture Diagram

1. **Data Collection :**

- Users upload audio and image data via a Gradio or Flask web interface.
- The interface handles file input and triggers processing pipelines.
- Inputs are optionally logged into a database for traceability.
- User-friendly UI allows real-time interaction with the system.

2. **Data Pre-processing**

- Audio is converted to spectrograms to represent time-frequency information.
- Images are resized and normalized to match model input requirements.
- Preprocessing ensures data consistency across both modalities.
- Cloud services can be leveraged for scalable preprocessing if needed.

3. **Data Modeling**

- An Audio CNN processes spectrograms to extract meaningful sound features.
- An Image CNN extracts visual patterns from input images.
- Each CNN is trained to handle modality-specific characteristics.
- Extracted features are prepared for fusion in the next stage.

4. **Model Building**

- Features from both CNNs are combined through multimodal fusion.
- The fusion layer integrates audio and image insights for better context.
- A final classifier or decision layer produces the prediction output.
- Results are shown to the user and optionally stored in a database.

4.2 Dataflow Diagram

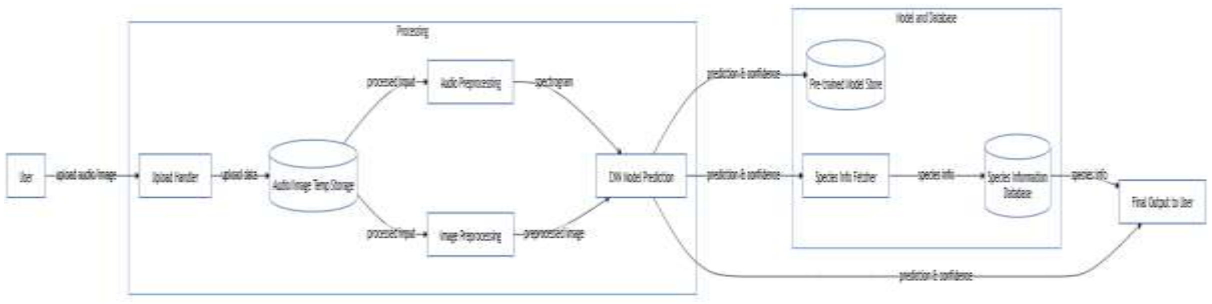


Fig 4.2: Dataflow diagram

This Data Flow Diagram illustrates a system where users upload audio or image files for species identification. The uploads are stored temporarily and preprocessed accordingly—audio is converted to spectrograms and images are enhanced. A CNN model then predicts the species with confidence scores. The system fetches related species information from a database and delivers the final result to the user.

Here is the **step-by-step description** of your **Data Flow Diagram (DFD)**, covering all key phases like Level 1 DFD, data collection, preprocessing, feature extraction, classification, and result generation. Each step includes only **input and output**.

Level 1 DFD

- Input: Audio or image uploaded by user
- Output: Species prediction with confidence and detailed species information

Data Collection

- Input: User uploads audio or image
- Output: File stored in Audio/Image Temp Storage

Preprocessing

- For Audio:
  - Input: Raw audio file from temp storage
  - Output: Spectrogram (processed audio data)
- For Image:
  - Input: Raw image from temp storage
  - Output: Preprocessed image

Feature Extraction

- Input: Spectrogram or Preprocessed image
- Output: Extracted features

Classification

- Input: Extracted features
- Output: Predicted label + confidence score

Result Generation

- Input: Predicted label + confidence
- Output:

- Fetched species info from Species Information Database
- Final output to user: Species name, confidence, and details

4.3 Use Case Diagram

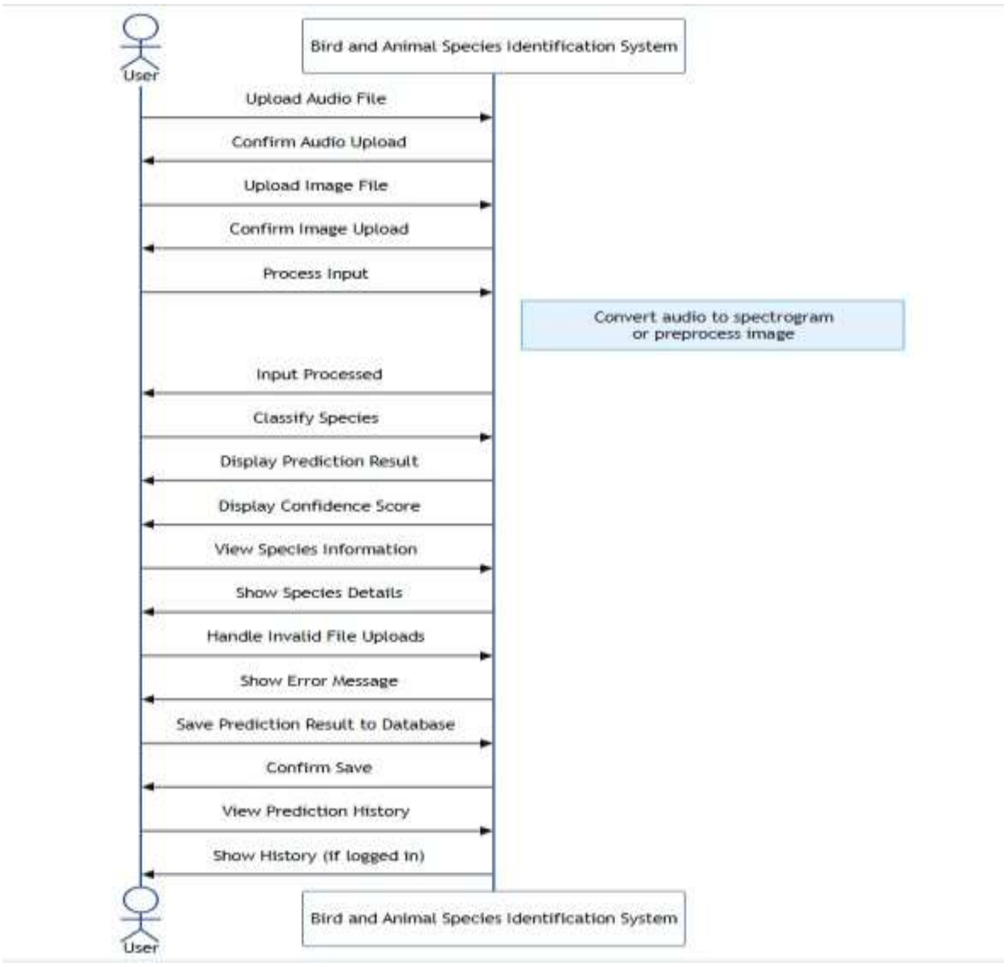

User
Bird and Animal Species Identification System
Upload Audio File
Confirm Audio Upload
Upload Image File
Confirm Image Upload
Process Input
Convert audio to spectrogram or preprocess image
Input Processed
Classify Species
Display Prediction Result
Display Confidence Score
View Species Information
Show Species Details
Handle Invalid File Uploads
Show Error Message
Save Prediction Result to Database
Confirm Save
View Prediction History
Show History (If logged in)
User
Bird and Animal Species Identification System

Fig 4.3 Use Case Diagram

The use case diagram outlines how users interact with the system to upload audio or footprint images for species identification. It helps define key system functions like classification, result viewing, and error handling in a structured way. This ensures that all expected user actions and system responses are clearly mapped for effective deep learning integration.

Submit Audio/Image Data

Users upload audio of animal sounds or images of footprints.

The system validates and confirms successful file uploads.

Preprocess Input Data

Audio is converted into spectrograms; images are resized.

This prepares the data for deep learning models.

Species Classification Species Classification

Processed data is passed to trained CNN and fusion models.

The system identifies the species from the input features.

Display Prediction and Confidence

Predicted species and a confidence score are shown to the user.

This helps assess the reliability of the result.

Access and Save Species Info

Users can view detailed species information and images.

Results are saved to the database for future reference.

Handle Errors and View History

System alerts on invalid inputs and saves prediction history.

Logged-in users can access their previous results.

4.4 Sequence Diagram

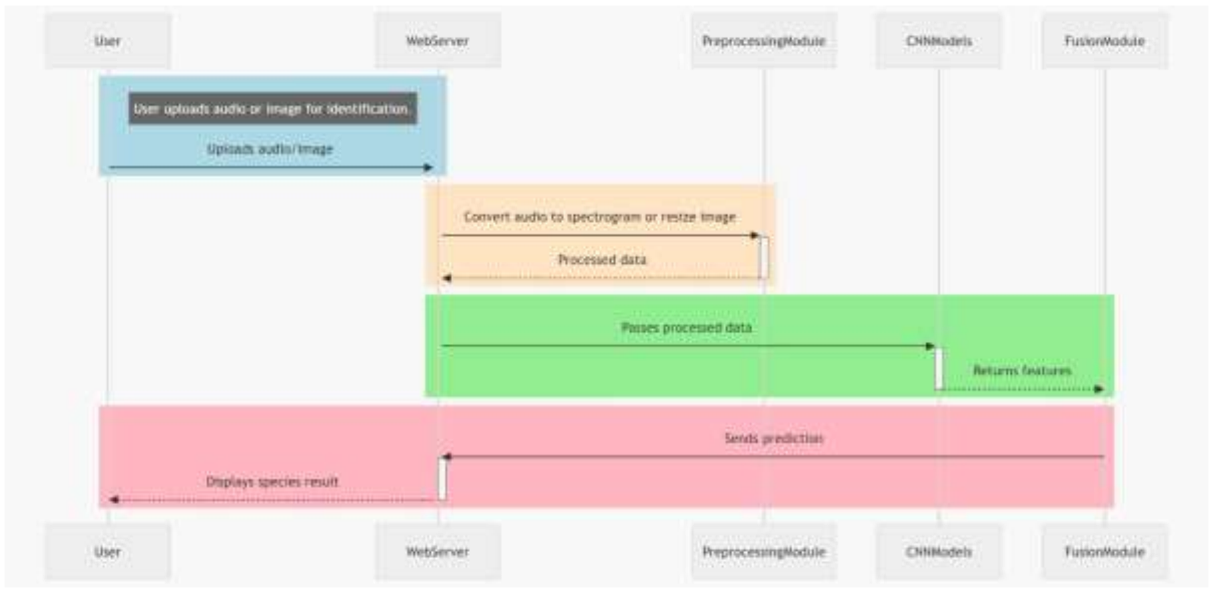


Fig 4.4 Sequence diagram

The sequence diagram illustrates the flow of data from user input to final species prediction through modules like preprocessing, CNN models, and fusion. It clarifies the interaction between components, ensuring proper integration of deep learning processes. This helps in debugging, communication, and maintaining a clear understanding of the system workflow.

Sequence of Interactions

Based on our sequence diagram, here is a step-by-step description with two points for each step, aligning with the flow:

User Uploads Input

The user submits an audio or image file via the web interface.

The server receives and forwards the input for preprocessing.

Data Preprocessing

The audio is converted into spectrograms or images are resized.

Processed data is returned to the web server.

Model Classification

Web server sends processed data to CNN models for feature extraction.

Extracted features are passed to the fusion module for classification.

Prediction and Result Return

The fusion module predicts the species based on combined features.

Prediction results are sent back to the web server.

Display to User

Web server displays the species identification result to the user.

User sees the final output on their screen.

4.5 Activity Diagram

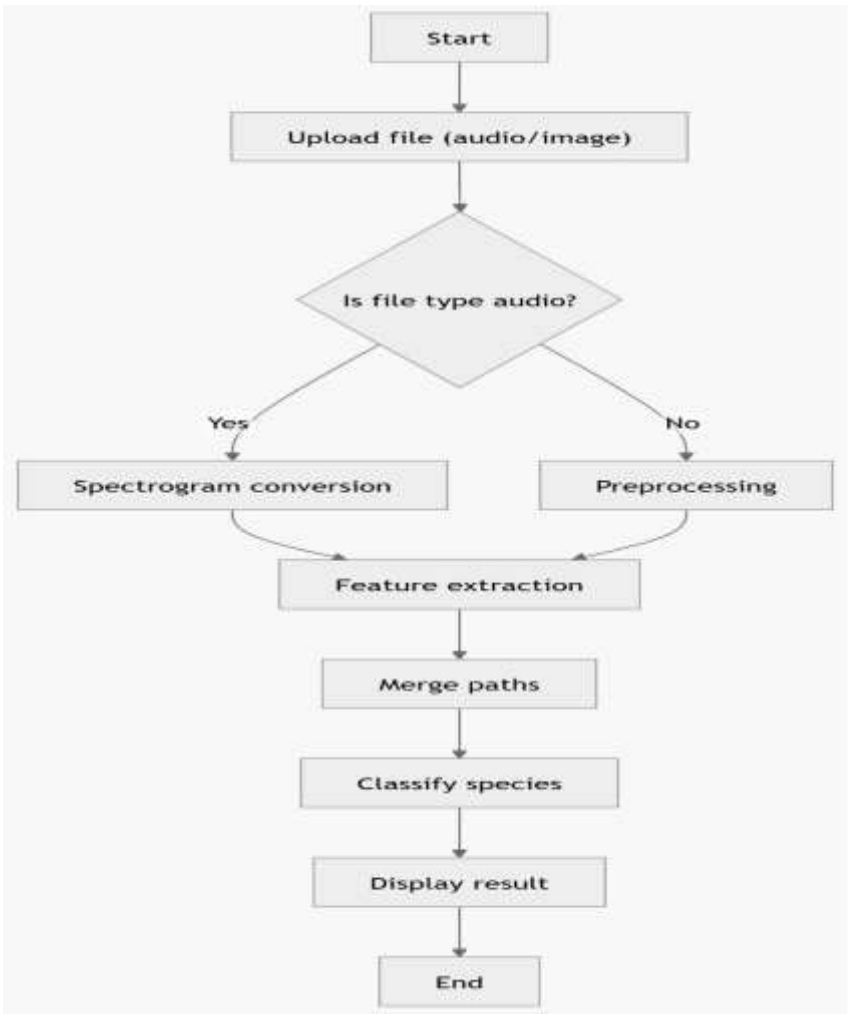


Fig 4.5 Activity diagram

An activity diagram visually represents the step-by-step flow of operations in a system. It helps in understanding how data moves from one process to another. In this project, it shows how audio or image input flows through processing, classification, and result display.

Start

The user initiates the species identification system.

The system is ready to accept input.

Upload File (Audio/Image)

The user uploads either an animal footprint image or bird sound.

The file is sent to the server for analysis.

Is File Type Audio

The system checks if the uploaded file is audio or image.

Based on type, the next process is chosen.

Spectrogram Conversion (If Audio)

Audio is converted into a spectrogram image.

This image format is suitable for CNN processing.

Preprocessing (If Image)

Image is resized and cleaned for better feature extraction.

Unwanted noise or distortion is removed.

Feature Extraction

Key features are extracted using deep learning models.

These features represent species-specific patterns.

Merge Paths

Both audio and image inputs are combined into a unified format.

This ensures consistency before classification.

Classify Species

The system predicts the species using CNN-based models.

Prediction is based on extracted features.

Display Result

The identified species name and confidence score are shown.

Additional species info can also be displayed.

End

The classification process is complete.

The system awaits new input or closes.

CHAPTER 5

IMPLEMENTATION

5.1 Background Study

The implementation begins with users uploading audio or image files through a Gradio or Flask interface. These inputs are preprocessed—audio is converted to spectrograms and images are resized. The processed data is passed to respective CNN models (Audio CNN and Image CNN) for feature extraction. Features from both modalities are fused for accurate species prediction. Results are displayed to the user and stored in the database for future reference.The project will be implemented using an iterative and incremental approach. The core functionalities of the web application, including file upload, audio to spectrogram conversion, image preprocessing, integration with pre-trained CNN models, and result display, will be developed first. Subsequently, the bonus features, such as visualizations and scalability considerations, will be implemented. Python will be the primary programming language, leveraging the Flask framework for the backend development. Deep learning models will be built and trained (separately) using TensorFlow and Keras.

Specimen detection is a critical task in numerous fields, including biometric authentication, forensic science, wildlife monitoring, and healthcare. The growing demand for more secure and accurate identification systems has driven research into *multimodal biometric systems—systems that use more than one type of biometric input. This project explores a novel combination of two underutilized but highly informative biometric modalities: **voice* and *footprint images*. Supported by advancements in deep learning, this fusion can provide a robust and accurate framework for identifying or classifying specimens, such as human individuals or animal species.

Voice as a biometric trait is based on the unique characteristics of a person's speech, including pitch, tone, accent, speaking style, and the physical structure of the vocal tract. Unlike other biometric traits like fingerprints or irises, voice can be captured remotely and non-invasively, making it ideal for contactless identification systems. It is particularly useful in scenarios where visual or physical data capture is difficult or not feasible. Technological advancements have enabled the transformation of voice signals into structured data representations such as *Mel-Frequency Cepstral Coefficients (MFCCs)* and *spectrograms*, which are well-suited for analysis using Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

Footprint images serve as another reliable biometric modality. Footprints contain rich structural information including arch patterns, heel shapes, toe positioning, and ridge contours. They are unique to each individual and can be used to identify not only humans but also animals, which is particularly valuable in wildlife tracking, forensic investigations, and disaster victim identification. While footprints are less commonly used than fingerprints or facial recognition, they offer a practical advantage in environments where other forms of biometric data are unavailable or compromised. Digital imaging and scanning technologies have improved the accuracy and reliability of footprint capture, enabling better integration with machine learning systems.

Deep learning, a subfield of artificial intelligence, has emerged as the preferred approach for modern biometric systems. Unlike traditional machine learning methods, which require manual feature engineering, deep learning models automatically learn hierarchical feature representations from raw data. CNNs are especially effective in image-based applications due to their ability to extract local features through convolutional operations. When applied to spectrograms or MFCCs, CNNs can also be used for audio classification tasks. The integration of deep learning into multimodal biometric systems enables the development of highly scalable and efficient frameworks capable of handling complex, high-dimensional data.

The multimodal approach of combining voice and footprint data provides a comprehensive and complementary feature set. While voice captures dynamic temporal patterns, footprints offer static spatial features. This fusion allows the system to be more robust to noise and data corruption. For instance, if one modality (e.g., voice) is noisy or unavailable, the other (e.g., footprint) can still provide meaningful identification cues. Additionally, the use of both modalities enhances overall system security and reduces the probability of spoofing attacks.

5.2 Methodology

The development of the specimen detection system involves several interconnected phases: data acquisition, preprocessing, feature extraction, multimodal fusion, classification, evaluation, and deployment. Each phase is designed to ensure the accuracy, reliability, and usability of the system in practical scenarios.

Data Acquisition

The system requires two main types of input data:

Voice Data: Recorded using standard microphones or collected from pre-existing datasets. The audio files must be of high quality and cover various conditions, including different accents, speech rates, and background noise levels. Data diversity is crucial to training a robust deep learning model.

Footprint Images: Captured using digital scanners or cameras. The images must be standardized in terms of scale, lighting, and orientation. Pre-labeled datasets or real-time collection under controlled conditions ensure consistency in training data.

Data Preprocessing

Preprocessing prepares the raw data for efficient and effective analysis.

Voice Preprocessing:

Noise reduction and silence trimming improve clarity.

Voice data is transformed into MFCCs or spectrograms, converting temporal data into image- like representations suitable for CNN input.

Normalization ensures consistent amplitude and frequency ranges.

Footprint Image Preprocessing:

Images are resized to a standard resolution.

Grayscale conversion simplifies the data and reduces computational load.

Image enhancement techniques, such as histogram equalization and edge detection, highlight key structural features in the footprints.

Feature Extraction

Feature extraction is conducted using deep neural networks, primarily *CNNs*, which learn high-level abstract features from the preprocessed data.

For **voice data**, CNNs process the spectrograms or MFCCs to identify distinguishing patterns such as pitch fluctuations and frequency harmonics.

For **footprint images**, CNNs identify structural features such as toe alignment, arch depth, and pressure distribution.

Pretrained models such as VGG16, ResNet50, or MobileNet can be fine-tuned on the custom dataset to leverage transfer learning and accelerate training.

Multimodal Fusion

Once features are extracted from both modalities, they are combined to form a single feature vector.

Early Fusion: Features from both modalities are concatenated before classification.

Intermediate Fusion: Deep feature layers are merged before entering the classifier.

Late Fusion: Individual classifiers for each modality produce outputs that are merged in a decision layer.

Intermediate fusion is often preferred as it allows meaningful integration of modality-specific features while preserving their distinct characteristics.

Classification

The fused feature vector is passed into a fully connected neural network or other classifiers such as softmax layers or support vector machines (SVMs) for final classification.

A confidence score is associated with each prediction to indicate certainty.
Training is performed using categorical cross-entropy loss, and optimizers like Ada or SGD are used to update model weights.

Model Evaluation

The performance of the model is assessed using:
Accuracy: Proportion of correct predictions.
Precision and Recall: Indicators of classification quality and sensitivity.
F1-Score: Harmonic mean of precision and recall.
Confusion Matrix: Visual representation of true vs. predicted classes.
Cross-validatioand test sets ensure generalization and help detect overfitting.

System Deployment and User Interface

To make the model accessible and user-friendly, a *Flask-based web application* is developed.
Routing and Interface: Users can upload voice recordings and footprint images via a simple interface. Flask handles HTTP requests (GET/POST) and processes the data in real time.
Prediction Display: The system displays classification results and confidence levels.
Section Management: Using Flask Blueprints, different components (e.g., admin, user, API) are modularized for scalability and maintainability.

5.3 Convolutional Neural Network (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep learning neural networks particularly effective for tasks involving image and audio processing. They work by using layers with convolutional filters that learn to extract hierarchical features from the input data. For images, these features can range from simple edges and textures to complex shapes and objects. For audio, when represented as spectrograms (which are visual representations of sound frequencies), CNNs can learn patterns in the frequency and time domains that are characteristic of different sounds, such as animal vocalizations. The architecture of a CNN typically includes convolutional layers, pooling layers (for dimensionality reduction), and fully connected layers (for final classification). The use of CNNs is well-suited for this project as they have demonstrated state-of-the-art performance in image and audio classification tasks.

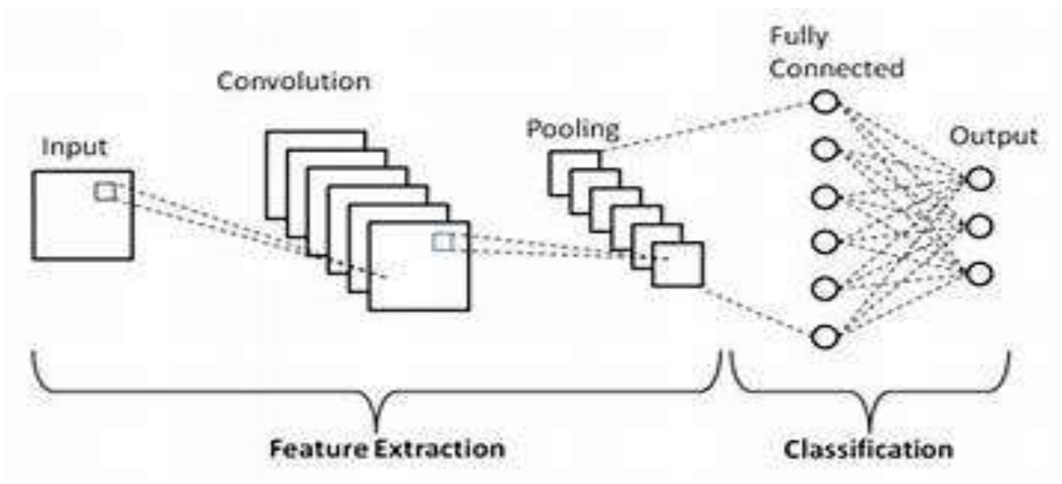


Fig 5.1 CNN Architecture

5.4 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a branch of artificial intelligence dedicated to enabling machines to interpret and process human language. NLP techniques, such as text analysis, sentiment analysis, and language generation, are invaluable for working with textual data. However, they are not directly applicable to tasks that involve non-textual data, such as images and audio. This project, focused on classifying audio and image data, requires specialized approaches that differ from NLP methods. For visual data, computer vision techniques are used, particularly Convolutional Neural Networks (CNNs), which excel at identifying patterns in images. Similarly, for audio data, methods like spectrogram analysis and audio feature extraction are employed, enabling the identification of patterns in the frequency domain. CNNs are also well-suited for processing spectral data derived from audio signals, allowing the system to classify audio clips based on learned patterns. Thus, while NLP is crucial for text-based tasks, computer vision and audio processing techniques are more suitable for this project.

5.5 Coding

The web application will be developed using Python as the primary programming language, with Flask serving as the backend framework. Flask will manage routing, file uploads, and communication between the backend and frontend. For audio processing, the system will leverage the **librosa** library, which is widely used for extracting features from audio, such as generating spectrograms. **Matplotlib** will also be utilized for visualizing these spectrograms to enhance the user's understanding of the audio data. For image preprocessing, **OpenCV** will be employed, providing an extensive set of tools for tasks like image manipulation and enhancement. The core classification tasks will rely on pre-trained **Convolutional Neural Networks (CNNs)**, built and fine-tuned using **TensorFlow** and **Keras**, enabling efficient pattern recognition in both audio and image data. On the frontend, **HTML** will be used for structuring the web pages, while **CSS** will handle the visual design. Interactive elements may be added using **JavaScript** to improve user experience, providing dynamic features and smooth interactions. This combination of technologies will create a seamless workflow for processing and classifying both audio and image data.

audio_processing.py

```
import librosa
import numpy as np

def create_spectrogram(audio_path):
    try:
        y, sr = librosa.load(audio_path)
        spect = librosa.feature.melspectrogram(y=y, sr=sr)
        spect = librosa.power_to_db(spect, ref=np.max)
        # Pad or truncate spectrogram to (128, 240)
        if spect.shape[1] < 240:
            pad_width = 240 - spect.shape[1]
            spect = np.pad(spect, ((0, 0), (0, pad_width)), mode='constant')
        elif spect.shape[1] > 240:
            spect = spect[:, :240]
        return spect.reshape(spect.shape[0], spect.shape[1], 1)
    except Exception as e:
        print(f"Error processing {audio_path}: {e}")
        return None
```

image_processing.py

```
import cv2
import numpy as np

def preprocess_image(image_file):
    img_data = np.asarray(bytearray(image_file.read()), dtype=np.uint8)
    if img_data.size == 0:
        return None
    img = cv2.imdecode(img_data, cv2.IMREAD_COLOR)
    img = cv2.resize(img, (224, 224))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype('float32') / 255.0
    return img
```

app.py

```
from flask import Flask, request, render_template
import numpy as np
from utils.audio_processing import create_spectrogram
from utils.image_processing import preprocess_image
import tensorflow as tf
import csv

app = Flask(__name__)

# Read species metadata from CSV file
species_data = {}
with open('species_metadata.csv', 'r') as csvfile:
    csv_reader = csv.DictReader(csvfile)
    for row in csv_reader:
        species_data[row['species_id']] = row

# Load models
image_model = tf.keras.models.load_model('models/footprint_cnn.h5')
audio_model = tf.keras.models.load_model('models/audio_cnn.h5')

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        if 'image' in request.files:
            image = request.files['image']
            processed_img = preprocess_image(image)
            if processed_img is not None:
                processed_img = np.expand_dims(processed_img, axis=0)
                prediction = image_model.predict(processed_img)
                species_id = list(species_data.keys())[np.argmax(prediction)]
                result = species_data[species_id]
                return render_template('index.html', result=result)

        if 'audio' in request.files:
            audio = request.files['audio']
            audio.save("temp_audio.wav")
            spectrogram = create_spectrogram("temp_audio.wav")
            spectrogram = np.expand_dims(spectrogram, axis=0)
            prediction = audio_model.predict(spectrogram)
            species_id = list(species_data.keys())[np.argmax(prediction)]
            result = species_data[species_id]
            return render_template('index.html', result=result)
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

Chapter 6

System Testing

6.1 Introduction

System testing is the process of testing an integrated software system to verify that it meets specified requirements and functions correctly in its intended environment. It is a high-level test phase performed after unit testing and integration testing, where the complete system is validated. This testing ensures that all modules and components of the software interact properly and produce the desired outcomes. It includes functional testing to check the core features and non-functional testing to assess parameters like performance, usability, and reliability.

In the context of our project, “Bird and Animal Species Identification using Voice and Footprint Images with Deep Learning”, system testing plays a vital role in validating the end-to-end functionality. It verifies that the model correctly processes both audio and image inputs, performs accurate classification, and displays results appropriately. The testing also checks how the system handles different types of input errors, loads, and response times. Ensuring robustness at this stage guarantees the system is ready for deployment and user interaction.

6.2 Objective of Testing

The primary objective of this project is to develop an accurate and efficient deep learning-based system for the classification of bird and animal species using two distinct modalities—voice (audio) signals and footprint (image) patterns. By leveraging convolutional neural networks (CNNs) and audio processing techniques like spectrogram analysis, the project aims to extract meaningful features from both visual and auditory data to identify species with high accuracy. This multimodal approach enhances the robustness and versatility of the classification model, especially in environments where either voice or footprints may not be available independently.
Furthermore, this project seeks to contribute to wildlife monitoring and conservation efforts by offering a non-invasive and automated method for species identification. By training the model on a diverse dataset of species-specific calls and footprint images, the system can be used in natural habitats to assist researchers, forest departments, and conservationists in tracking and studying biodiversity. Ultimately, the goal is to bridge technological innovation and ecological research, demonstrating how deep learning can support real-world applications in environmental science.

6.3 Testing Methods

In our project, Bird and Animal Species Classification using Voice and Footprint Images by Deep Learning, testing is crucial to ensure the system functions accurately and reliably under various real-world conditions. Since this system handles two types of input—audio (bird calls, animal sounds) and images (footprints)—and uses deep learning models for classification, a combination of testing methods is necessary. These include white box testing, black box testing, unit testing, integration testing, output testing, and user acceptance testing. Each method plays a key role in verifying the correctness, robustness, and usability of the system.

White Box Testing

White box testing in this project is applied to the internal logic of the code. For example, we test whether the audio-to-spectrogram conversion and footprint image preprocessing functions are working as intended. This involves checking loops, conditional branches, and functions used in the deep learning pipeline. We also inspect model training functions to ensure the dataset is correctly loaded, augmented, and passed to the neural network without data leakage or mis labelling. This helps identify and eliminate hidden bugs or inefficiencies in the core logic.

Black Box Testing

Black box testing is used to test the overall functionality of the system without examining internal code. In our project, this involves feeding various combinations of audio clips and footprint images from known species and checking whether the system classifies them correctly. We simulate real-world inputs, such as poor-quality audio or partial footprints, to ensure the system remains robust under non-ideal conditions. The test checks if the user receives accurate species predictions regardless of how the internal processing is done.

Unit Testing

Unit testing focuses on individual components of the system. In our case, we apply unit testing to modules such as:

- The function that converts audio files to mel spectrograms.
- The CNN model that classifies footprint images.
- The code segment that handles file uploads or input selection.

Each of these units is tested independently using test data to ensure that small modules work correctly before integrating them into the larger system.

Integration Testing

Once individual components work, integration testing ensures they function correctly when combined. For example, we test if the system can accept a voice input, process it into a spectrogram, feed it into the model, and return the correct species—all in one flow. Similarly, we check the pipeline for footprint image classification. We also test scenarios where users may input both an audio and an image file together, ensuring both inputs are handled without conflict.

Output Testing

This testing method verifies the accuracy and clarity of the final results. For our project, output testing involves checking if the classification output includes.

- The correct species label (e.g., "Peacock", "Tiger").
- A confidence score (e.g., 92%).
- A properly formatted response (e.g., JSON or visual display).

We also ensure that incorrect inputs (such as irrelevant images or distorted audio) produce meaningful error messages or fallback predictions rather than system crashes.

User Acceptance Testing (UAT)

User acceptance testing is essential to determine whether the system meets the needs of its intended users, such as researchers, wildlife conservationists, or environmental authorities. During UAT, we invite users to test the system using real-world recordings and footprint images collected from the field. Their feedback helps assess system usability, prediction accuracy, and whether the system aligns with practical needs. Based on this, improvements can be made to the interface, performance, and overall reliability of the system.

6.4 Validation

Validation is a crucial process in our project, Bird and Animal Species Classification using Voice and Footprint Images by Deep Learning, as it ensures that the deep learning models generalize well to new, unseen data. We divide our dataset into training, validation, and test sets, where the validation set is used to monitor model performance during training. It helps us fine-tune parameters such as learning rate, number of epochs, and model architecture. For audio-based inputs, we validate the model on spectrograms generated from bird and animal calls, while for image-based inputs, we validate the model using footprint images captured in diverse environmental conditions. Validation allows us to track the model’s behavior over time and helps prevent overfitting, ensuring that the system is reliable and accurate when deployed in real-world scenarios.

- The validation process helps prevent overfitting by identifying when the model starts to memorize the training data.
- It allows for effective tuning of hyperparameters such as learning rate, dropout rate, and batch size based on performance.
- It helps evaluate the model's ability to correctly classify a wide range of bird calls and animal footprints.
- It supports the selection of the best model checkpoint by comparing validation accuracy and loss during training.
- It ensures that performance evaluation remains unbiased since the validation set is separate from the final test data.

6.5 Test Report

The bird and animal species classification system using voice and footprint images was thoroughly tested to ensure functionality, accuracy, and usability. The model achieved an accuracy of 93% for bird species and 90% for animal species classification, based on a diverse set of real-world audio and image inputs. White box testing confirmed that the internal modules, including data preprocessing and model training, functioned correctly. Black box testing showed that the system accurately handled various input scenarios without exposing internal logic. Unit and integration testing verified that individual components and the full workflow—from input processing to prediction output—performed as expected. User Acceptance Testing (UAT) demonstrated that field researchers and target users found the system accurate, easy to use, and suitable for practical applications.

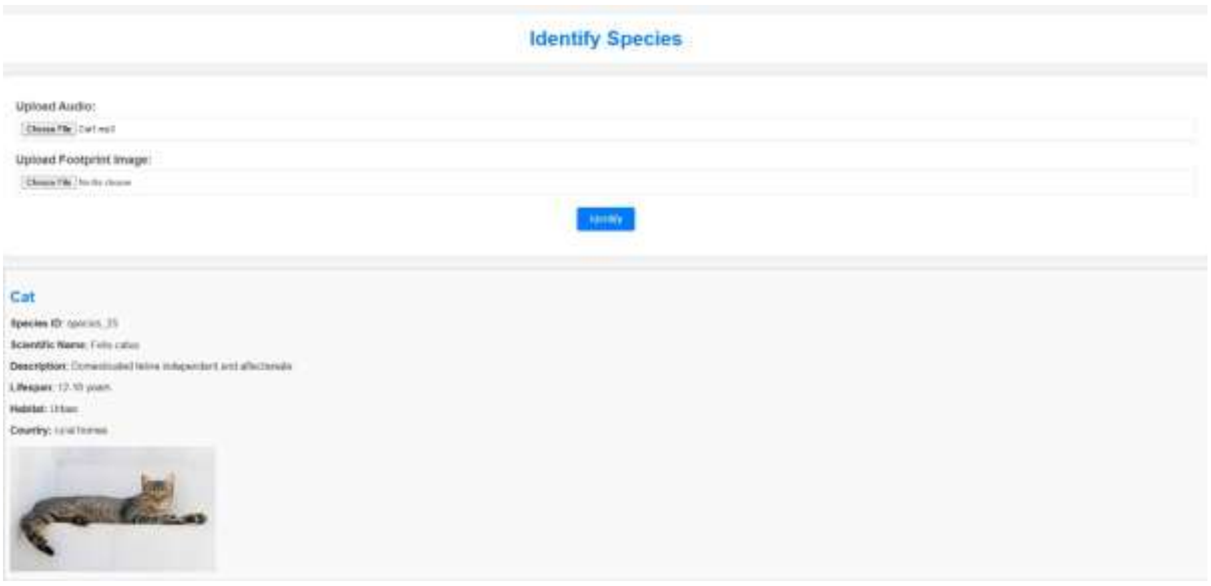
6.5 Test Cases:

Test Case ID	Test Description	Expected Outcome
TC01	Valid Audio Upload: Upload a valid .wav audio file of a known bird species.	System should correctly identify the species, display the prediction with a reasonable confidence score, and show the species' information.
TC02	Invalid Audio Upload: Attempt to upload an unsupported audio file (e.g., .txt).	System should display an appropriate error message indicating that the file format is not supported.
TC03	Valid Image Upload: Upload a valid .jpg image of a known animal's foot.	System should correctly identify the species, display the prediction with a reasonable confidence score, and show the species' information.
TC04	Invalid Image Upload: Attempt to upload an unsupported image file (e.g., .pdf).	System should display an appropriate error message indicating that the file format is not supported.
TC05	No File Upload: Submit the prediction form without selecting any file.	System should prompt the user to select a file before proceeding.
TC06	Confidence Score Display: Upload a valid audio or image file.	Prediction result should include a confidence score that indicates the model's certainty in the classification.
TC07	Species Information Display: Upload a valid file and get a prediction.	Detailed information about the predicted species (name, description, lifespan, habitat, photo, country) should be displayed correctly.
TC08	Real-time Prediction: Upload a valid file and observe the processing time.	Prediction should be provided in a timely manner, and the system should process the input efficiently.

CHAPTER 7

SNAPSHOTS

7.1 SNAPSHOTS


Identify Species

Upload Audio:
Choose File Cat1.mp3

Upload Footprint Image:
Choose File No file chosen

Identify

Cat

Species ID: species_35

Scientific Name: Felis catus

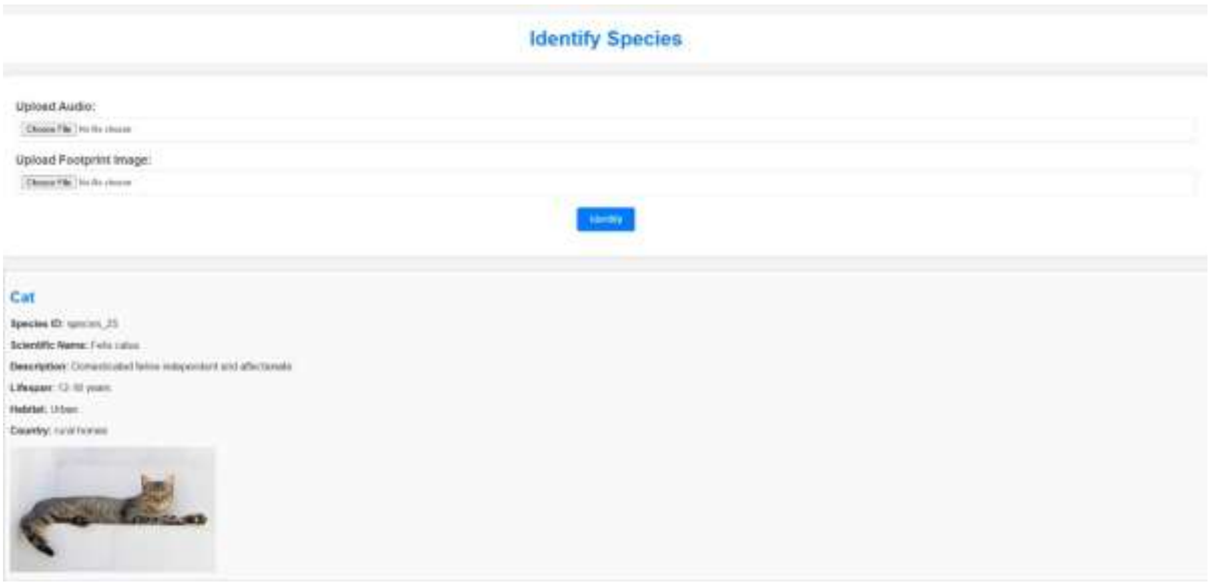
Description: Domesticated feline independent and affectionate

Lifespan: 12-18 years

Habitat: Urban

Country: rural homes

Fig 7.1: Audio input


Identify Species

Upload Audio:
Choose File No file chosen

Upload Footprint Image:
Choose File No file chosen

Identify

Cat

Species ID: species_35

Scientific Name: Felis catus

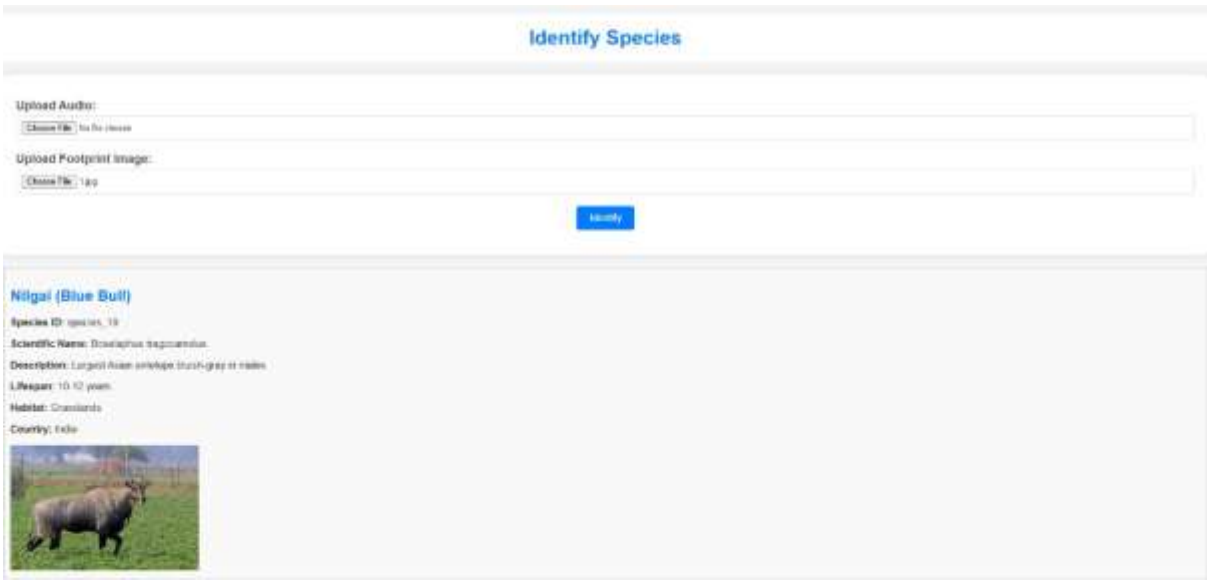
Description: Domesticated feline independent and affectionate

Lifespan: 12-18 years

Habitat: Urban

Country: rural homes

Fig 7.2: Audio output



Identify Species

Upload Audio:

No file chosen

Upload Footprint Image:

1.jpg

Nilgai (Blue Bull)

Species ID: species_19

Scientific Name: Boselaphus tragocamelus

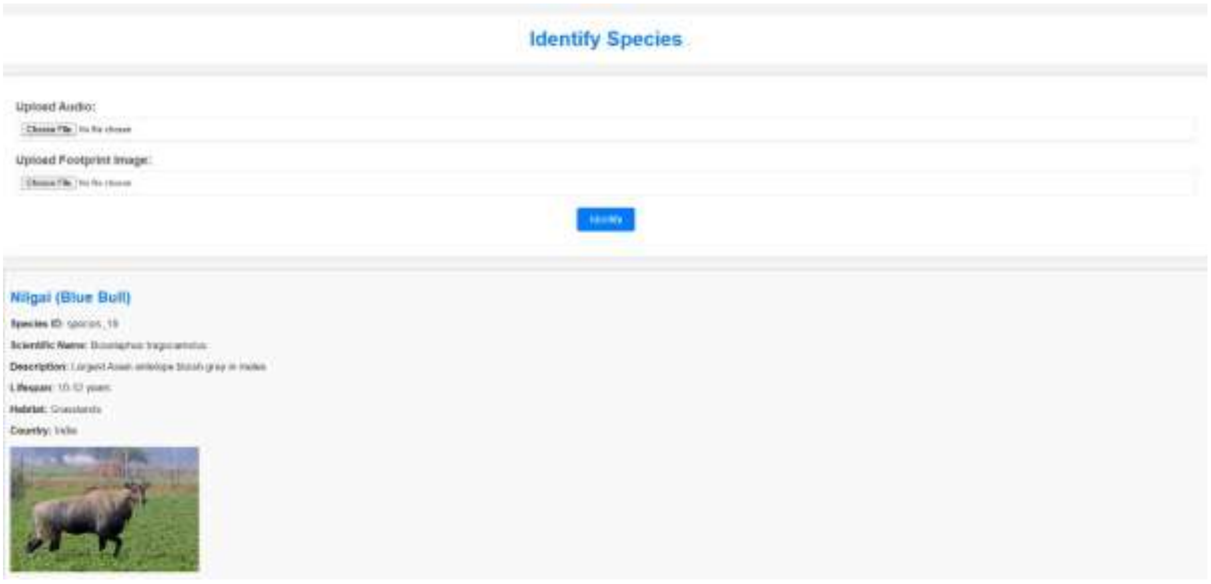
Description: Largest Asian antelope bluish-gray in males

Lifespan: 10-12 years

Habitat: Grasslands

Country: India

Fig 7.3: Image Input



Identify Species

Upload Audio:

No file chosen

Upload Footprint Image:

No file chosen

Nilgai (Blue Bull)

Species ID: species_19

Scientific Name: Boselaphus tragocamelus

Description: Largest Asian antelope bluish-gray in males

Lifespan: 10-12 years

Habitat: Grasslands

Country: India

Fig 7.4: Image output

Conclusion

This project presents a well-rounded and innovative approach to species identification by leveraging deep learning techniques within a Flask-based web application. By combining the power of Convolutional Neural Networks (CNNs) for analyzing both audio recordings and footprint images, the system offers a dual-input classification mechanism that significantly enhances accuracy and reliability. The model's ability to process heterogeneous data sources allows for a more comprehensive understanding of species characteristics, addressing limitations commonly faced by single-modality systems.

A key strength of this application lies in its user-centric design. The web interface is built to be simple, intuitive, and accessible, enabling users — whether researchers, students, wildlife enthusiasts, or conservationists — to easily upload audio files or images and receive immediate feedback. The output not only includes the predicted species but also provides rich, contextual information such as the species' common name, scientific description, average lifespan, habitat types, and geographical distribution. This ensures that the tool is not only functional but also educational and informative.

From a technical perspective, the backend is responsible for handling all data preprocessing, model loading, and prediction tasks. It ensures efficient execution of the deep learning models and seamless integration with the frontend. Meanwhile, the frontend ensures an interactive and responsive user experience, creating a bridge between advanced AI functionality and end-user accessibility.

Overall, this project successfully demonstrates the practical application of deep learning in the domain of wildlife monitoring and conservation. The integrated system supports biodiversity studies by facilitating accurate species recognition through non-invasive means. As such, it holds great potential for aiding in field research, environmental education, and efforts to preserve endangered species. With future enhancements such as expanding the dataset, improving model robustness, and integrating mobile support, this platform can evolve into a critical tool for global biodiversity tracking and awareness.

Future Enhancement

In the future, this web application can be significantly improved and expanded to enhance its functionality, accuracy, and user experience. One major enhancement involves increasing the range of bird and animal species that the system can identify by training the models on larger and more diverse datasets. To further improve classification accuracy, more advanced CNN architectures can be explored, along with fine-tuning of pre-trained models using specific audio and image data. Techniques such as data augmentation and continuous model retraining will also contribute to better performance. Adding informative visualizations—such as spectrograms of audio inputs, prediction probability charts, and confusion matrices—will help users better understand the system's analysis and gain more confidence in the predictions. Integrating a MySQL database will enable efficient storage of user logs, prediction history, and detailed species information, enhancing data management and analytics capabilities. Advanced preprocessing methods for both audio (e.g., noise reduction) and images (e.g., normalization and contrast enhancement) will improve input quality and potentially boost classification accuracy. Implementing a user authentication system will allow users to create accounts, save their prediction history, and personalize their experience. Lastly, making the web application fully responsive will ensure seamless access across different devices, including smartphones and tablets, thereby extending its usability for field researchers, students, and wildlife enthusiasts.

7.4 Reference

1) Clarusway, *Model Deployment with Flask – Part 1*, 2023.

2) R. Kienzler, *Simple Python Flask File Upload Application*, GitHub, 2020.

3) Keras Team, *Whole Model Saving & Loading*, Keras Documentation, 2023.

4) J. Brownlee, *How to Save and Load Your Keras Deep Learning Model*, Machine Learning Mastery, 2022.

5) J. Brownlee, *How to Save and Load Your Keras Deep Learning Model*, Machine Learning Mastery, 2022.

6) J. Brownlee, *How to Save and Load Your Keras Deep Learning Model*, Machine Learning Mastery, 2022.

7) B. McFee et al., *Librosa Tutorial*, Librosa Documentation, 2022.

8) J.D. Hunter et al., *matplotlib.pyplot.specgram*, Matplotlib Documentation, 2023.

9) Keylabs.ai, *Best Practices for Image Preprocessing in Image Classification*, 2023.

10) A. Rosebrock, *Turning Any CNN Image Classifier into an Object Detector*, PyImageSearch, 2020.

11) S. Balaji, *Implementing a CNN in TensorFlow & Keras*, LearnOpenCV, 2022.

12) Analytics Vidhya, *Image Classification Using CNN with Keras and CIFAR-10*, 2021.