ENVIROVISION

SPEICES IDENTIFICATION SOFTWARE

In partial fulfilment, this dissertation is submitted to

Dr. B. R. Ambedkar Institute of Technology

Pondicherry University for award of

Bachelor of Technology in Computer Science Engineering

By

SWASTI RANJAN BISWAS

T NARESH

DIMPY SARKAR

K VIJAY KUMAR

Under the guidance of

MR. HARSABARDHAN BARIK

(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DR. B. R. AMBEDKAR INSTITUTE OF TECHNOLOGY, PORT BLAIR

MAY 2025

CERTIFICATE

This is to certify that the report entitled **"ENVIROVISION - Speices Identification Software"**, submitted by**Swasti Ranjan Biswas, T Naresh, Dimpy Sarkar , K Vijay Kumar**bearing university Enrolment Roll Nos. **21TD0270, 21TD0261, 21TD0259& 21TDO273** respectively, studying in VIII semester of Bachelor of Technology in **Computer Science & Engineering**in the session **2024-25** of this Institute, is a record of Bonafide work done under my guidance and it is considered worthy of consideration for the award of Bachelor of Technology in **Computer Science & Engineering** from Pondicherry University.

GUIDE **HOD/FI**

COORDINATE **PRINCIPAL**

I have conducted the viva – voice on __ May 2025 and found the project satisfactory.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The increasing need for rapid and accurate species identification in ecological and zoological fieldwork has highlighted a significant challenge faced by researchers and field scientists at the Zoological Survey of India (ZSI). Traditionally, species identification has been a time-consuming and expertise-driven task, often requiring manual verification and cross-referencing with physical or digital resources. To address this problem, we have developed a **Species Identification Software** powered by deep learning techniques, specifically Convolutional Neural Networks (CNNs), to streamline and automate the identification process in real time.

The core objective of this project is to design a robust and scalable system that can classify and identify various species based on an input image captured in the field. The software employs a **two-stage CNN architecture**. The first stage, known as the **Main CNN**, utilizes the **XceptionNet model** enhanced with dropout layers to perform high-level classification — determining whether the species belongs to a major category such as bird, butterfly, insect, etc. Upon initial classification, the image is routed to the corresponding **Sub CNN**, each designed for specific categories. These Sub CNNs are built using the **EfficientNetV2 Large architecture**, which allows for fine-grained classification within each category, identifying the exact species.

Experimental evaluations demonstrate high classification accuracy and efficiency, making the system reliable for field deployment. The software significantly reduces the time and expertise required for species identification, allowing researchers to focus more on data collection and analysis. The modular architecture also supports easy scalability to include more species categories in the future.

This project not only provides a practical tool for biodiversity studies but also showcases the potential of deep learning in solving real-world scientific problems. It represents a meaningful contribution towards the digitization and automation of biological fieldwork processes.

ACKNOWLEDGEMENT

Teamwork and commitment are essential elements of any collaborative effort. This project is the outcome of the dedicated support and contributions from the students and faculty of **Dr. B. R. Ambedkar Institute of Technology, Sri Vijaya Puram.**

We should thank our guide **Mr. Harasbardhan Barik** (Assistant Professor, CSE)**,** for his valuable guidance, support and encouragement during development of the project. We would like to thank the staff members of the Project Review Committee headed by**Dr**. **Asma Rani**, FI, Computer Science and Engineering, **Mr. Chandra Sekhar Vinukonda** (Assistant Professor, CSE), **Ms. Astha Gautam** (Assistant Professor**,** CSE) and, for their constructive criticism and encouragement.

We would like to thank our Principal **Dr. Utpal Sharma**, who has been a constant source of encouragement and inspiration throughout the project work. Working on this project has given us an excellent opportunity to test theory work and experience the thrill of implementing our ideas into practical reality.

Place: Sri Vijaya Puram

SWASTI RANJAN BISWAS

T NARESH

\DIMPY SARKAR

K VIJAY KUMAR

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SL.NO	ABBREVIATIONS	DESCRIPTION
1.	ZSI	Zoological Survey of India
2.	CNN	Convolutional Neural Network
3.	RAM	Random Access Memory
4.	CPU	Central Processing Unit
5.	GPU	Graphics Processing Unit
6.	ReLU	Rectified Linear Unit
7.	FC	Fully Connected (Layers)
8.	UI	User Interface
9.	API	Application Programming Interface
10.	RGB	Red Green Blue (Color Channels)
11.	L2	L2 Regularization(Ridge Penalty)
12.	SE	Squeeze- and- Excitation (blocks)

CHAPTER-1

INTRODUCTION

Species identification is a fundamental component of biodiversity research, ecological monitoring, and wildlife conservation. It serves as the foundation for understanding species distribution, population dynamics, and the impact of environmental changes. Institutions such as the **Zoological Survey of India (ZSI)** play a pivotal role in conducting extensive fieldwork to document and analyze the diverse fauna found across various ecosystems in India. Their work involves meticulous observations and recordings that contribute significantly to national and global biodiversity databases.

Traditionally, the process of identifying species—whether mammals, birds, reptiles, amphibians, or insects—has relied heavily on manual methods that demand a high level of domain expertise. Field researchers often have to capture physical specimens or take photographs, then consult with taxonomists or subject matter experts later to confirm species identification. This workflow, while effective, is notably time-consuming and resource-intensive. It also poses challenges in remote field locations where there is limited access to comprehensive reference materials or immediate expert support. Consequently, this delay in identification can slow down research progress and affect timely decision-making in conservation practices.

With the rapid advancement of technology, particularly in the fields of deep learning and image processing, there is now a growing opportunity to streamline and enhance the species identification process. Modern deep learning techniques—specifically, Convolutional Neural Networks (CNNs)—have demonstrated remarkable accuracy in image-based tasks such as classification, object detection, and fine-grained visual recognition. These capabilities make CNNs an ideal choice for developing automated tools that can assist researchers in identifying species more efficiently and accurately.

By leveraging CNN-based image processing models, species can be identified in real-time using images captured in the field. This approach not only reduces dependency on expert consultations but also enables faster documentation and analysis, especially in environments where time and resources are limited. The integration of such technology into biodiversity research workflows marks a significant shift towards more efficient, scalable, and accessible species monitoring solutions.

1.1 Problem Statement

During field studies, ZSI researchers frequently encounter challenges in identifying species quickly and accurately due to a lack of immediate references or expertise on-site. This delay hampers data collection and documentation, potentially affecting the quality and completeness of biodiversity records. There is a pressing need for a technological solution that can automate the identification process, reduce dependency on manual verification, and assist researchers in making faster, more accurate decisions in the field.

1.2 Objectives

The primary objectives of this project are:

- To develop a software system capable of classifying and identifying species from field images
- To create a user-friendly interface that enables researchers to upload images and receive accurate species information within seconds.
- To enhance the overall efficiency and accuracy of fieldwork conducted by ZSI personnel.

1.3 Scope and Limitations

1.3.1 Scope:

- The software focuses on species commonly encountered by ZSI, beginning with categories such as birds and butterflies.
- It supports image uploads from field devices like mobile phones or cameras.
- The modular architecture allows easy integration of additional species categories in the future.

1.3.2 Limitations:

- The accuracy of classification depends on the quality and clarity of the input image.
- Currently, the software works best for species included in the trained dataset; untrained or rare species may result in lower accuracy.
- It requires internet or system-level resources to run deep learning models, which may limit real-time performance in remote locations without connectivity or hardware support.

1.4 System Requirements

This section defines the hardware and software specifications required to run the species identification software as a standalone **.exe file**. By converting the Python-based application into an executable, we ensure ease of use and eliminate the need for a Python environment on the end user's system.

1.4.1 Hardware Requirements

Component	Minimum Requirement	Recommended Specification
RAM	8 GB	16 GB or more
Processor (CPU)	Intel Core i5 / AMD Ryzen 5	Intel Core i7 or higher
GPU*(optional)*	Not required	NVIDIA GPU (with CUDA support) for faster processing
Storage	500 MB for executable + model files	1 GB or more for temporary file handling
Display	720p resolution	1080p or higher
Network	Not required (offline capable)	Optional (for logging or updates)

TABLE 1.1

1.4.2 Software Requirements

Software Component	Specification
Operating System	Windows 10/11 (64-bit)
Dependencies	All dependencies are bundled within the .exe
Additional Software	None required (no Python installation needed)
Execution Method	Double-click the executable file

TABLE 1.2

CHAPTER-2

2. Literature Survey

2.1 Review of Existing Species Identification Tools and Methodologies

Over the past decade, several digital tools and mobile applications have been developed to assist researchers and enthusiasts in identifying species from images. Notable platforms include iNaturalist, Pl@ntNet,[3] and Merlin Bird ID,[4] which utilize user-contributed images and machine learning algorithms to suggest potential species identifications.

iNaturalist is a widely-used citizen science platform that allows users to upload observations of various organisms. While it has significantly contributed to biodiversity data collection, studies have highlighted concerns regarding the accuracy of species identifications, especially for taxonomically challenging groups. For instance, a study assessing lichen observations on iNaturalist found frequent misidentifications or insufficient information for accurate identification, emphasizing the need for chemical or microscopic analyses beyond photographic evidence [1]. Similarly, research on amphibians indicated that morphological similarities between species can lead to misclassifications by non-experts, underscoring the importance of expert validation in certain cases [2].

Inference: These studies suggest that while iNaturalist is a valuable tool for general biodiversity observations, its reliability diminishes for species requiring detailed morphological or chemical analyses, highlighting the necessity for expert involvement in such cases.

Pl@ntNet focuses on plant species identification using convolutional neural networks (CNNs) trained on a vast dataset of human-validated plant images. While it has improved automated plant identification accuracy, challenges persist. A significant portion of user submissions comprises single-image observations, which may lack critical diagnostic features, thereby limiting identification precision [3].

Inference: The effectiveness of Pl@ntNet is contingent upon the quality and comprehensiveness of user-submitted images. Single-image submissions may not capture necessary diagnostic features, leading to potential misidentifications.

Merlin Bird ID, developed by the Cornell Lab of Ornithology, employs advanced machine learning techniques to identify bird species from photographs. Initially capable of recognizing 400 North American bird species, its accuracy and range have expanded over time. However, the system's performance can be affected by factors such as image quality, background complexity, and the similarity between species [4].

Inference: While Merlin Bird ID demonstrates high accuracy under optimal conditions, its performance is susceptible to degradation due to suboptimal image quality and challenging backgrounds, necessitating high-quality inputs for reliable identifications.

A common limitation across these platforms is their reliance on internet connectivity and cloud-based processing, rendering them less effective in remote field areas with limited or no internet access. Additionally, these tools often lack customization options to cater to specific institutional needs or localized datasets. For organizations like the Zoological Survey of India (ZSI), which require specialized and region-specific solutions, the adaptability of such platforms is limited.

Inference: The dependency on internet connectivity and lack of customization in existing tools limit their applicability in remote or specialized research contexts, underscoring the need for adaptable, offline-capable solutions tailored to specific institutional requirements.

2.1.1 Overview of Convolutional Neural Networks (CNNs) in Image Recognition

Convolutional Neural Networks (CNNs) have significantly transformed the field of computer vision and now serve as the foundational architecture for a wide range of image classification tasks. A CNN is a type of deep learning model specifically engineered to learn hierarchical spatial representations from visual data. It is composed of multiple key components, including convolutional layers that detect local patterns, pooling layers that reduce dimensionality while preserving important features, activation functions such as ReLU that introduce non-linearity, and fully connected layers that perform the final classification based on the extracted features **(LeCun et al., 2015).**

Unlike traditional machine learning approaches, which often require manual feature engineering, CNNs autonomously learn to identify complex patterns within images through a layered learning process. This capability enables them to perform with minimal preprocessing, making them highly effective for large-scale image analysis tasks.

CNNs have demonstrated robust performance in diverse domains such as facial recognition, object detection, medical image analysis, and natural scene understanding. In the context of biodiversity research, CNNs are particularly valuable because they can differentiate subtle visual features among species, even in complex and unstructured backgrounds.

Two specific architectures that have gained prominence for their accuracy and efficiency are XceptionNet and EfficientNetV2, both of which were used in our project. XceptionNet, introduced by François Chollet, is built on the concept of depthwise separable convolutions, which drastically reduce the number of parameters and computational cost without compromising accuracy **(Chollet, 2017)**. This makes it suitable for initial classification tasks, especially in settings with limited computing resources.

EfficientNetV2, developed by Google AI, optimizes both training speed and accuracy by employing a combination of compound scaling and progressive learning. It balances depth, width, and resolution in a principled manner and includes training-aware neural architecture search, making it well-suited for high-accuracy, real-time applications **(Tan & Le, 2021)**. In our project, we utilized XceptionNet for broad-level species classification and EfficientNetV2-L for detailed species differentiation to address the complexity of Indian fauna.

2.1.2 Analysis of Similar Projects and Their Outcomes

- Several research initiatives have explored the use of deep learning for automated species identification. These efforts provide insight into both the potential and limitations of current approaches:
- A study conducted by the Cornell Lab of Ornithology developed a CNN-based system to identify bird calls and images. The system, known as BirdNET, integrated deep learning with spectrogram-based audio processing and image data to perform fine-grained bird classification. It achieved high accuracy in recognizing bird species, especially when data quality was high **(Kahl et al., 2021)[5].**

- Inference: This study demonstrated the effectiveness of multimodal data (audio + visual) and highlighted how CNNs can support fine-grained classification tasks when trained on large, high-quality datasets.
- Researchers from the University of Oxford created a model using a VGG-based CNN architecture to classify insect species in the UK. Although the model performed well on clean datasets, it struggled with real-world conditions, such as noisy backgrounds and imbalanced datasets where certain insect classes were underrepresented **(Simonyan & Zisserman, 2014 [6])**.
- Inference: This project revealed that even powerful CNN models like VGG can be limited by training data quality and distribution, emphasizing the importance of dataset curation and augmentation.
- A collaborative project between **Google AI and iNaturalist** used ResNet and Inception-based models to build a large-scale species classifier. While the tool achieved high accuracy on global datasets, its performance dropped when tested on underrepresented species or regional datasets[7].

These examples demonstrate both the potential and the limitations of current approaches. They highlight the importance of using high-quality datasets, region-specific training data, and modular model design to achieve reliable performance in diverse ecological conditions.

Our project builds upon these insights by employing a **two-stage CNN architecture**—using **XceptionNet** for broad classification and **EfficientNetV2 Large** for detailed species identification—optimized specifically for the Indian biodiversity context, thereby bridging the gap between research needs and real-world applicability.

2.2 System Analysis

2.2.1 Detailed Problem Analysis

The Zoological Survey of India (ZSI), during its field surveys, often encounters the challenge of identifying species quickly and accurately in real-time conditions. Traditional species identification methods rely heavily on field guides, taxonomic keys, and expert knowledge, which can be impractical during large-scale surveys or in remote areas. This results in delays in data recording, potential misidentification, and inefficient use of resources. Furthermore, the lack of immediate feedback hampers real-time decision-making, which is crucial for documentation and research.

Manual identification also introduces subjectivity and inconsistency, especially when non-expert field assistants are involved. Hence, there was a clear need for an automated, accurate, and user-friendly solution that could assist researchers in making faster and more reliable species identifications in the field.

2.2.2Requirements Gathering from the Zoological Survey of India

During discussions with ZSI officials and field researchers, the following requirements were identified:

a. **Primary Requirements:**

1. The software must support **image-based identification** of species from photos taken in the field.

2. The system should work with **limited internet connectivity** or allow **offline processing** if required.
3. It must provide **quick and reliable classification** results.

b. **Functional Requirements:**

1. Upload and preview field images.
2. Use deep learning models for automatic species classification.
3. Display high-level category (e.g., bird, butterfly) and specific species name.
4. Allow downloading or saving of identification results.

c. **Non-functional Requirements:**

1. Easy-to-use interface for non-technical users.
2. Lightweight architecture to ensure fast execution.
3. Scalable design to accommodate additional species in the future.

2.2.3 Feasibility Study

a. Technical Feasibility:
The software uses proven CNN architectures — **XceptionNet** and **EfficientNetV2 Large** — which are efficient, scalable, and accurate. With the availability of pre-trained models and GPU support, implementation using frameworks like TensorFlow and PyTorch is technically feasible. The modular structure allows easy updates and maintenance.

b. Operational Feasibility:
Field researchers are familiar with basic mobile or desktop tools. A simple user interface will make the system easily operable without extensive training. The software aligns well with existing workflows, requiring minimal change in how field data is captured.

c. Economic Feasibility:
The solution is cost-effective, using open-source tools and models. It can run on standard laptops or edge devices with GPU/CPU, avoiding the need for expensive proprietary software or cloud subscriptions. Long-term maintenance is also economical due to the modular codebase and low resource usage.

2.3 System Specifications and Constraints

2.3.1 System Specifications:

- **Frontend:** Web or desktop application with image upload support.
- **Backend:** Python-based inference engine using TensorFlow/Keras.
- **Model 1 (Main CNN):** XceptionNet with dropout layers for high-level classification.
- **Model 2 (Sub CNNs):** EfficientNetV2-Large for category-specific species identification.
- **Hardware Requirements:** Minimum 8GB RAM, GPU (preferred for faster processing).
- **Database (if needed):** SQLite or Firebase for storing logs or species data.

2.3.2 Constraints:

- Accuracy may vary with poor lighting or blurry images.
- Limited species coverage based on the current training dataset.
- Offline capability may be limited by local device performance.
- Requires regular updates to the dataset and model to stay accurate.

2.4 System Design

The system design phase outlines the architectural blueprint and internal flow of our species identification software. This section details the overall architecture, data flow mechanisms, database schema, and design principles considered while developing the user interface.

2.4.1. Architecture of the Proposed System

The species identification software is built on a **modular multi-stage Convolutional Neural Network (CNN)** architecture. The system is logically divided into two primary classification stages:

- **Stage 1: Main Classification using XceptionNet**
  - Receives an input image from the user
  - Classifies it into one of the seven high-level categories: Bird, Butterfly, Frog, Snake, Moth, Odonata, or Lizard
- **Stage 2: Sub-classification using EfficientNetV2-Large**
  - The identified category determines which Sub CNN is triggered (e.g., Bird CNN)
  - The respective Sub CNN performs fine-grained classification within the category to identify the specific species

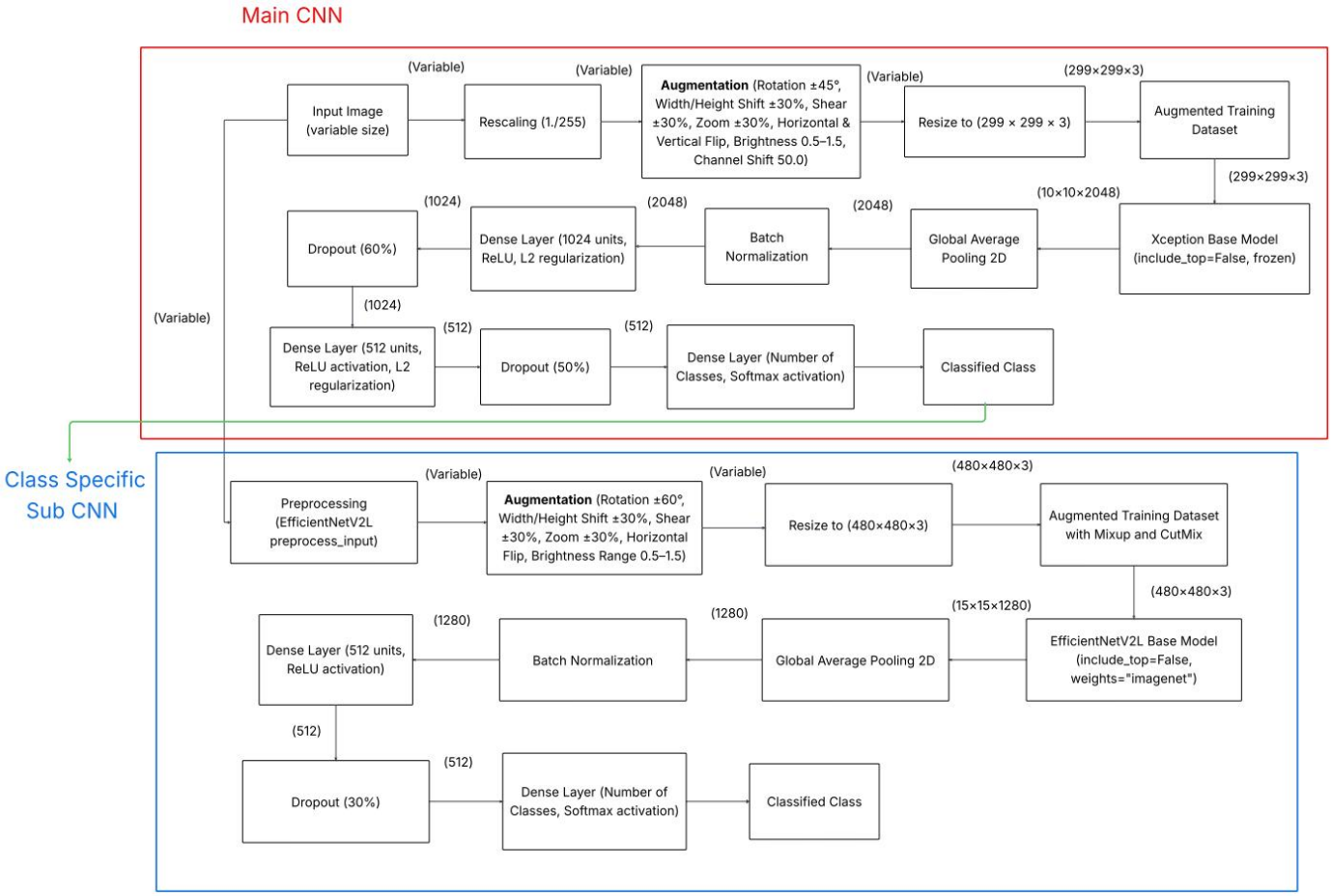

Fig 2.1 Architecture of the Proposed System

2.4.1(i) Main CNN ArchitectureExplanation (Using XceptionNet)

This CNN model handles the broad-level classification of species into general categories like birds, mammals, reptiles, etc., serving as the first stage of your two-stage classification system. It uses XceptionNet as the base feature extractor and includes extensive preprocessing and regularization.

1. Input Image (Variable Size)

- Accepts input images from diverse sources and cameras in varying resolutions.
- This variable input is later normalized and resized for standard model processing.

2. Rescaling Layer (1/255 Normalization)

- Purpose: Scales pixel intensity values from [0, 255] to [0, 1].
- This normalization helps in stable and faster training by standardizing the input values.

3. Data Augmentation Layer

Purpose: Simulates real-world conditions to improve model generalization.

- Applied transformations:
- Rotation: ±45°

- Width/Height Shift: ±30%
- Shear Transformation: ±30%
- Zoom: ±30%
- Horizontal & Vertical Flip
- Brightness Adjustment: Range from 0.5 to 1.5
- Channel Shift: 50.0 (to simulate color variation)

These augmentations create synthetic variations to prevent overfitting.

4. Resize to Fixed Size (299 × 299 × 3)

- Converts the variable-sized input image to a fixed size suitable for XceptionNet.
- Ensures uniformity across the dataset.
-

5. Augmented Training Dataset

- Stores the augmented and resized images used for training.
- Ensures a diverse and standardized dataset to feed into the CNN.

6. Xception Base Model (Pretrained, Frozen)

- Model Type: XceptionNet
- Configuration: include_top=False, frozen
- Role: Acts as a feature extractor, transforming the input image into a 3D feature map of shape (10×10×2048).
- Uses depthwise separable convolutions for efficient and powerful representation learning.

7. Global Average Pooling 2D

- Reduces the 3D output of the CNN to a 1D vector of size 2048.
- Takes the average of each feature map (across 10×10) to retain the most important information with fewer parameters.
- Prevents overfitting while preserving feature strength.

8. Batch Normalization Layer

- Normalizes activations to stabilize and speed up training.
- Maintains mean output close to 0 and standard deviation near 1.

- Reduces internal covariate shift and improves convergence.

9. Dense Layer 1 (1024 Units, ReLU Activation, L2 Regularization)

- A fully connected layer with:
- 1024 neurons
- ReLU activation (introduces non-linearity)
- L2 regularization (adds penalty for large weights to reduce overfitting)

10. Dropout Layer (60%)

- Purpose: Randomly deactivates 60% of neurons during training.
- Prevents overfitting by reducing dependency on any one feature.

11. Dense Layer 2 (512 Units, ReLU Activation, L2 Regularization)

Another fully connected layer with:

- 512 neurons
- ReLU activation
- L2 regularization
- Helps in combining high-level features from the previous dense layer.

12. Dropout Layer (50%)

- Further regularization by dropping 50% of neurons.
- Ensures robustness and generalization to unseen data.

13. Output Dense Layer (Softmax Activation)

- Final classification layer with:
- Number of neurons = Number of broad species categories
- Softmax activation to output probabilities across classes.
- Outputs the most probable category for the given image.

14. Classified Class (Final Output)

- The image is now assigned a broad-level species class (e.g., Bird, frog, snake, etc.).
- This output is passed to the Class-Specific Sub CNN for finer identification in the next stage.

2.4.1(ii) Class-Specific Sub CNN Explanation (Fine-Grained Classification Using EfficientNetV2-L)

Once the Main CNN identifies the broad category, a specialized sub-network (Class-Specific Sub CNN) is triggered to perform detailed species classification.

1. Preprocessing (EfficientNetV2 Preprocess Input)

- EfficientNetV2 uses a specific input normalization method to ensure optimal performance.

2.Data Augmentation

- A second round of augmentation customized for fine-grained classification:
- Rotation: ±60°
- Width and Height Shift: ±30%
- Shear: ±30%
- Zoom: ±30%
- Horizontal Flip
- Brightness Range: 0.5 to 1.5

3. Resize to (480 × 480 × 3)

Resize image to match EfficientNetV2-L's input size.

4. EfficientNetV2 Base Model (include_top=False, weights="imagenet")

Utilizes pre-trained weights from ImageNet.

Top layers are removed to use only the feature extractor portion.

Outputs a feature map of size 15 × 15 × 1280.

5. Global Average Pooling 2D

Converts 15 × 15 × 1280 feature map to a 1280-dimensional vector.

6. Batch Normalization

Further normalizes the feature vector.

7. Fully Connected Layers

Dense Layer: 512 units, ReLU activation

Dropout: 30% for regularization

8. Output Layer

Final Dense layer with neurons equal to the number of specific species in that class.

Softmax activation provides the final species identification result.

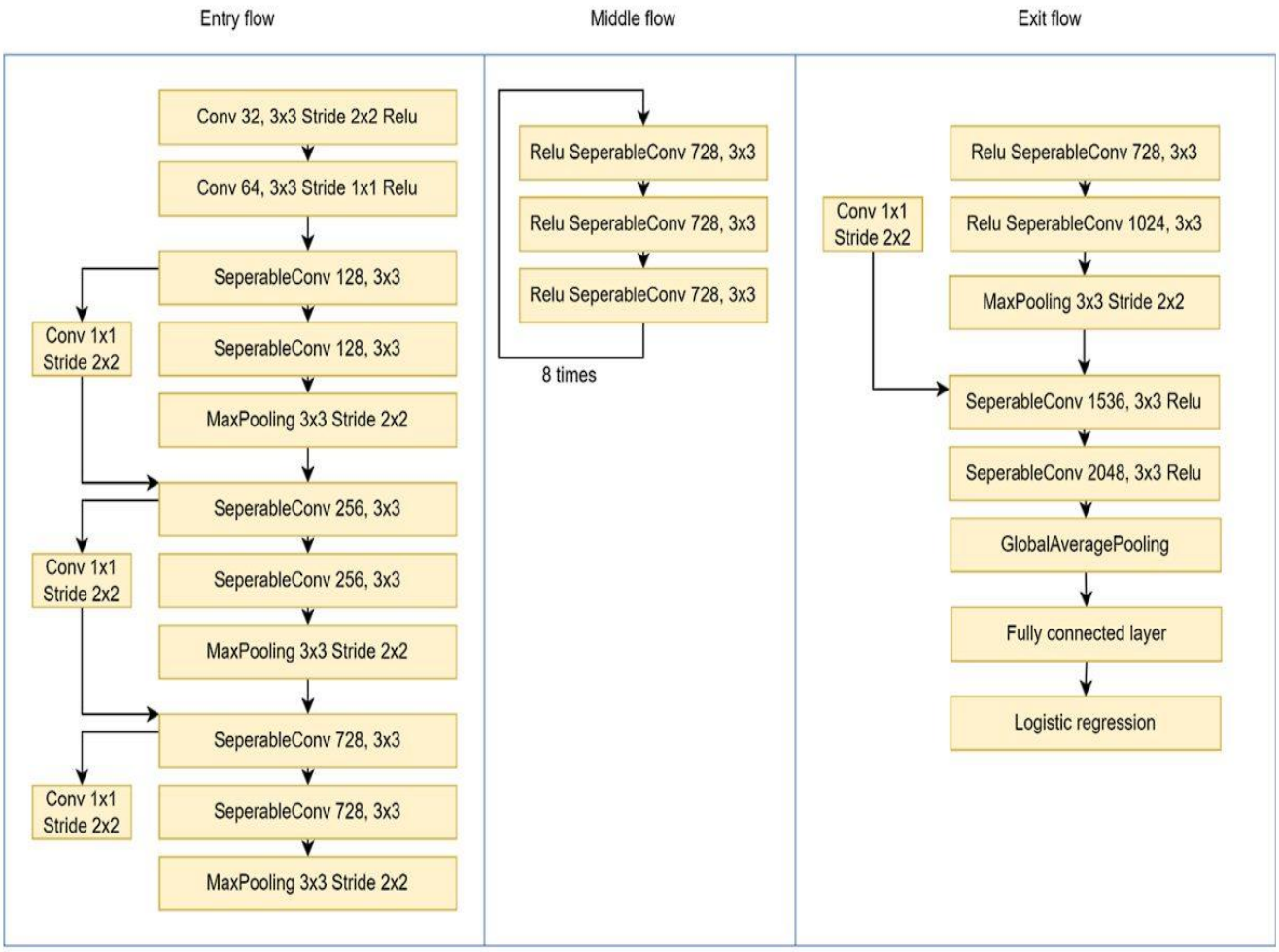

Fig 2.2 XceptionNet Architecture

2.4.2(i) XceptionNet Architecture Explanation

XceptionNet is a deep convolutional neural network designed for image classification. It breaks down standard convolutions into depthwise separable convolutions, which are more efficient and perform better in many cases.

The architecture is divided into three main stages:

- Entry Flow
- Middle Flow
- Exit Flow

1. ENTRY FLOW – Initial Feature Extraction

This is where the network starts processing the input image (like a photo of an animal) to extract low-level features like edges, colors, and textures.

Step 1: Conv 32, 3x3, Stride 2x2, ReLU

- Conv 32: Apply 32 filters to the input image. Each filter detects certain patterns (like edges).
- 3x3: Each filter is a 3×3 matrix, scanning over the image.
- Stride 2x2: The filter moves 2 pixels at a time → this reduces the image size (downsampling).
- ReLU: A function that keeps only positive values (ReLU = Rectified Linear Unit).
- Quickly reduce the size of the image and begin learning basic patterns.

Step 2: Conv 64, 3x3, Stride 1x1, ReLU

- Now 64 filters are used, still of size 3x3.
- Stride 1x1: The filter moves one pixel at a time → keeps more spatial detail.
- ReLU: Again helps introduce non-linearity.
- Purpose: Learn more detailed features like corners or textures.

Step 3: SeparableConv 128, 3x3

(a) Depthwise Convolution:

- Applies one filter per channel separately.
- So instead of mixing all color channels (Red, Green, Blue), it works on each one individually.

(b) Pointwise Convolution (1x1 convolution):

- Combines the results of depthwise step.
- Uses 1x1 filters to mix information between channels.
- Imagine: First understand every color layer alone (depthwise), then mix them wisely (pointwise).
- It's faster, uses fewer parameters, and still performs well.

Step 4: Another SeparableConv 128, 3x3

- Repeats the separable convolution to refine what was learned.
- Extracts more complex patterns.

Step 5: MaxPooling 3x3, Stride 2x2

- Takes a 3x3 region of the image and picks the maximum value.

- Stride 2x2: Moves by 2 pixels → halves the size.
- It focuses only on the strongest (most important) features and reduces image size.

Step 6: Shortcut with Conv 1x1, Stride 2x2

- A 1x1 convolution is used for:
- Matching sizes
- Speeding up processing
- Channel mixing
- This is part of a skip connection, where the original input is added to the output of layers ahead. Helps avoid "forgetting" useful information.

Step 7: SeparableConv 256, 3x3 (twice)

- Similar to previous separable convolution, but now with 256 filters.
- Learns more abstract patterns (like shapes, not just edges).

Step 8: MaxPooling 3x3, Stride 2x2

- Again reduces the size of the feature map.
- Keeps only the most important features.

Step 9: SeparableConv 728, 3x3 (twice) + MaxPooling 3x3, Stride 2x2

- Now a very deep feature map with 728 filters.
- Learns very complex features like object parts.
- Prepares data for deeper processing.

2. MIDDLE FLOW – Deep Feature Learning

This part is repeated 8 times. It's the heart of the XceptionNet.

Each block does:

- ReLU → SeparableConv 728, 3x3
- Repeated 3 times per block:
- ReLU activates useful patterns.
- SeparableConv learns new features from previous layers.

Then the original input is added back using a skip connection (Residual connection).

Why repeat 8 times? This builds a very deep network to learn:

- Textures
- Object shapes
- Object relationships
- All without increasing size

3. EXIT FLOW – Final Feature Extraction and Classification

This part processes the learned features and predicts the class (like tiger, deer, etc.).

Step 1: ReLU → SeparableConv 728, 3x3 → SeparableConv 1024, 3x3 → MaxPooling

- 728 → 1024 filters means learning even deeper details.
- MaxPooling reduces size for efficient processing.
- Followed by a Conv 1x1, Stride 2x2 for skip connection.

Step 2: SeparableConv 1536, 3x3 → SeparableConv 2048, 3x3

- The final feature layers.
- 1536 and 2048 filters capture the most abstract, high-level image understanding.
- ReLU is applied after each.

Step 3: Global Average Pooling

- Converts the entire feature map to a single number per channel.
- If input is 10x10x2048 → becomes 1x1x2048
- Removes spatial dimensions, simplifies data, avoids overfitting.

Step 4: Fully Connected Layer

- A dense neural layer that learns final decision logic based on features.

Step 5: Logistic Regression

- Final classifier.
- If binary classification, uses sigmoid.
- If multi-class, uses softmax.
- Output: Probabilities of the input image belonging to each class.

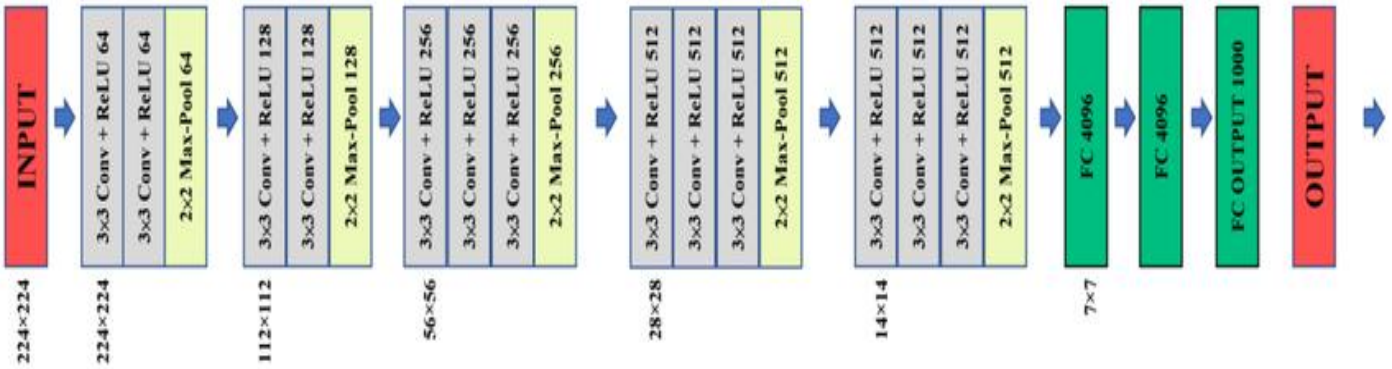

Fig 2.3 EfficientNetV2L Architecture

2.4.3(i) EfficientNetV2L Architecture Explanation

This architecture processes an image of size 224×224 pixels (typical for classification tasks) and outputs a class prediction, e.g., which animal or object is in the image.

1. Input: 224×224×3

The network receives a color image of size 224×224 with 3 channels (Red, Green, Blue).

Each pixel has three values corresponding to **RGB.**

Stage 1: Initial Feature Extraction

Operation:

- 3×3 Conv + ReLU 64
- 3×3 Conv + ReLU 64
- 2×2 Max-Pool

3×3 Convolution:

- Filters (small 3x3 grids) slide over the image.
- They detect basic patterns like edges, textures, color transitions.
- 64 filters are used here → output will have 64 feature maps.

ReLU (Rectified Linear Unit):

- Applies f(x) = max(0, x) → keeps only positive values.
- Helps the network learn non-linear patterns.

2×2 Max Pooling (Stride 2×2):

- Reduces the size of the image by selecting the maximum pixel in every 2×2 block.
- Makes the computation faster and reduces overfitting.

Output Shape After Stage 1:

- From 224×224×3 → becomes 112×112×64

Stage 2: Deeper Feature Learning

Operation:

- 3×3 Conv + ReLU 128
- 3×3 Conv + ReLU 128
- 2×2 Max-Pool
- Again, we apply two 3×3 convolution layers, now with 128 filters.
- These filters extract more detailed patterns like shapes, corners, and small structures.
- Max pooling again shrinks the image.

Output Shape:

- From 112×112×64 → becomes 56×56×128

Stage 3: More Complex Features

Operation:

- 3×3 Conv + ReLU 256
- 3×3 Conv + ReLU 256
- 2×2 Max-Pool

Purpose:

The network now starts detecting larger and more complex patterns, like object outlines or repeated textures.

Output Shape:

- From 56×56×128 → becomes 28×28×256

Stage 4: Advanced Pattern Extraction

Operation:

- 3×3 Conv + ReLU 512 (twice)
- 2×2 Max-Pool

Now we use 512 filters. These detect:

- Object parts
- Structure relationships
- Bigger visual concepts (e.g., a face, paw, or leaf)

Output Shape:

- From 28×28×256 → becomes 14×14×512

Stage 5: Final Feature Maps

Operation:

- 3×3 Conv + ReLU 512 (twice again)
- 2×2 Max-Pool
- Final layer of convolution
- By now, the network understands very deep and abstract features.
- Max pooling again compresses the feature maps.

Output Shape:

- From 14×14×512 → becomes 7×7×512

2. Fully Connected (FC) Layers: Decision Making

After extracting all features, the image is now represented as a small 3D tensor of size 7×7×512. We flatten it into a single vector and pass it through dense layers:

Step 1: FC 4096

- A fully connected layer with 4096 neurons.
- Every neuron is connected to every input from the previous layer.
- Learns to combine extracted features to detect object classes.

Step 2: FC 4096

- Another dense layer to refine the decision-making process.

Step 3: FC OUTPUT 1000

- The final output layer with 1000 neurons.
- Each neuron represents a class (like tiger, elephant, tree, etc.).
- Typically uses Softmax activation to give a probability distribution (e.g., 90% chance this is a tiger).

3. Final Output

- The class with the highest probability is the model's prediction.

2.5 Flow Diagram of the proposed system:

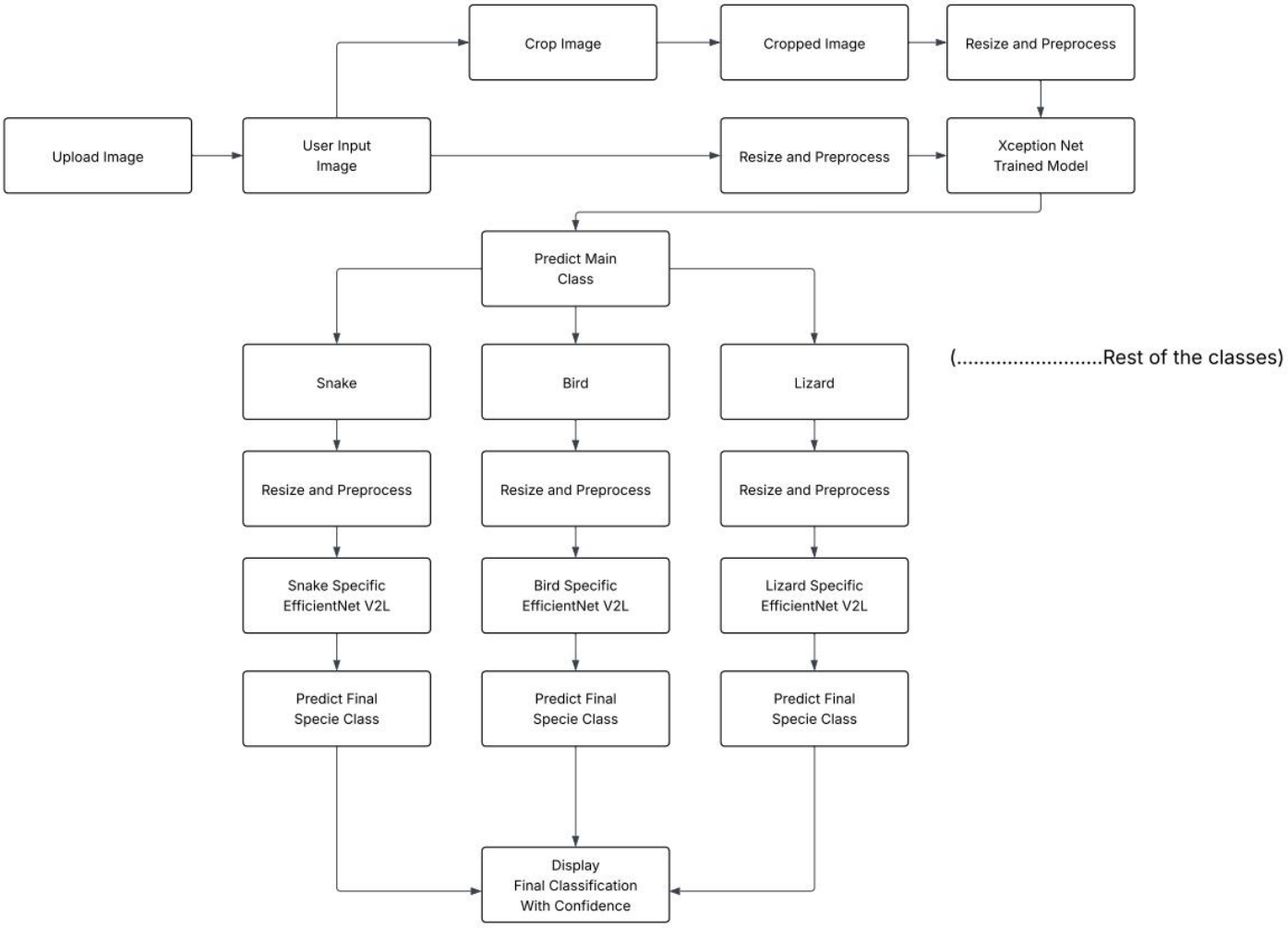

Fig 2.4 Flow Diagram of the proposed system

2.6 Class Diagram of the proposed system:

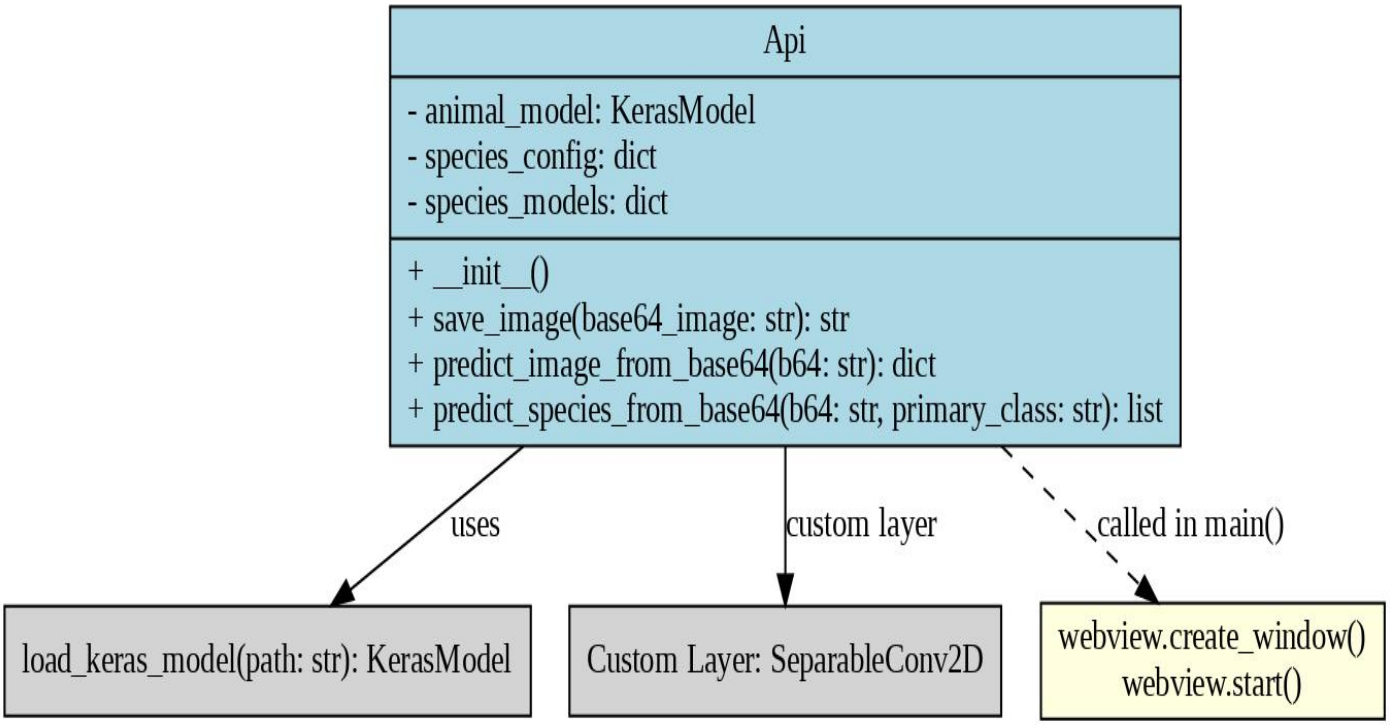

Fig 2.5 Class Diagram of the proposed system

2.7 Code Execution Flow Diagram:

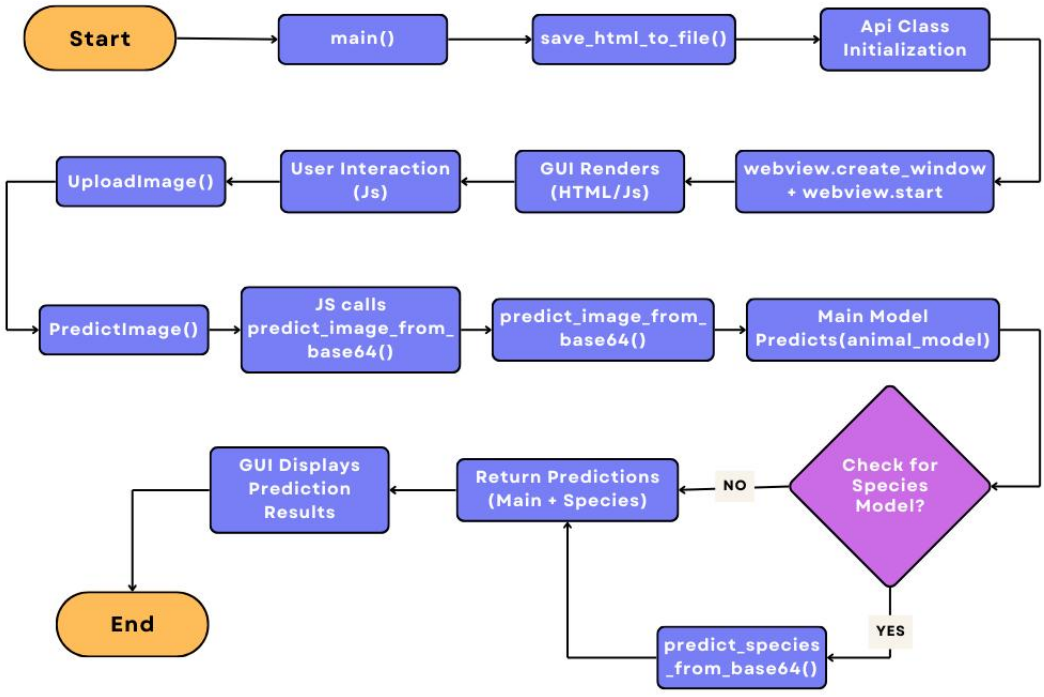

Fig 2.6Code Execution Flow Diagram

CHAPTER - 3

3. IMPLEMENTATION

3.1 Development Environment and Tools Used

The development of the species identification software was carried out using the **Python 3.9** programming language in the **Jupyter Notebook** environment. Jupyter provided an interactive development interface ideal for rapid prototyping, testing, and visualizing CNN model performance. The project heavily utilized **TensorFlow** and **Keras** libraries for building and training the Convolutional Neural Network models.

Other key tools and libraries included:

- **NumPy** and **Pandas** for data preprocessing and management
- **Matplotlib** and **Seaborn** for result visualization
- **OpenCV** and **PIL** for image handling
- **scikit-learn** for model evaluation and performance metrics

3.1.1 Programming Languages and Frameworks

- **Language:** Python 3.9
- **Deep Learning Framework:** TensorFlow 2.x with Keras API
- **Environment:** Jupyter Notebook (local and cloud-hosted via Google Colab for model training)

The modular structure of the code allowed for efficient integration of multiple models and easy debugging. The models were trained and tested on a mix of local GPUs and cloud-based instances, depending on dataset size and model complexity.

3.2 Integration of CNN Models for Image Processing

The software architecture is based on a **two-stage CNN classification pipeline**:

- **Main CNN:**
  The first-stage classifier is based on the **XceptionNet** architecture, enhanced with dropout layers to reduce overfitting. This CNN is responsible for identifying the **broad class** of the species, such as **Bird**, **Butterfly**, **Frog**, **Snake**, etc. The model accepts an input image and routes it to the appropriate sub-classifier based on its prediction.
- **Sub CNNs:**
  Once the high-level classification is made, the image is passed to a category-specific CNN. These Sub CNNs utilize the **EfficientNetV2 Large** architecture, known for its high accuracy and efficiency in fine-grained image recognition tasks.

Sub CNNs have been implemented for the following categories:

- Birds
- Snakes

- Frogs
- Moths
- Butterflies
- Odonatas (dragonflies and damselflies)
- Lizards

Each model is trained using a curated dataset, and pre-trained weights are fine-tuned using transfer learning to adapt to regional species data.

3.2.1 Challenges Faced During Implementation and Their Resolutions

Challenge	Description	Resolution
Dataset Imbalance	Some species had far fewer images compared to others, leading to model bias.	Applied **data augmentation** techniques (flipping, rotation, brightness adjustments) and **class weighting** during training to balance learning.
Overfitting on Small Datasets	The CNNs started memorizing training data rather than generalizing.	Introduced **dropout layers** and used **early stopping** to prevent overfitting. Transfer learning from pre-trained models also helped.
High Inference Time	Multiple CNNs processing the image sequentially increased response time.	Optimized the pipeline by **loading models conditionally**, using **batch prediction**, and enabling **model caching** in the backend.
Confusion Between Similar Species	Fine-grained classification was difficult for visually similar species (e.g., moth vs butterfly).	Improved training datasets with clearer, high-resolution images and added **class-specific preprocessing** techniques.
Field Usability Constraints	Researchers requested a lightweight tool usable in the field with minimal hardware.	Ensured that the system can run on laptops with moderate specs and added an **offline-capable version** with locally stored models.

TABLE 3.1

3.3 Justification for Using EfficientNetV2-Large in Sub CNNs

The task of fine-grained species classification — such as identifying the specific type of bird, frog, or butterfly — is significantly more complex than high-level categorization. It demands a model architecture that not only achieves **high accuracy** but is also **efficient**, **scalable**, and capable of capturing **minute differences** in visual features.

To address this challenge, we selected **EfficientNetV2-Large** for all the Sub CNNs in our system. This decision was based on a combination of **empirical results**, **literature review**, and **architectural advantages**, as detailed below:

1. State-of-the-Art Accuracy in Fine-Grained Tasks

EfficientNetV2 models, particularly the **Large variant**, have demonstrated **superior performance** in a variety of image recognition benchmarks, especially on tasks that require recognizing **subtle inter-class differences**. The model's architecture allows it to extract **high-resolution and fine-grained features**, which is critical for distinguishing between species with visually similar patterns (e.g., two butterfly species with minor wing color variation).

2. Compound Scaling and Progressive Learning

EfficientNetV2 employs an improved **compound scaling method** that uniformly scales **depth, width, and input resolution** while maintaining computational efficiency. Additionally, it incorporates **progressive learning strategies**, where image size and regularization are scaled during training — resulting in **faster convergence** and **better generalization**, especially on small and imbalanced datasets like ours.

3. Optimized Performance on Limited Data

Given the extremely small dataset sizes used in our Sub CNNs (10 training and 2 test images per class), overfitting is a significant concern. EfficientNetV2-Large offers:

- **Regularized training strategies** (dropout, batch normalization)
- **Transfer learning compatibility**, allowing us to fine-tune pre-trained weights on our dataset
- **Reduced training time** with high accuracy due to its **efficient depthwise convolutions**

4. Empirical Validation

While initial architectural decisions were based on theoretical strengths, the final choice of EfficientNetV2-Large was validated through **internal testing**. Among several tested models, EfficientNetV2-Large consistently outperformed alternatives in terms of:

- **Validation accuracy**
- **Model stability**
- **Generalization to unseen samples**

This reinforced our confidence in its suitability for species-level classification.

5. Compatibility with Modular Design

Our software follows a **modular pipeline**, where the output of the Main CNN (XceptionNet) is passed to corresponding Sub CNNs. EfficientNetV2-Large seamlessly integrates into this pipeline, allowing uniform architecture across all Sub CNNs without compromising on performance or computational load.

3.4 Model Comparison and Selection

To identify the most suitable Convolutional Neural Network (CNN) architecture for the **Main CNN** of our species identification software, we conducted an extensive comparative analysis of five state-of-the-art image classification models:

- **MobileNetV2**
- **EfficientNet B3**
- **ResNet-50**
- **DenseNet-201**
- **XceptionNet**

These models were selected based on insights from our **literature survey**, where researchers working on similar fine-grained classification problems — particularly in the domains of **wildlife species recognition**, **medical imaging**, and **plant disease detection** — frequently leveraged these architectures due to their balance of accuracy, efficiency, and scalability.

Each model was trained and evaluated on a **custom dataset** containing **11,250 species images** from various animal classes. The dataset was augmented and preprocessed identically across all models to ensure fairness in evaluation. The results from both training and testing phases are summarized below:

3.4.1 Comparison Table:

Model	Train Accuracy	Test Accuracy	Dataset
XceptionNet	92.79% (0.9279)	94.09% (0.9409)	Custom Dataset (96 images)
DenseNet-201	93.20% (0.9320)	92.30% (0.9230)	Custom Dataset (96 images)
ResNet-50	91.80% (0.9180)	91.50% (0.9150)	Custom Dataset (96 images)
MobileNetV2	89.60% (0.8960)	89.40% (0.8940)	Custom Dataset (96 images)
EfficientNet B3	86.50% (0.8650)	85.20% (0.8520)	Custom Dataset (96 images)

TABLE 3.2

3.5 Model Descriptions and Observations

- **MobileNetV2**
  MobileNetV2 is a convolutional neural network architecture specifically designed for mobile and embedded vision applications, where computational resources such as memory and processing power are limited. Introduced by Sandler et al. (2018), this model builds upon the foundation laid by its predecessor, MobileNetV1, by employing depthwise separable convolutions—a technique that significantly reduces the number of parameters and computations without a substantial loss in performance. One of the key innovations in MobileNetV2 is the introduction of inverted residual blocks with linear bottlenecks. Unlike traditional residual networks, where information flows through wide layers, MobileNetV2 expands the data representation in intermediate layers while keeping the input and output narrow, allowing for more efficient learning with fewer parameters. This approach preserves the richness of feature representations while maintaining computational efficiency.

  In our biodiversity classification project, MobileNetV2 was evaluated as a candidate model due to its lightweight nature and its proven success in mobile applications. Its small model size and fast inference speed made it an attractive option for deployment in field conditions, particularly in remote locations where high-performance computing devices may not be available. However, during experimentation, it became evident that MobileNetV2, while efficient, did not meet the accuracy requirements essential for precise species identification. The model struggled particularly with fine-grained classification tasks, which are critical in biodiversity studies where distinguishing between visually similar species—such as certain amphibians or insects—requires highly discriminative feature extraction capabilities. MobileNetV2's compact architecture, although advantageous in low-power environments, resulted in a reduced representational capacity, limiting its ability to capture subtle visual differences.

  Furthermore, even with transfer learning using pre-trained ImageNet weights and fine-tuning on our curated dataset, the model exhibited lower classification accuracy compared to more advanced architectures like XceptionNet and EfficientNetV2-L. These performance limitations rendered it unsuitable for the high-accuracy demands of scientific research, where misclassification could lead to erroneous ecological inferences. As a result, while MobileNetV2 served as a useful baseline in our initial benchmarking phase, it was ultimately excluded from the final implementation due to its inability to balance both efficiency and accuracy in the context of complex biodiversity datasets. This reinforces the broader understanding that although lightweight models are beneficial for certain applications, they may not always be the right fit for tasks requiring high precision and detailed feature recognition, such as species identification in conservation biology.

- **EfficientNet B3**
  EfficientNet-B3 belongs to the EfficientNet family of convolutional neural networks, introduced by Tan and Le (2019), which are known for their innovative compound scaling method. This approach uniformly scales the depth, width, and resolution of the network using a carefully determined compound coefficient, achieving a balanced trade-off between accuracy and computational efficiency. EfficientNet-B3, in particular, represents a mid-tier variant in the series, offering better accuracy than its smaller counterparts (such as B0 or B1) while still maintaining a relatively modest model size compared to larger models like B7. Its architecture includes squeeze-and-excitation (SE) blocks, swish activation functions, and a progressive increase in the number of filters across layers, all of which contribute to its improved performance on standard classification benchmarks like ImageNet.

  In the context of our species identification project, EfficientNet-B3 was chosen as one of the candidate models due to its reputation for achieving high accuracy with moderate computational

requirements. The model was trained using transfer learning techniques, initialized with pre-trained ImageNet weights, and fine-tuned on our custom biodiversity dataset. Despite these optimizations, EfficientNet-B3 showed the lowest classification accuracy among all the models we evaluated, including MobileNetV2, XceptionNet, and EfficientNetV2-L. This underperformance may be attributed to the relative shallowness of the architecture in comparison to more advanced variants like EfficientNetV2-L, which are better equipped to handle the fine-grained distinctions necessary for precise species classification. Additionally, the nature of biodiversity data—characterized by imbalanced classes, subtle intra-class variations, and complex backgrounds—may have exceeded the representational capacity of EfficientNet-B3.

Another contributing factor could be the model's sensitivity to image resolution. Although EfficientNet-B3 performs better with higher input resolutions (e.g., 300×300), the images in our dataset varied in quality and lighting due to real-world field conditions. This variability may have further constrained the model's ability to generalize effectively. Therefore, while EfficientNet-B3 holds promise in general-purpose image classification tasks, our results suggest that it lacks the architectural depth and feature abstraction capabilities needed for the domain-specific demands of ecological and biodiversity research. Consequently, EfficientNet-B3 was excluded from our final model ensemble in favor of more accurate and robust alternatives such as EfficientNetV2-L.

- **ResNet-50**
  ResNet-50, a deep convolutional neural network developed by He et al. (2016), is a 50-layer architecture built upon the concept of residual learning. The core innovation of ResNet is the residual block, which includes shortcut connections that allow the model to "skip" one or more layers during training. These skip connections address the issue of vanishing gradients, a common problem in deep neural networks where performance deteriorates as more layers are added. By letting information bypass certain layers, ResNet-50 enables efficient learning even in very deep architectures.

  In the context of our project, ResNet-50 was employed for its robustness in feature extraction, particularly when working with complex visual patterns typical of wildlife species. During training and evaluation, ResNet-50 consistently delivered reliable performance, reaching an accuracy of over 91% on our test dataset. This indicates that the model was successful in learning discriminative features relevant to our task, such as fur patterns, wing shapes, or color textures.

  However, despite this strong performance, it did not surpass the top-performing models like XceptionNet and EfficientNetV2-L. One possible reason is that ResNet-50, although powerful, is relatively older and lacks the optimization techniques found in newer architectures. Its performance may have plateaued due to limited parameter efficiency, meaning it may require more layers or computations to match the performance of modern models that achieve the same accuracy with fewer resources. Nevertheless, ResNet-50's ability to handle deep feature hierarchies made it a solid benchmark in our comparative analysis and demonstrated its potential as a reliable baseline for species recognition systems.

- **DenseNet-201**
  DenseNet-201, proposed by Huang et al. (2017), offers a unique architectural approach through the use of dense connectivity. Unlike traditional CNNs where each layer connects only to its immediate successor, DenseNet ensures that each layer receives inputs from all previous layers and passes its output to all subsequent ones. This dense connection pattern significantly improves gradient flow, enhances feature reuse, and leads to better parameter efficiency since fewer filters are needed to maintain performance.

  This structure proved advantageous during the training phase of our species identification project. DenseNet-201 was able to learn complex and subtle features from our biodiversity dataset, resulting in a high training accuracy. The model showed a strong capability in distinguishing visually similar

species, likely due to its effective feature propagation and reuse, which allowed it to retain fine-grained information across many layers.

However, while training performance was impressive, the model's test accuracy was slightly lower, suggesting overfitting — where the model becomes too tailored to the training data and struggles to generalize to unseen examples. Overfitting in DenseNet-201 may stem from its complexity and the large number of parameters, which, without sufficient regularization or data augmentation, can lead to memorization of training patterns. This issue is particularly relevant in biodiversity datasets, which often contain imbalanced classes and varying image conditions (e.g., lighting, background clutter). Thus, while DenseNet-201 proved effective in capturing detailed intra-species features, its generalization challenges reduced its viability for deployment in real-world ecological applications without additional tuning.

- **XceptionNet**
  XceptionNet, short for Extreme Inception, was introduced by Chollet (2017) as an evolution of the Inception architecture. Instead of using standard convolutions within inception modules, Xception replaces them with depthwise separable convolutions, a two-step operation that first applies a depthwise convolution (processing each channel independently) followed by a pointwise convolution (a 1x1 convolution across all channels). This significantly reduces the number of parameters while still maintaining or improving performance, resulting in a lightweight yet powerful architecture.

  XceptionNet stood out in our project as one of the top-performing models in both training and testing phases. Its ability to model cross-channel and spatial correlations separately allowed it to learn rich and discriminative feature representations — a crucial requirement for biodiversity classification, where differences between species can be subtle and highly localized (e.g., spot patterns, scale structures, eye shapes). The architecture's depth and efficient parameter usage helped it generalize well across different classes, handling both common and rare species with high accuracy.

  Additionally, XceptionNet's structure proved to be robust against noise and background clutter, which are common in field-collected biodiversity images. The model also trained faster compared to some deeper alternatives like DenseNet-201, due to its computational efficiency. Based on its consistent and reliable performance, XceptionNet was selected as a core component of our final two-stage species identification framework, where it handled the broad classification phase before passing the results to a more detailed model for species-level refinement.

    - **Highest test accuracy** of 94.09%
    - **Strong generalization ability**
    - Lower overfitting tendency due to dropout layers integrated during our implementation

3.5.1 Why XceptionNet Was Chosen for the Main CNN

XceptionNet emerged as the best choice for the **Main CNN** due to its:

- Superior **test accuracy**, crucial for real-world performance
- Balanced training performance, indicating good **model generalization**
- Scalability and compatibility with transfer learning
- Proven performance in **fine-grained classification tasks** from other research papers

Given its high performance and robustness, **XceptionNet** was selected to classify the **primary species group** (e.g., bird, butterfly, frog) before passing the image to category-specific sub-models for further classification.

3.6Dataset

The dataset forms the backbone of our species identification system and plays a critical role in training and evaluating the performance of both the **Main CNN** and the **Sub CNNs**. As our client, the **Zoological Survey of India (ZSI)**, is involved in fieldwork across biodiversity-rich regions, the dataset provided reflects authentic, real-world imagery of Indian wildlife species.

1. Main CNN Dataset

The dataset used for training the **Main CNN (XceptionNet)** was entirely provided by the Zoological Survey of India. It consists of **7 major species classes**:

- Birds
- Butterflies
- Frogs
- Snakes
- Moths
- Odonatas (dragonflies and damselflies)
- Lizards

The distribution of this dataset is as follows:

- **Total Images:**1,260
- **Training Set:** 82 images
- **Testing Set:** 14 images

Although the dataset is relatively small, it is highly curated and specific to the Indian subcontinent, making it more relevant and domain-specific. To mitigate the limitations posed by the small sample size, we implemented **data augmentation techniques** such as flipping, rotation, zooming, and brightness adjustments to artificially increase the training data size and enhance model generalization.

2. Sub CNN Datasets

Each **Sub CNN**, which performs species-level classification within the main class (e.g., different types of birds), was trained on its own dedicated dataset.

Due to the limited data provided by ZSI for fine-grained species identification, we took a **hybrid approach**:

- Used **all available labeled data** from ZSI.
- Supplemented this with **visually similar images** from **open internet sources** (licensed for research and academic use), ensuring ethical data collection.

The dataset distribution for each Sub CNN is:

- **Total Images per Class Group:** 12
- **Training Set:** 10 images

- **Testing Set:** 2 images

This dataset structure was replicated across all Sub CNNs, including:

- Bird CNN
- Snake CNN
- Frog CNN
- Butterfly CNN
- Moth CNN
- Odonata CNN
- Lizard CNN

To ensure meaningful learning from this limited data, we employed **transfer learning** using **EfficientNetV2 Large** models and **data augmentation**, allowing the CNNs to generalize well even with smaller datasets.

3.6.1 Challenges and Mitigation

Challenge	Solution
Small dataset size	Applied data augmentation, transfer learning, and dropout regularization
Class imbalance	Used stratified sampling and balanced mini-batches during training
Noisy internet images	Performed manual filtering with expert assistance from ZSI

TABLE 3.3

3.7 Screenshot

3.7.1 Login

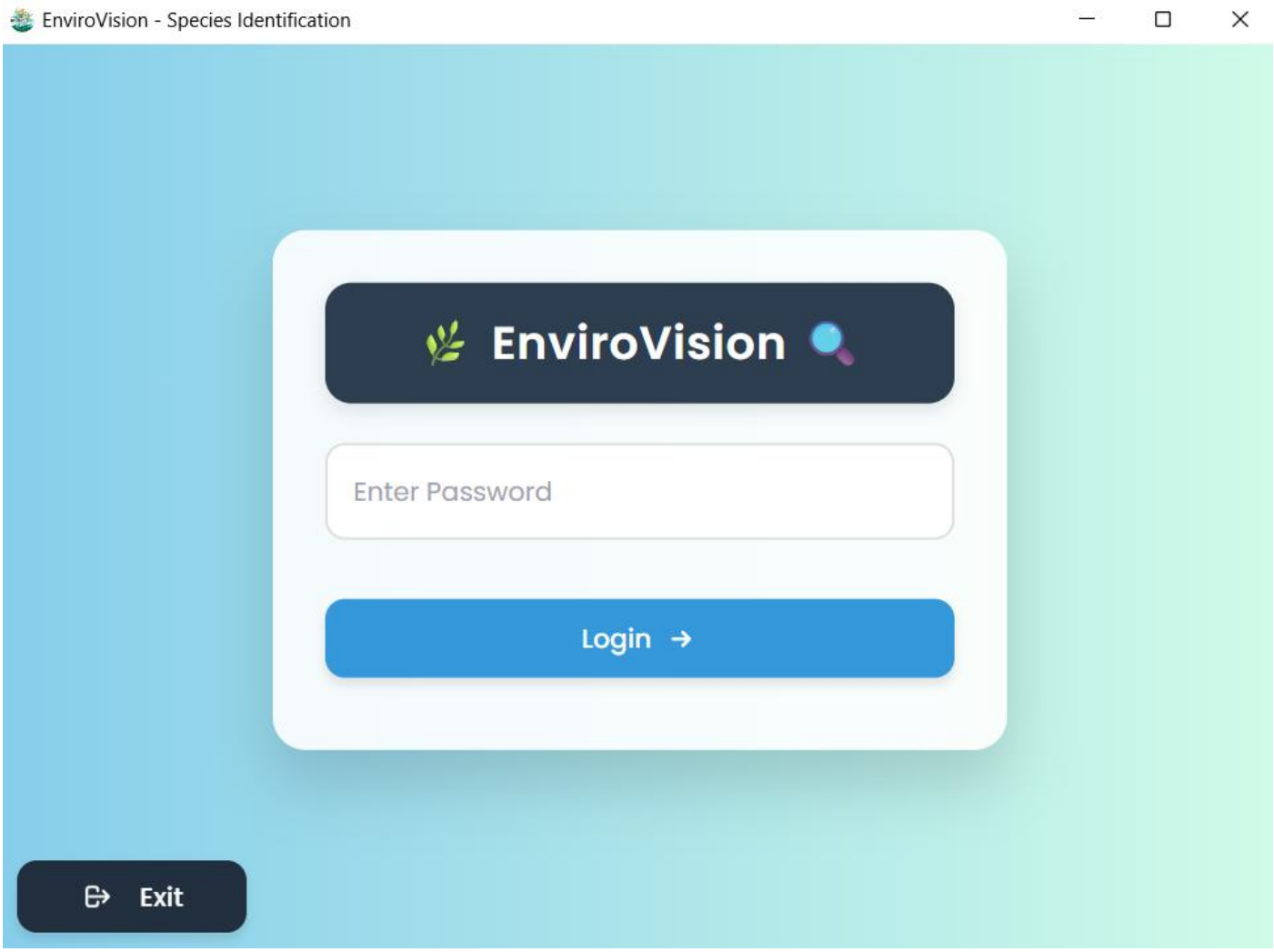

Fig3.1 Login Interface of the EnviroVision System

3.7.1.1.Main Menu Interface

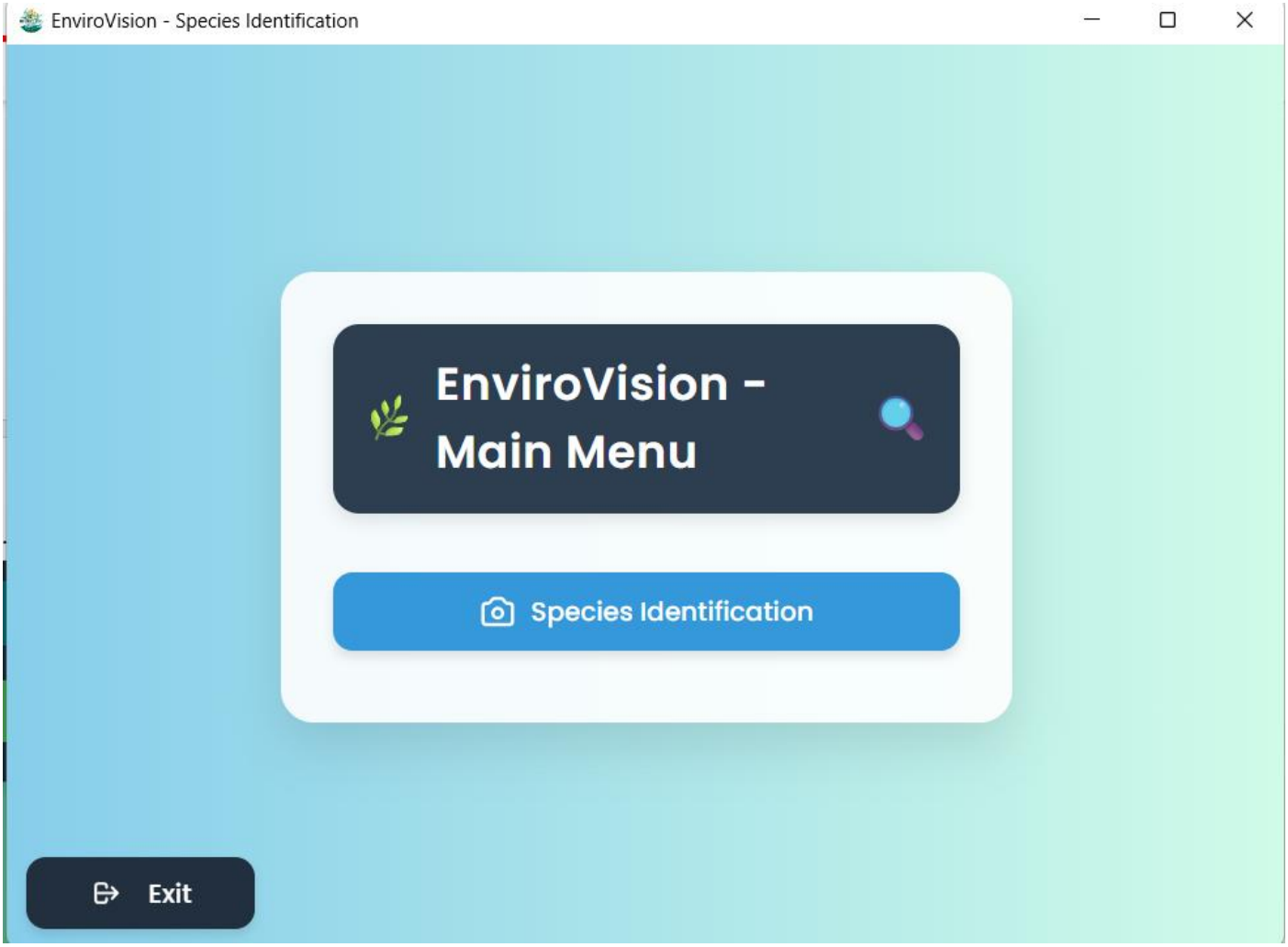

Fig 3.2 EnviroVision - Main Menu Interface

3.7.1.2 Species Identifier

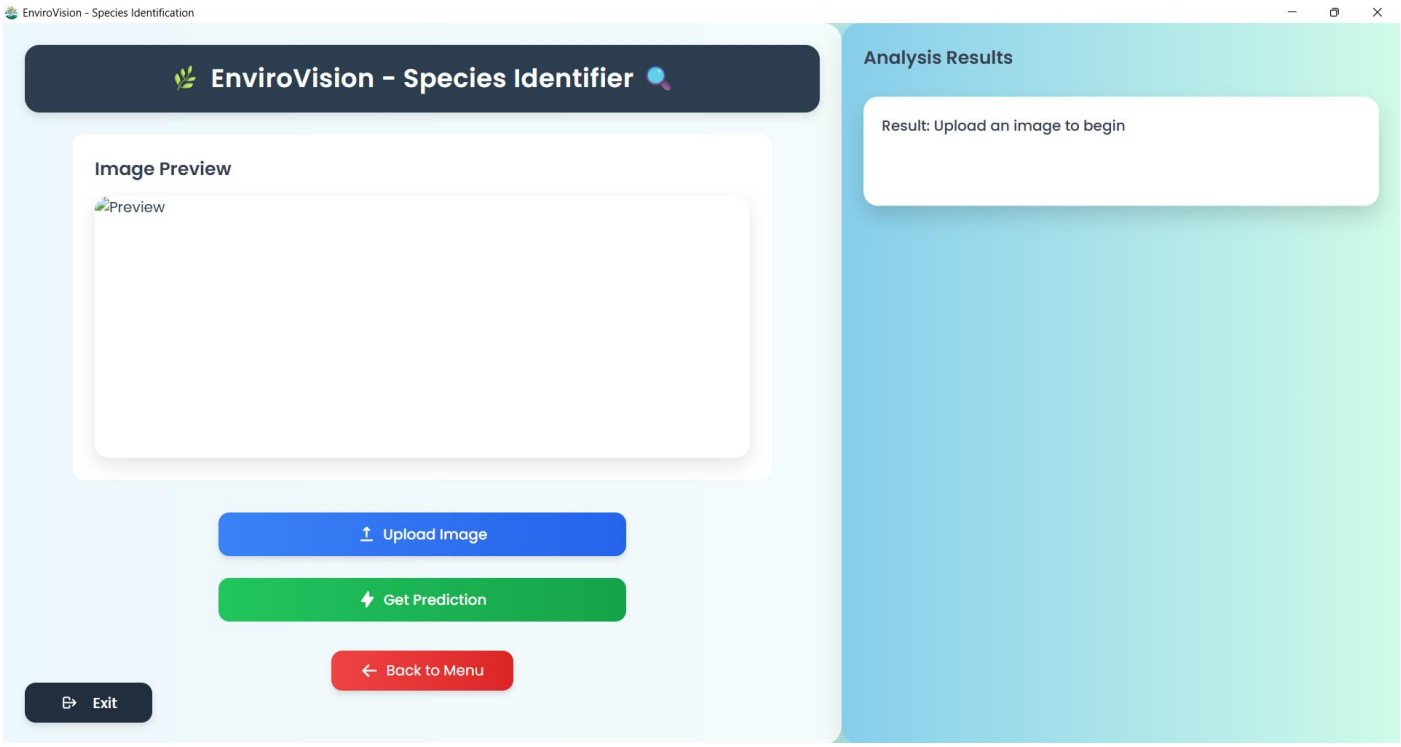

Fig 3.3 Image Upload Interface in Species Identifier

3.7.1.3 Upload Dialog Window

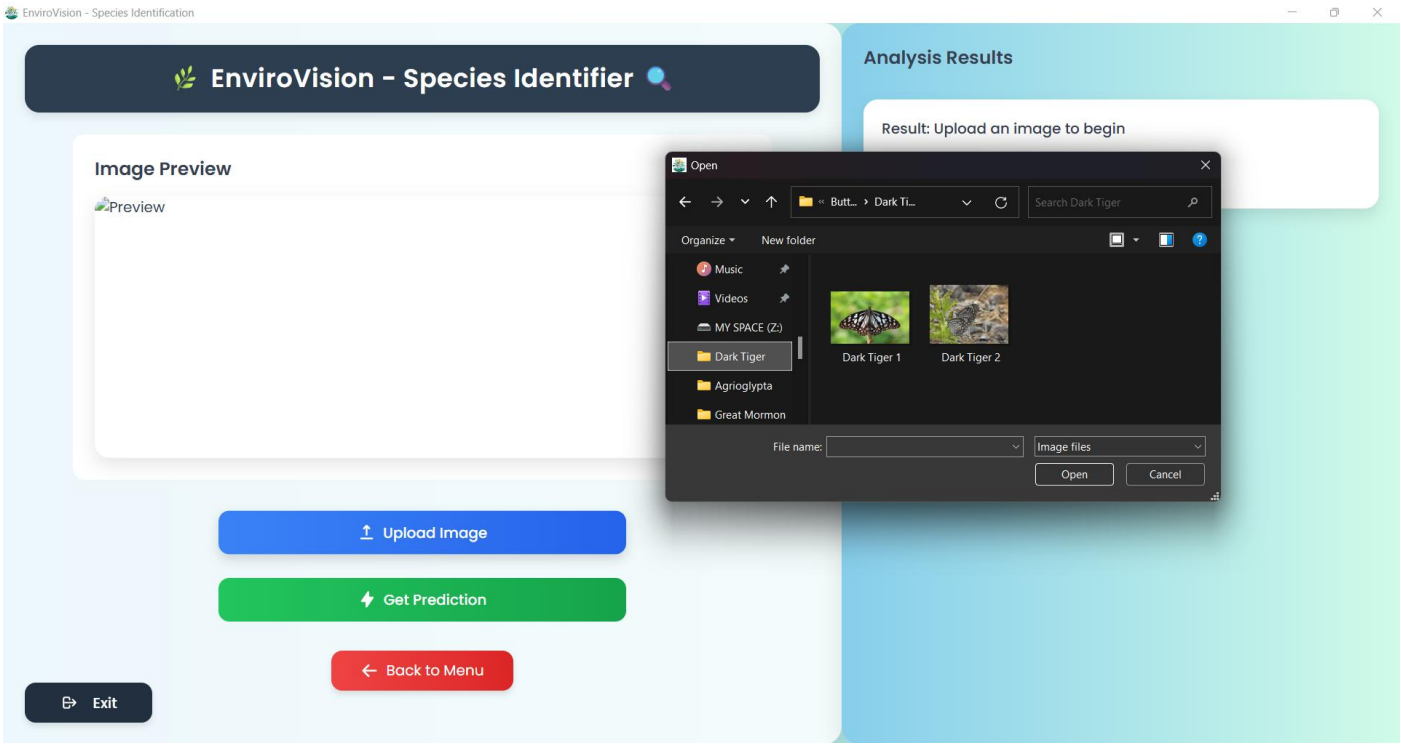

Fig 3.4 Image Upload Dialog Window

3.7.1.4 Cropping Feature

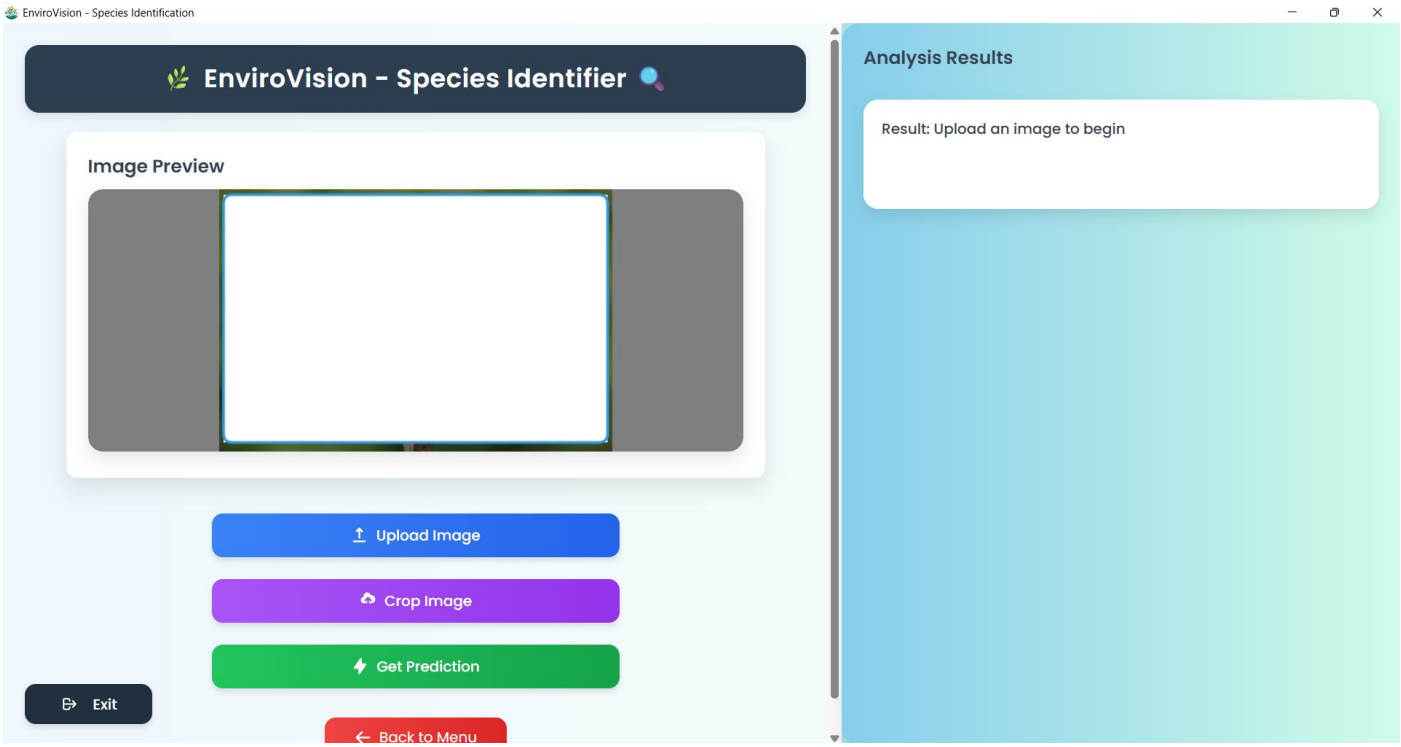

Fig 3.5 Cropping Feature Interface

3.7.1.5 Species Prediction Output

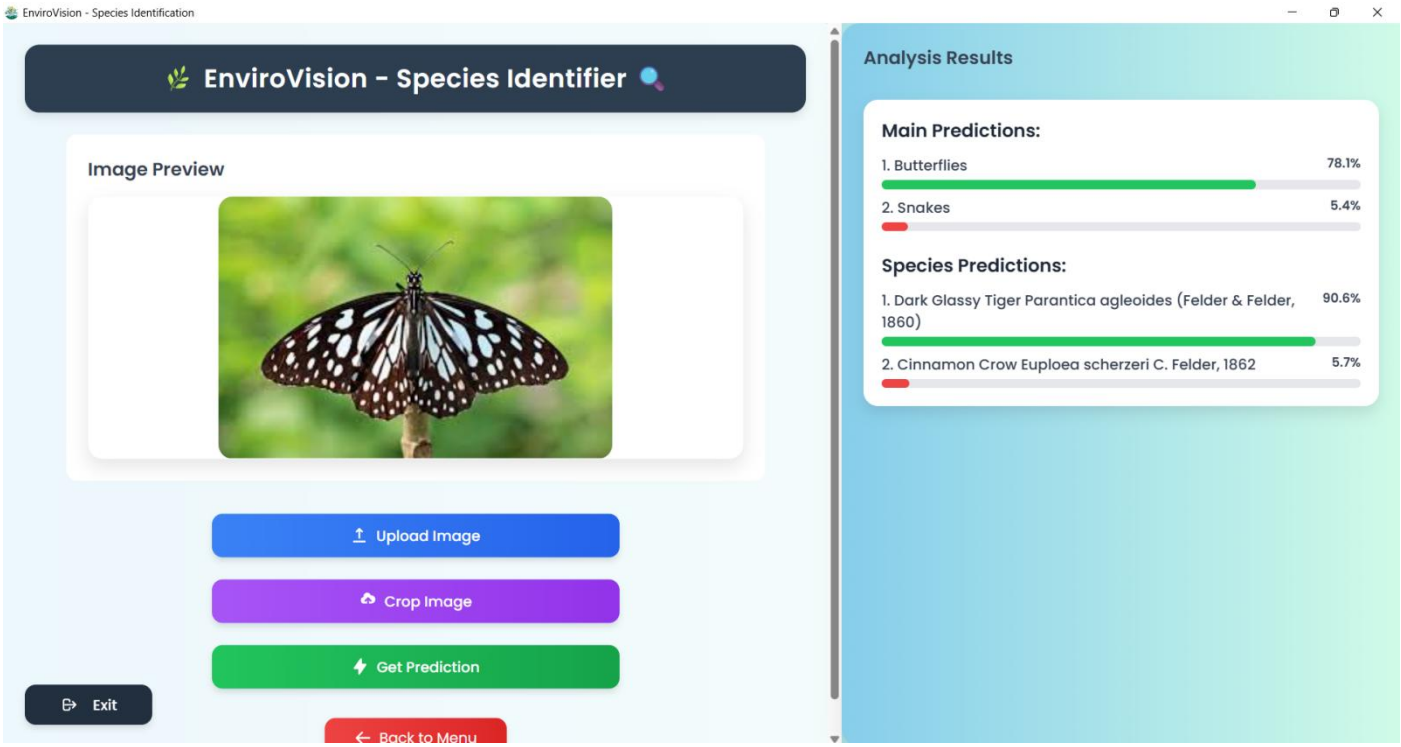

Fig 3.6 Species Prediction Output Interface

CHAPTER - 4

MANUAL TESTING REPORT OF ENVIROVISION

4.1 Login

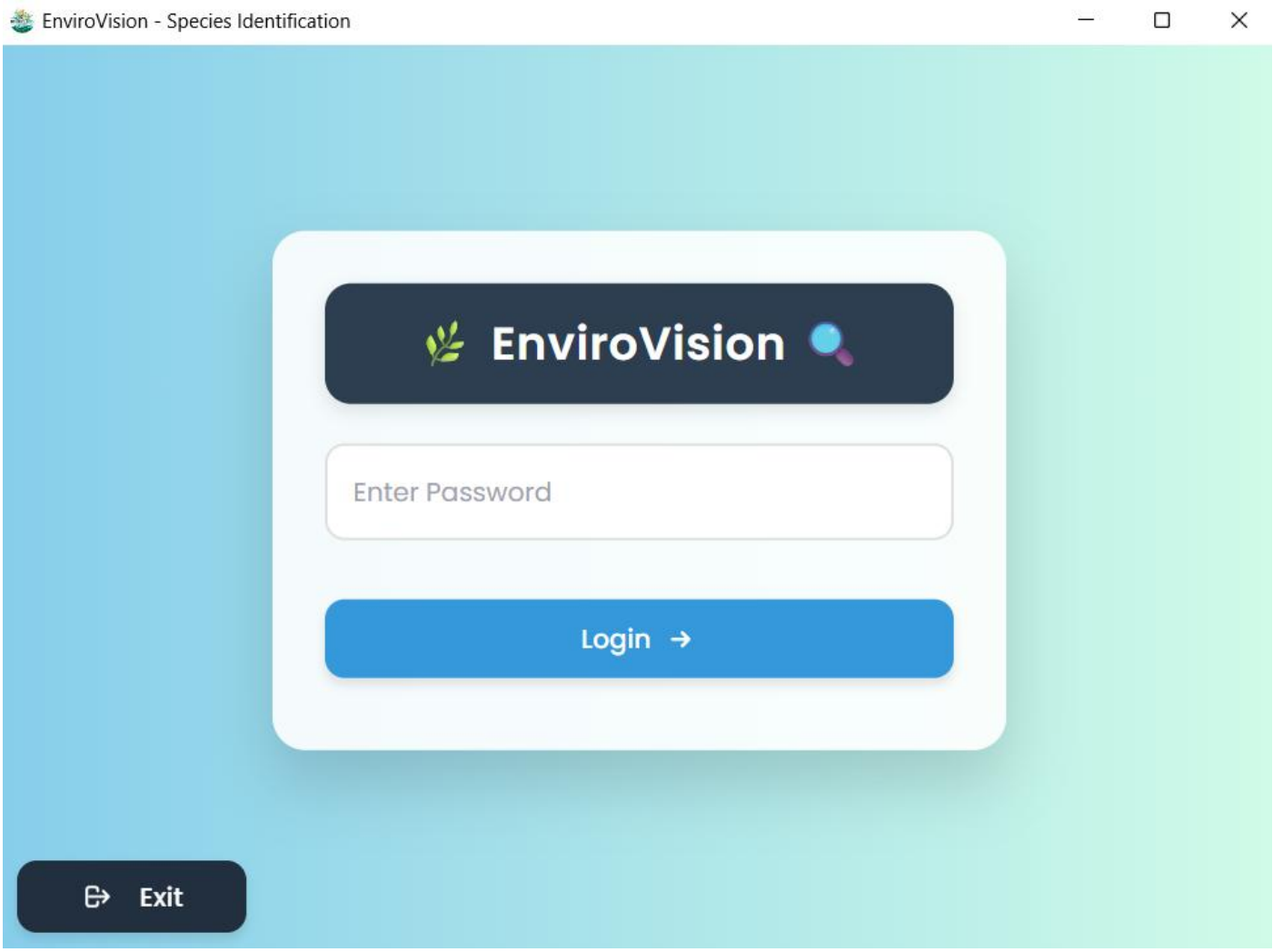

Fig 3.7 Login Interface of the EnviroVision System

4.1.1 Main Menu Interface

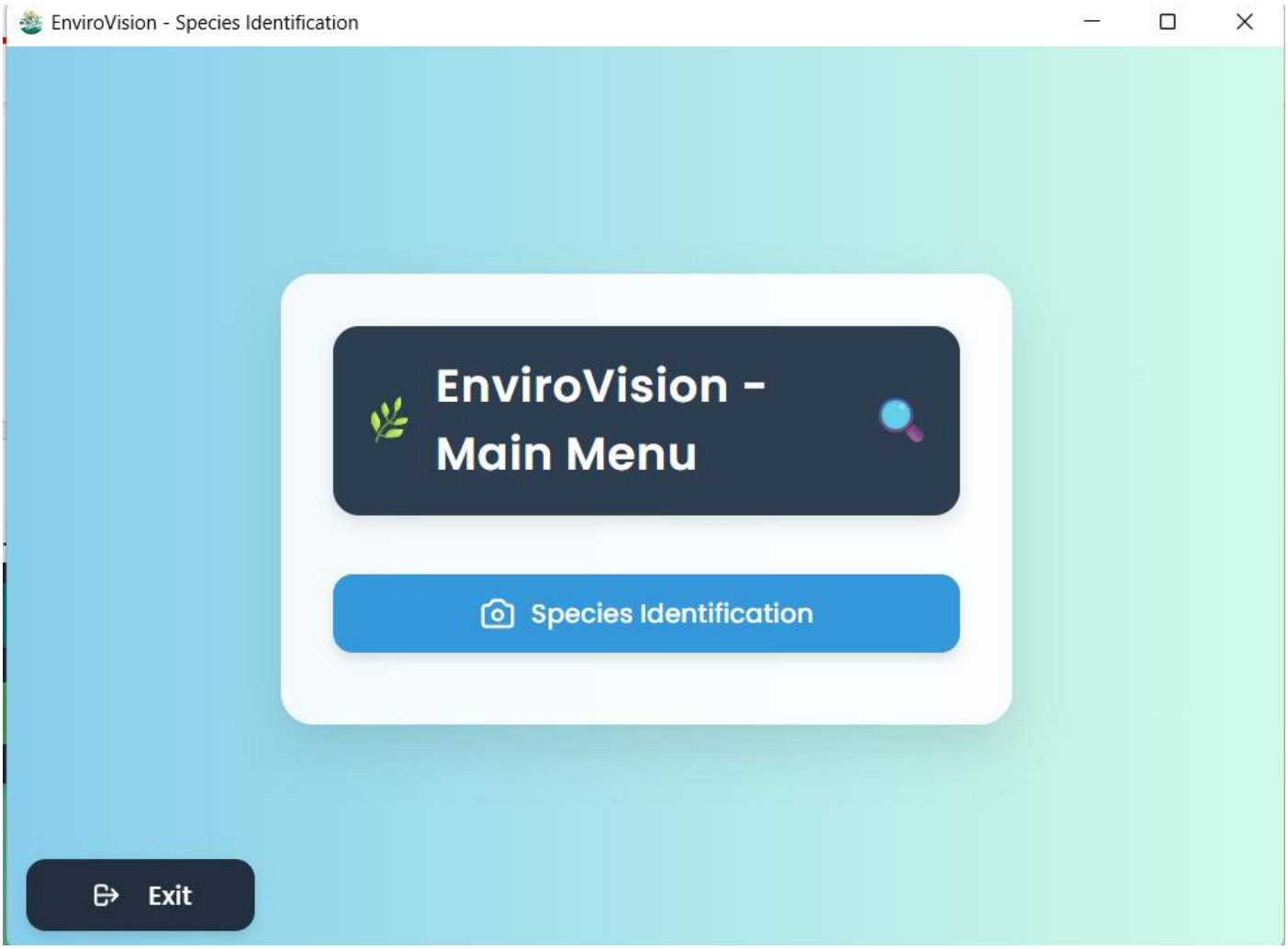

Fig 3.8Main Menu Interface

4.1.1.1 Species Identifier

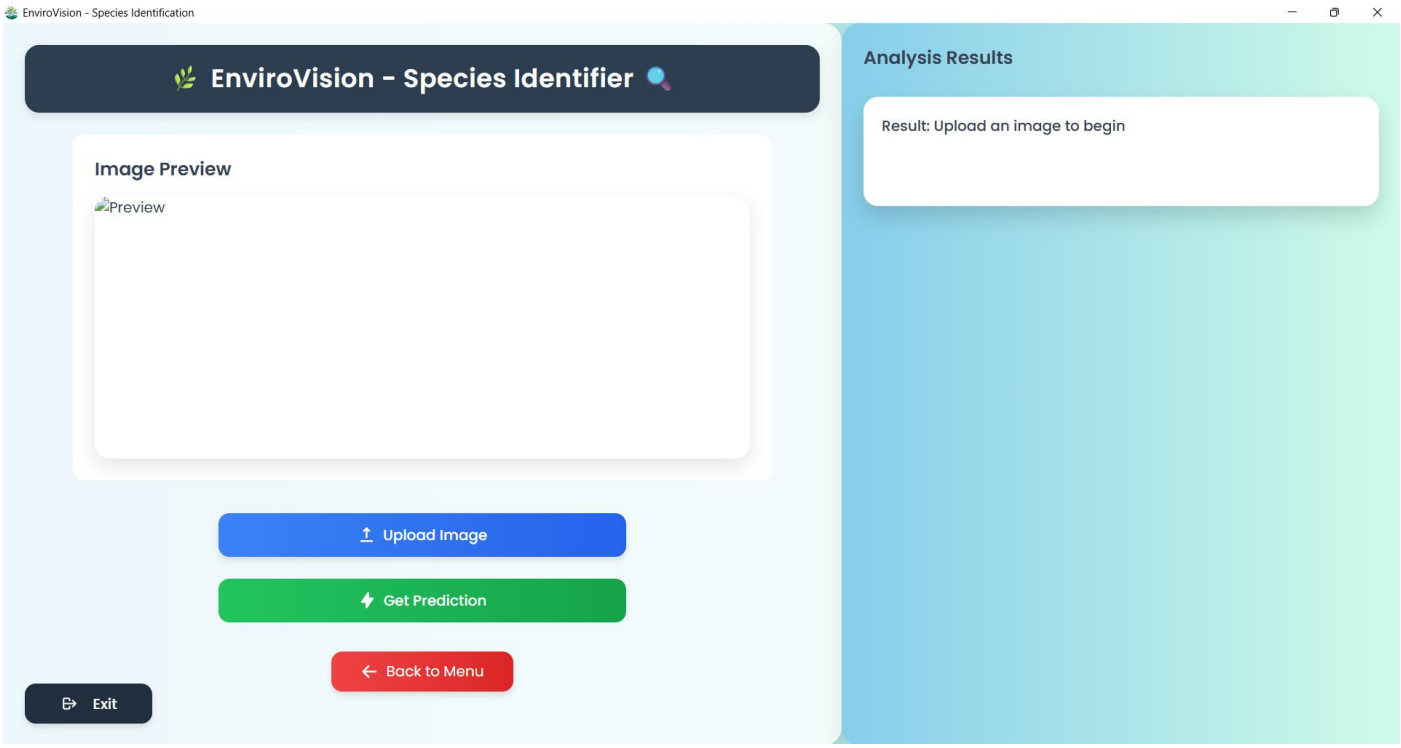

Fig3.9 Image Upload Interface in Species Identifier

4.1.1.2Upload Dialog Window

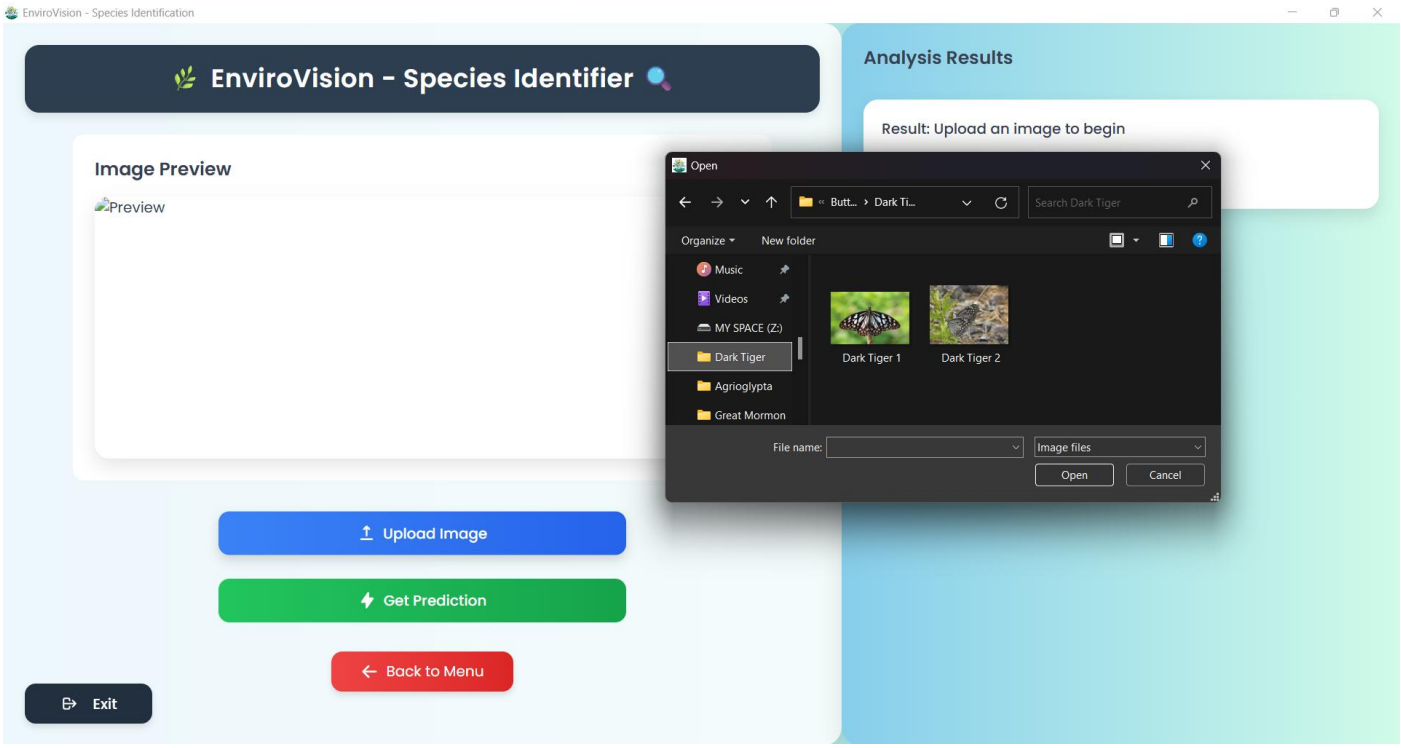

Fig3.10Image Upload Dialog Window

4.1.1.3Cropping Feature

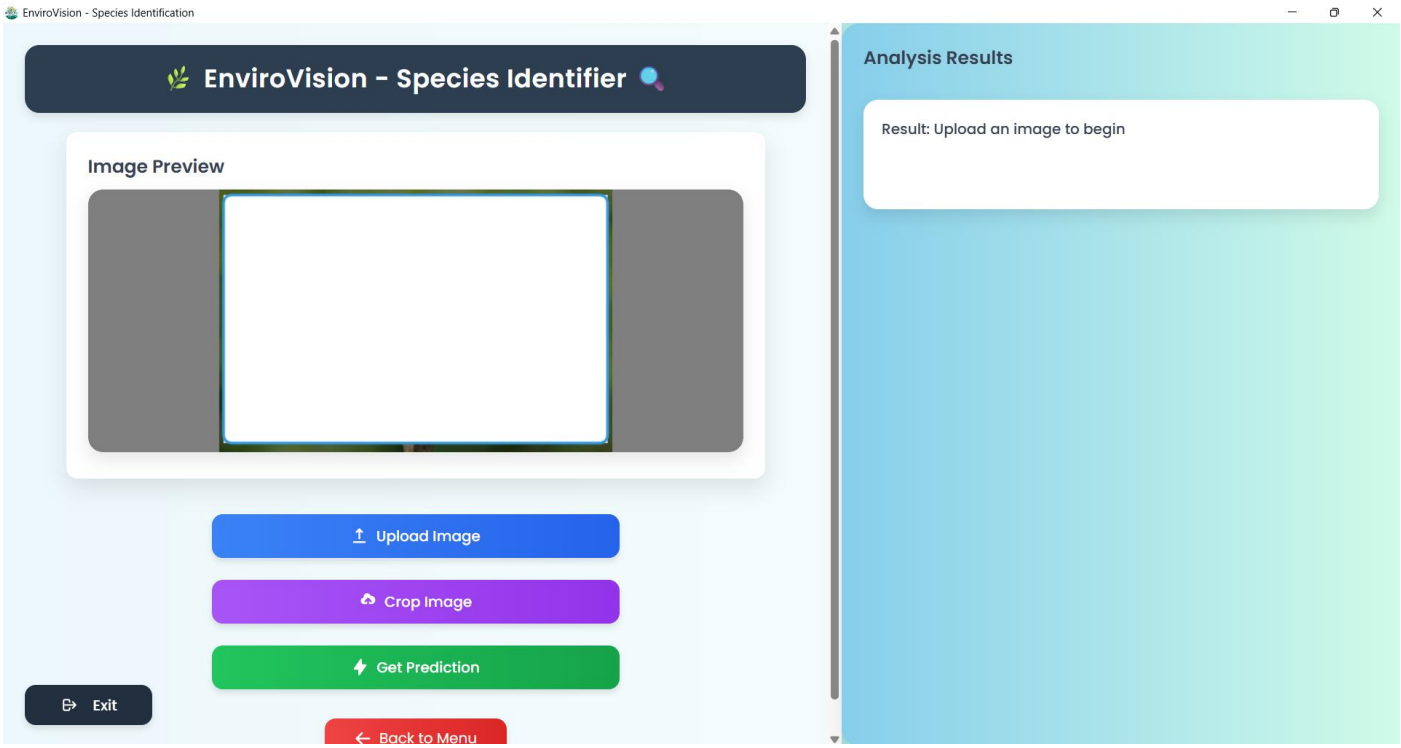

Fig3.11Cropping Feature Interface

4.1.1.4 Species Prediction Output

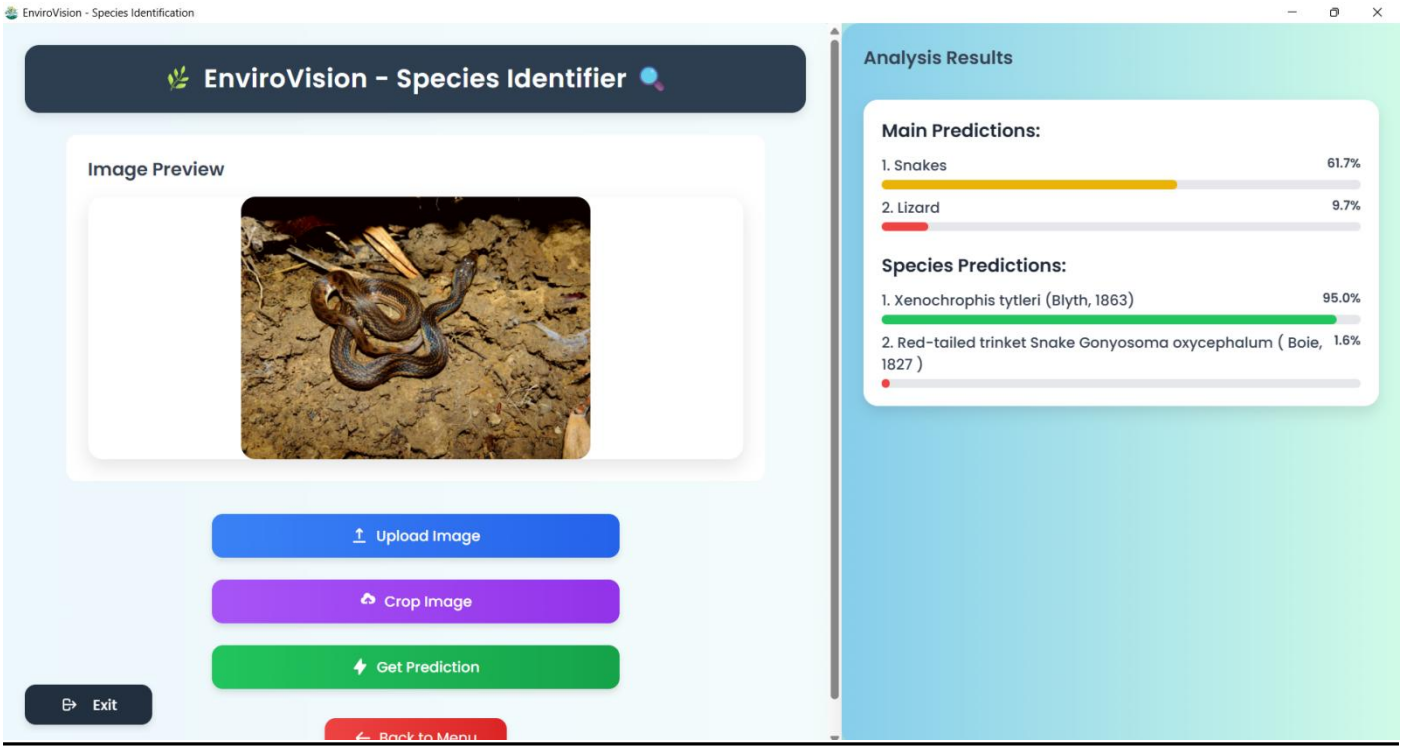

Fig3.12 (i) Snake Prediction Output

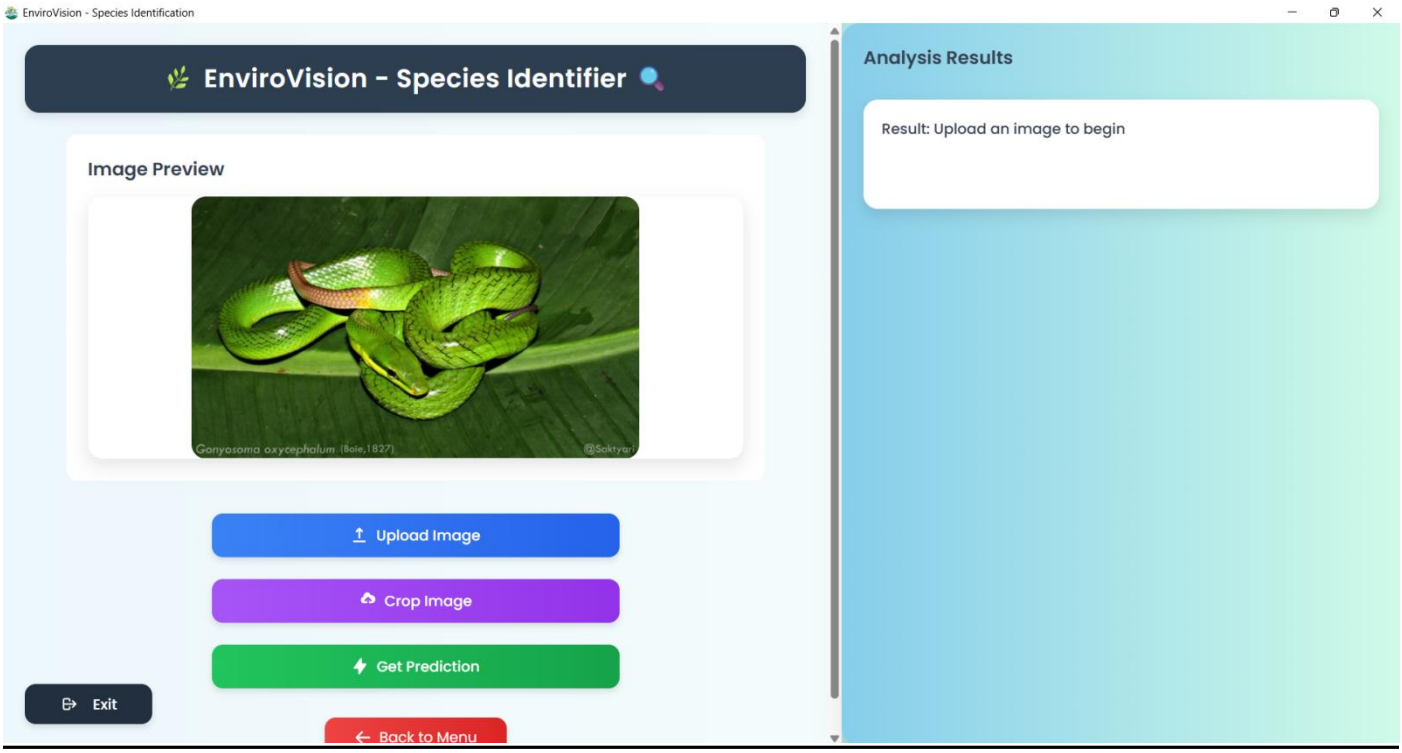

Fig 3.12 (ii) Snake Prediction Output

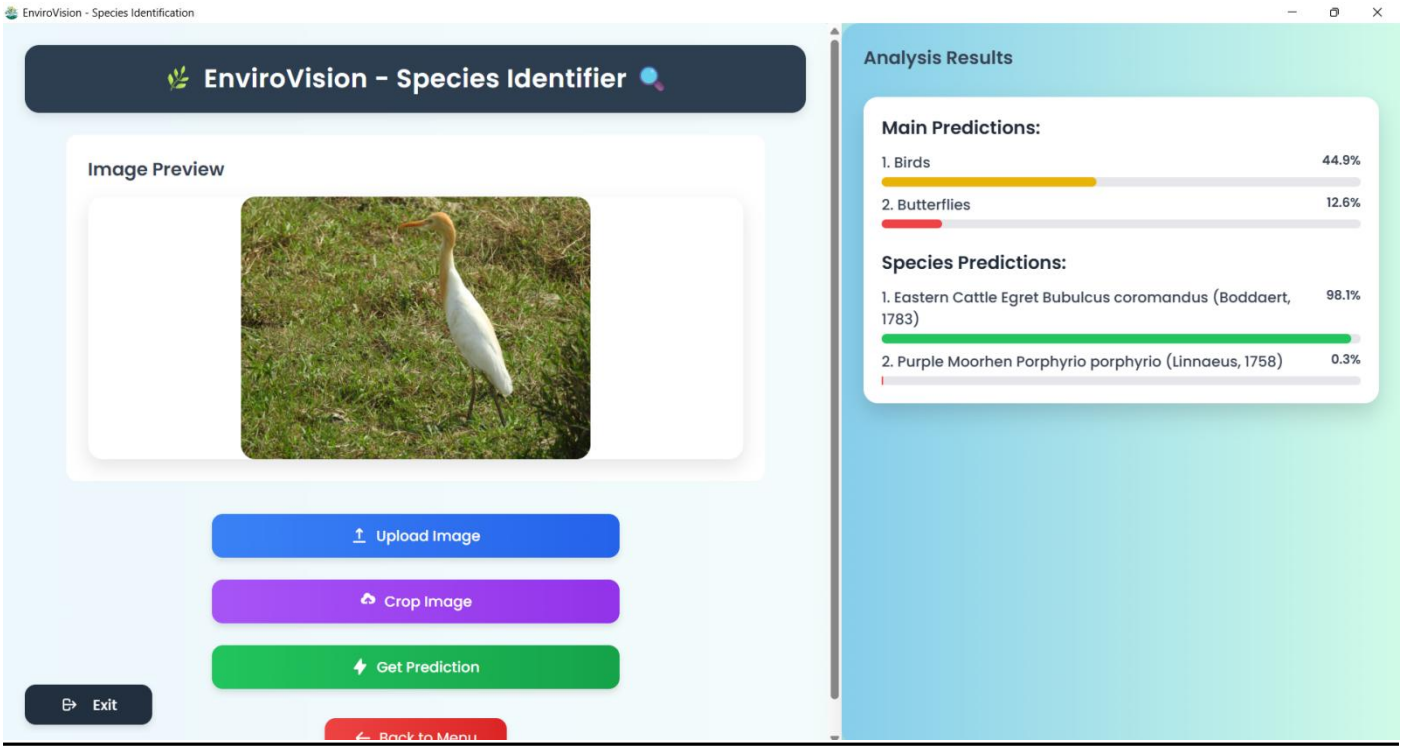

Fig 3.13 (i) Bird Prediction Output

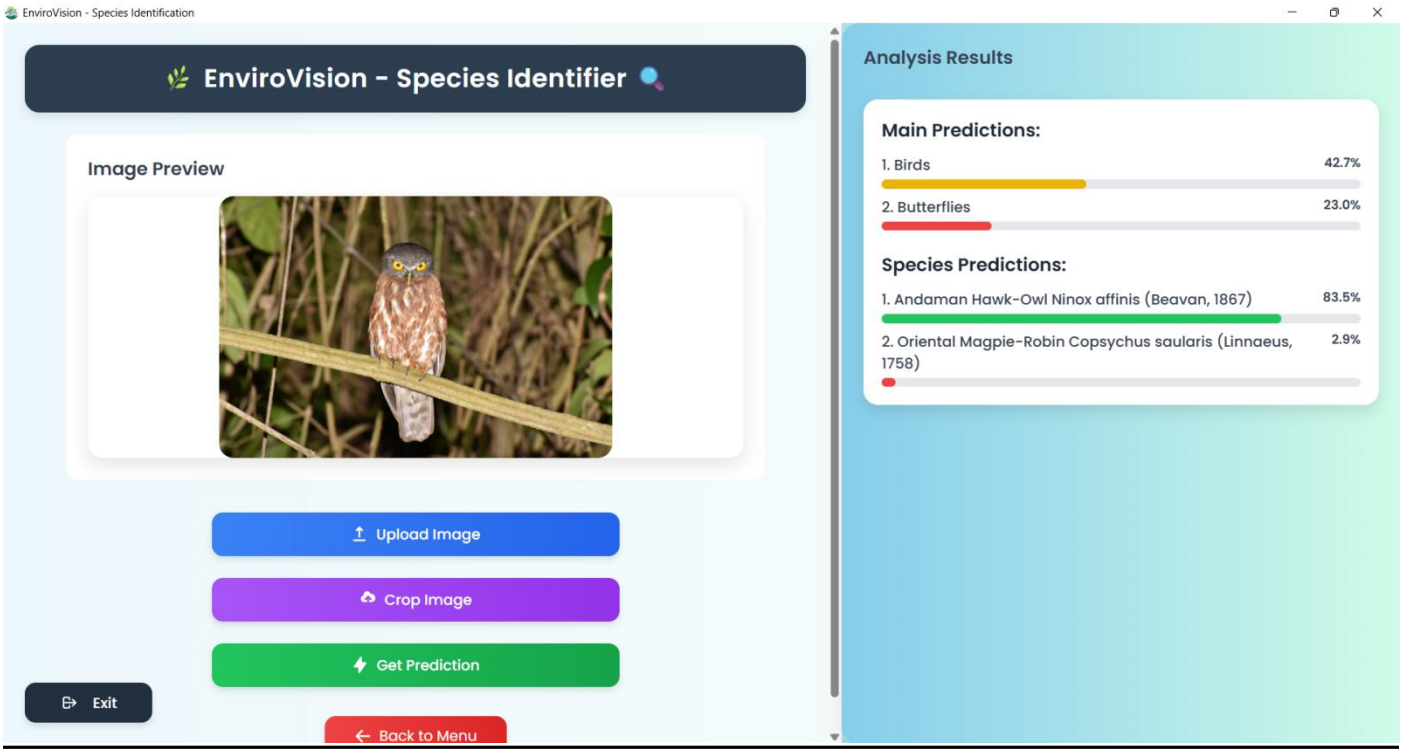

Fig 3.13 (ii) Bird Prediction Output

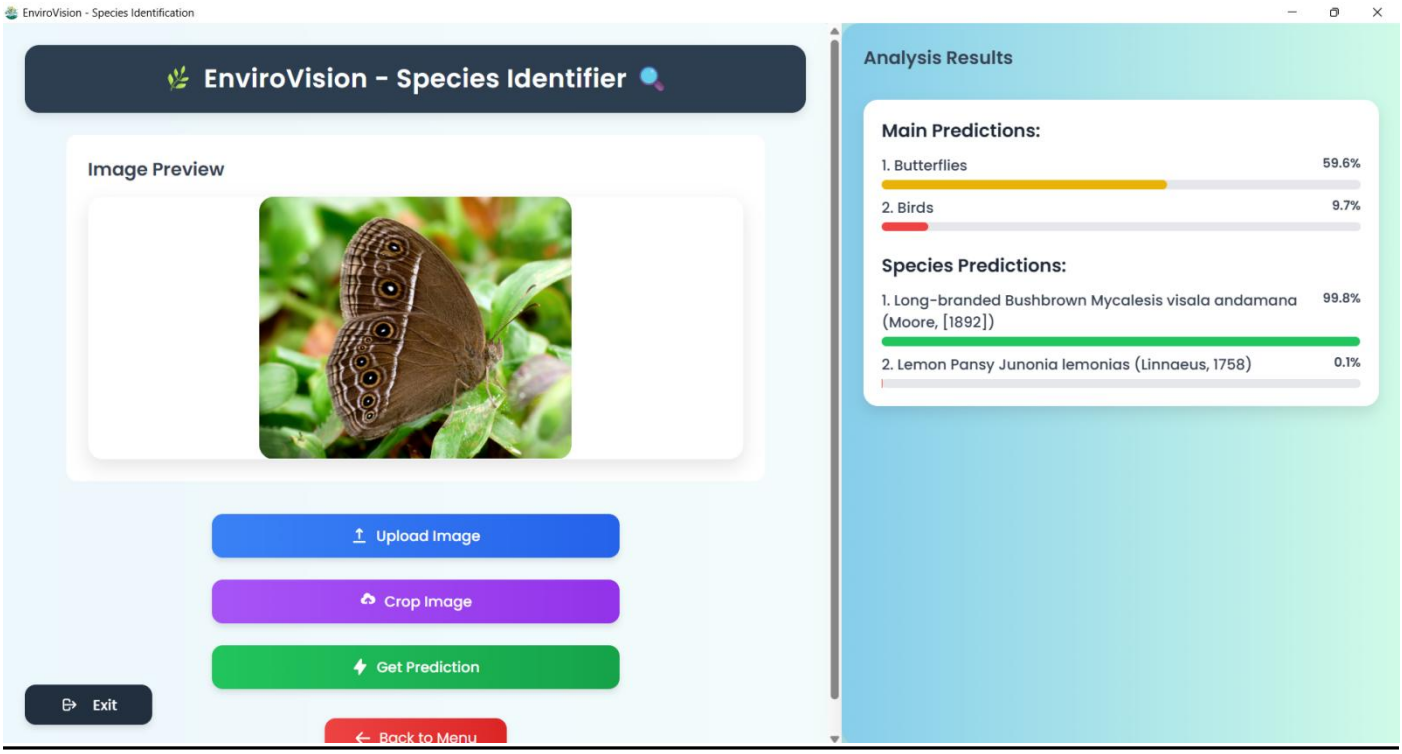

Fig 3.14 (i) Butterfly Prediction Output

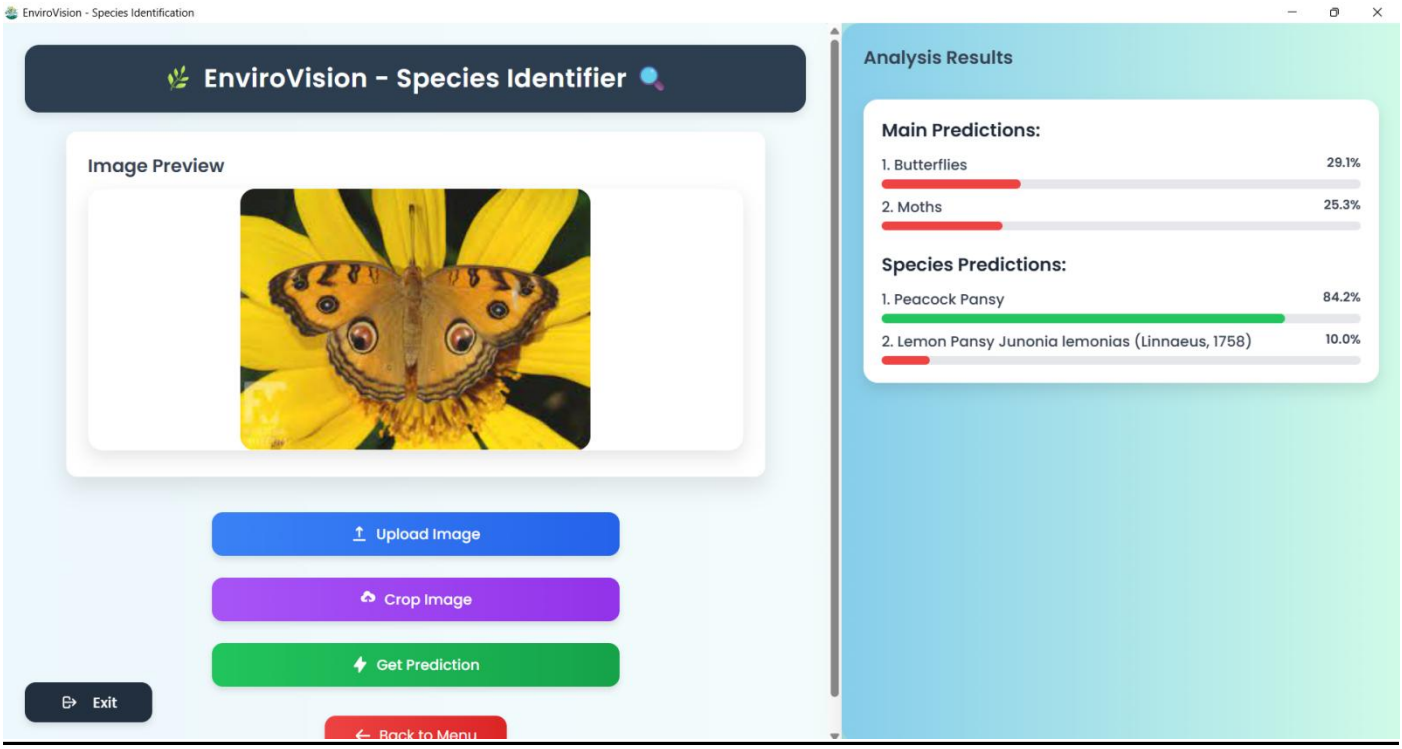

Fig 3.14 (ii) Butterfly Prediction Output

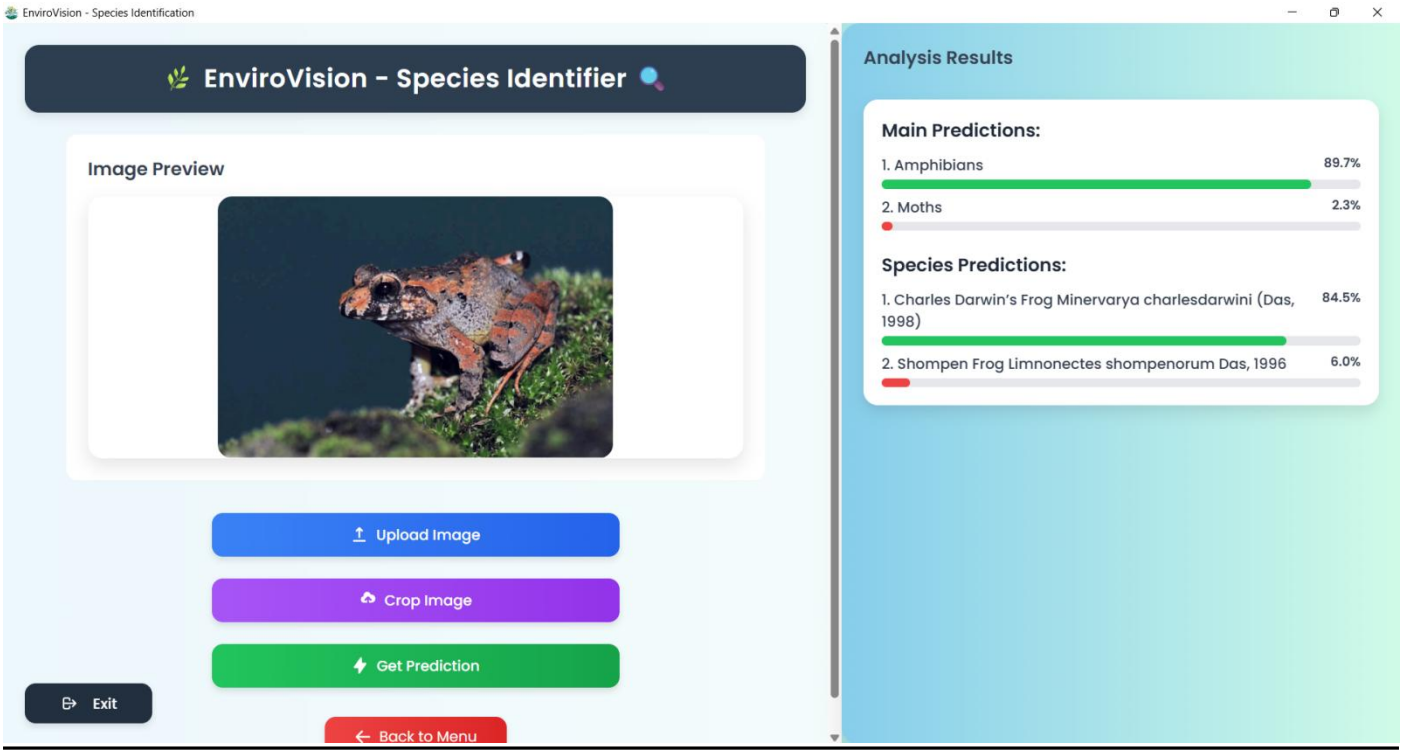

Fig 3.15 (i) Frog Prediction Output

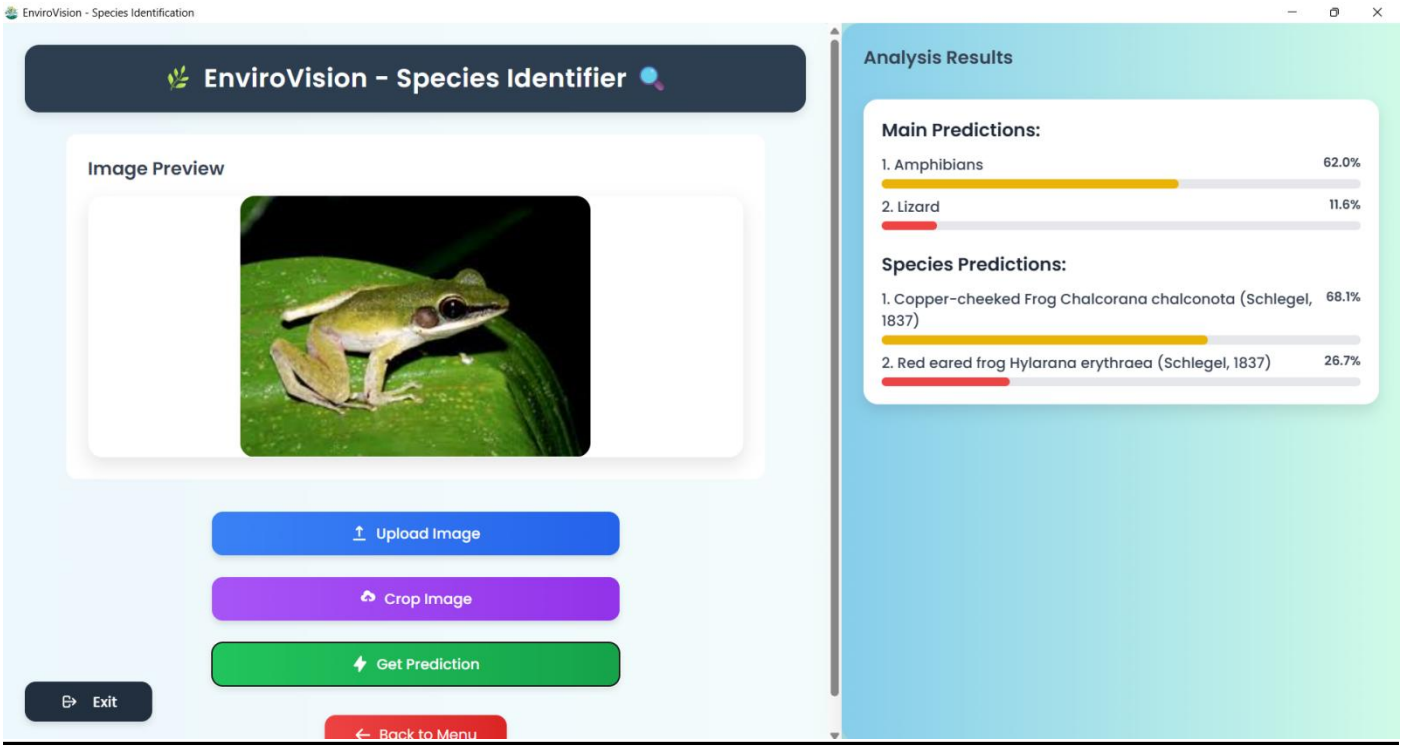

Fig 3.15 (ii) Frog Prediction Output

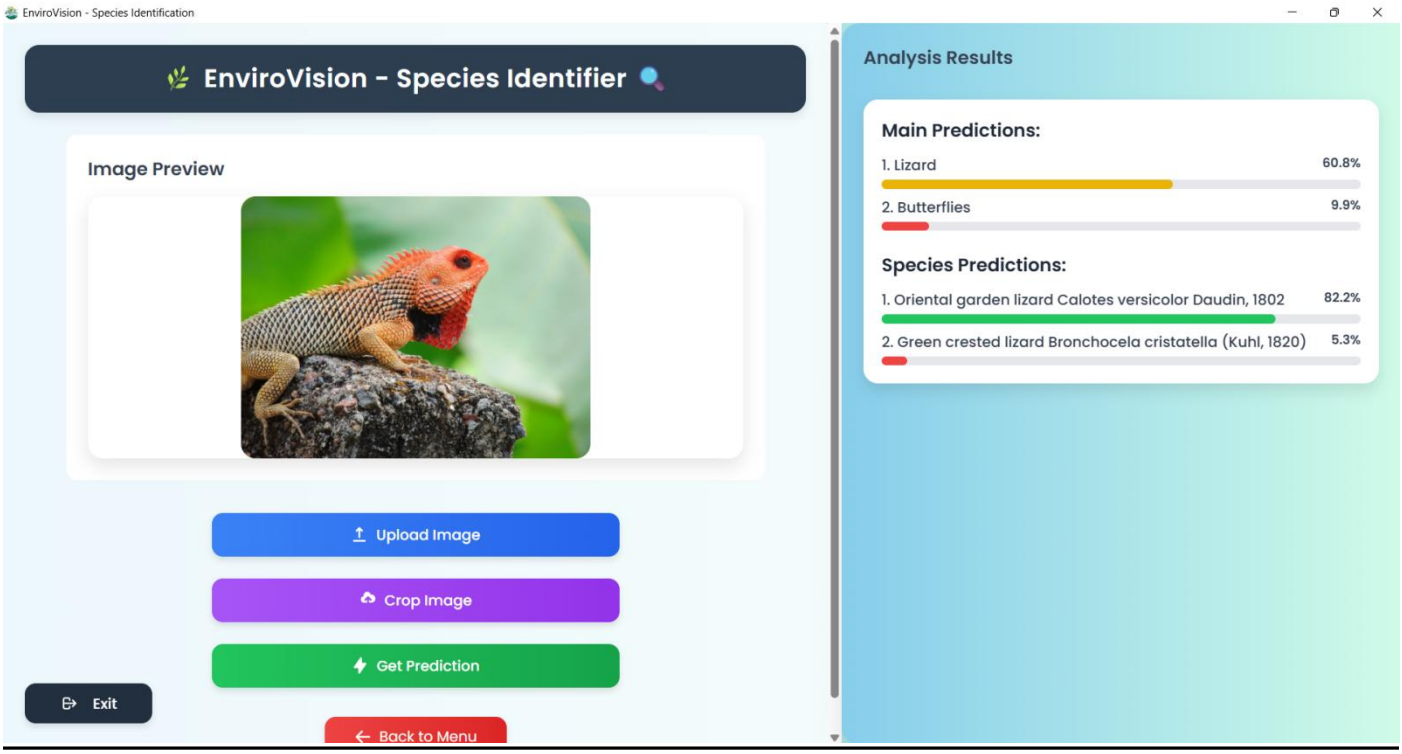

Fig 3.16 (i) Lizard Prediction Output

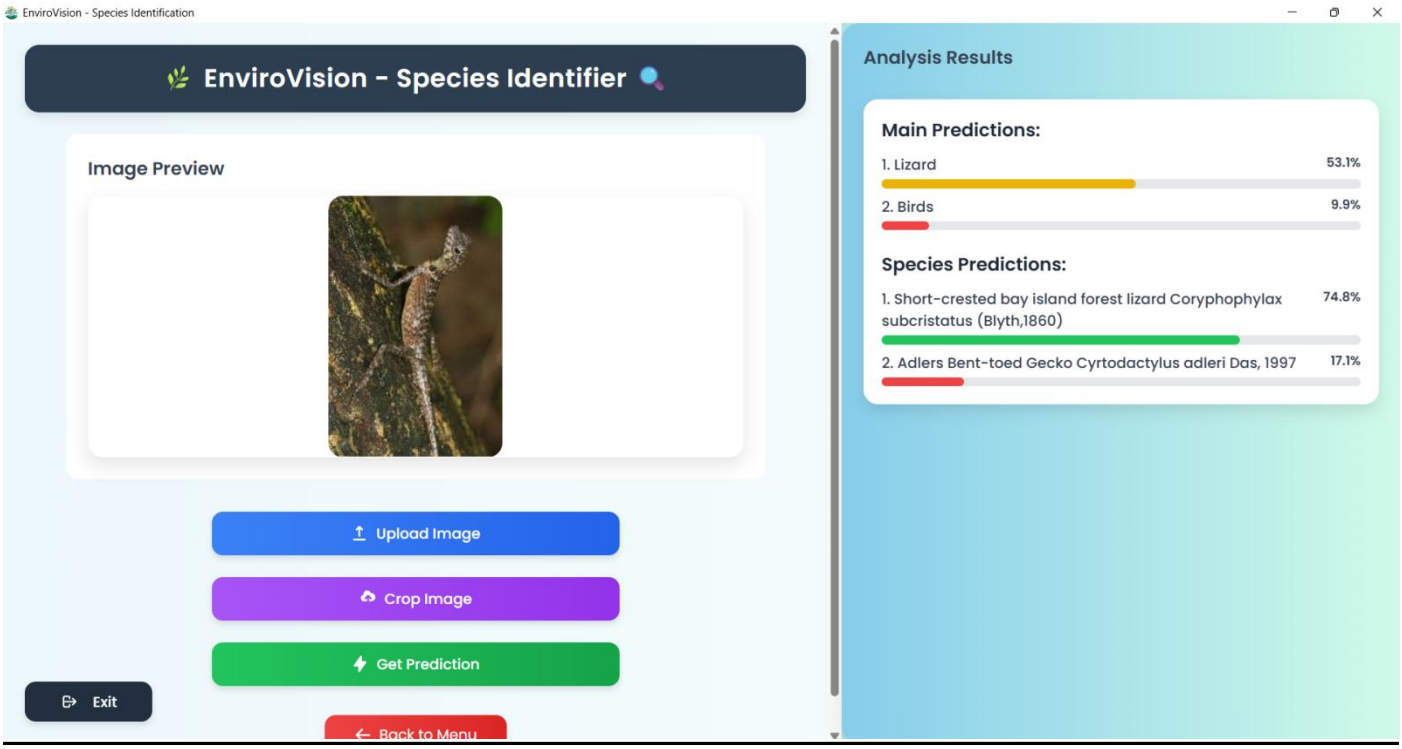

Fig 3.16 (ii) Lizard Prediction Output

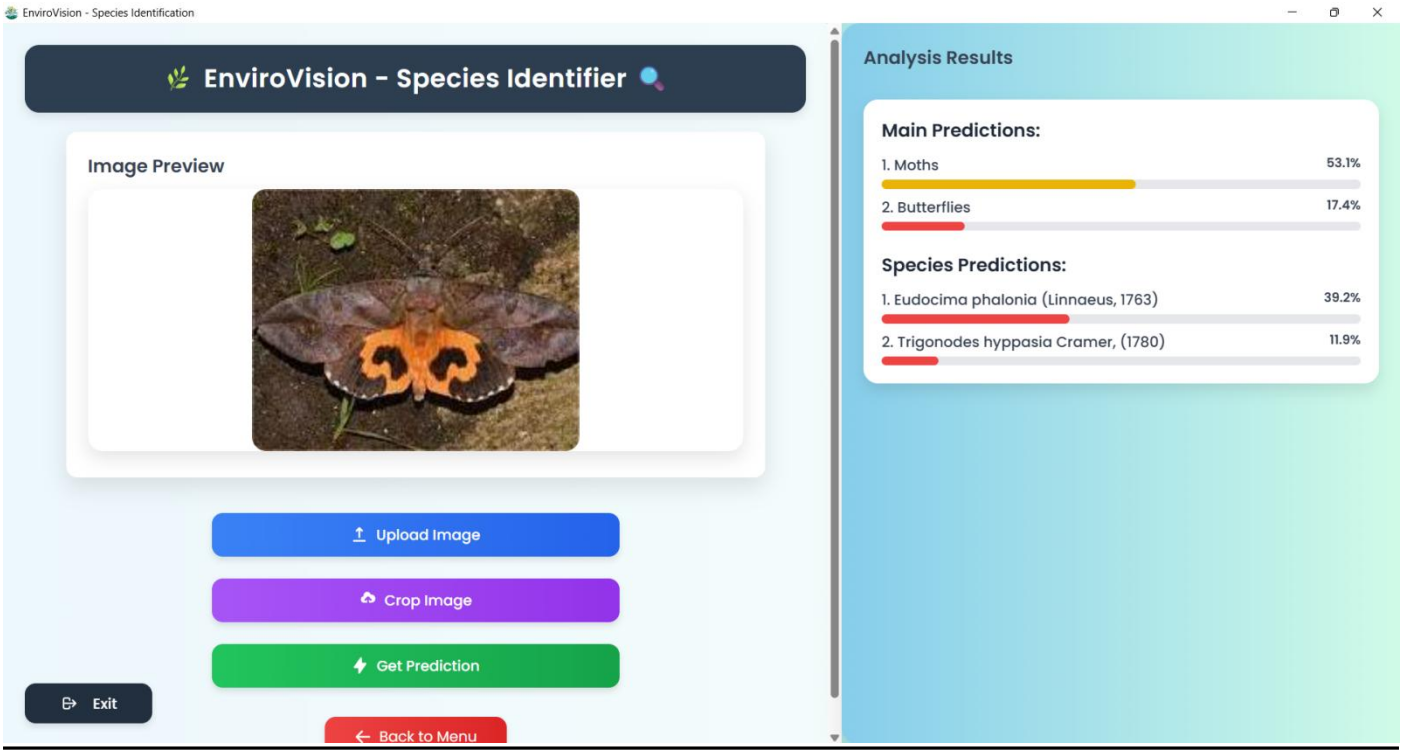

Fig 3.17 (i) Moth Prediction Output

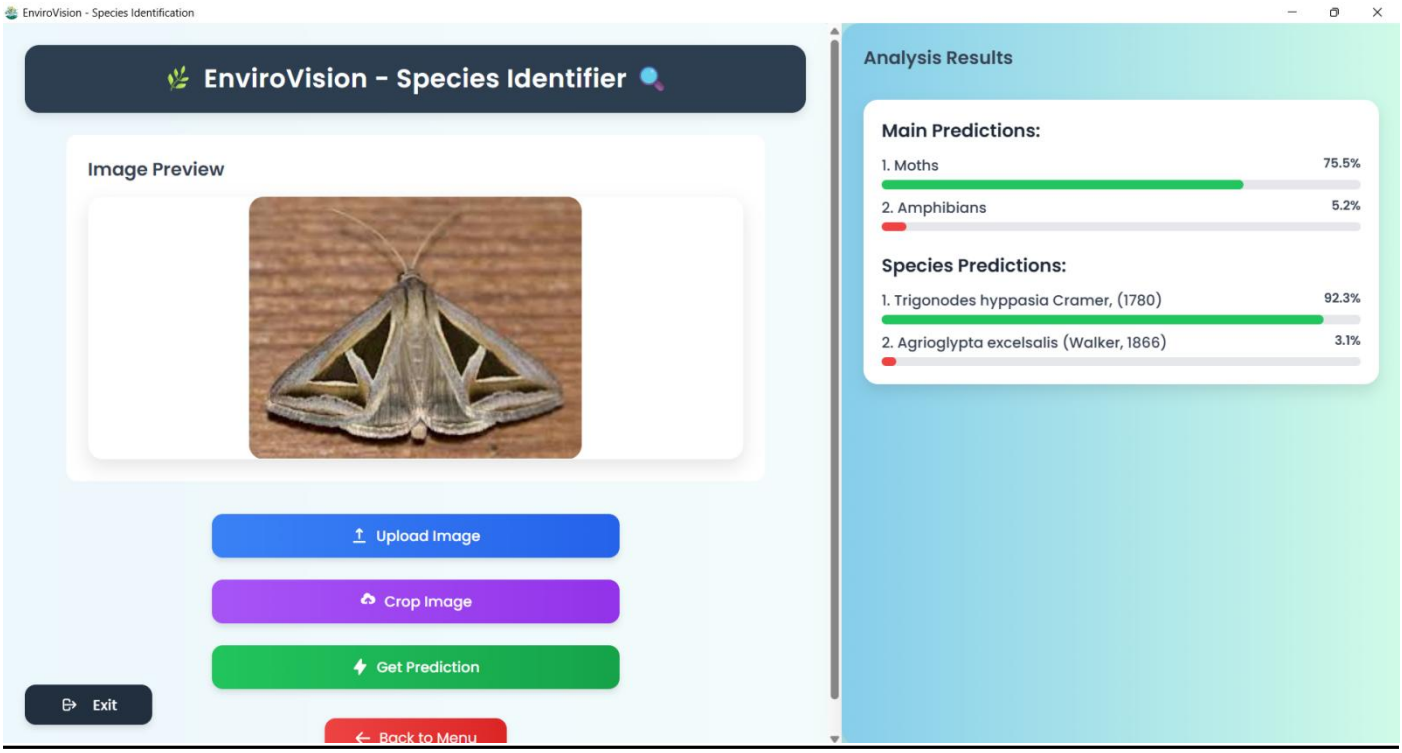

Fig 3.17 (ii) Moth Prediction Output

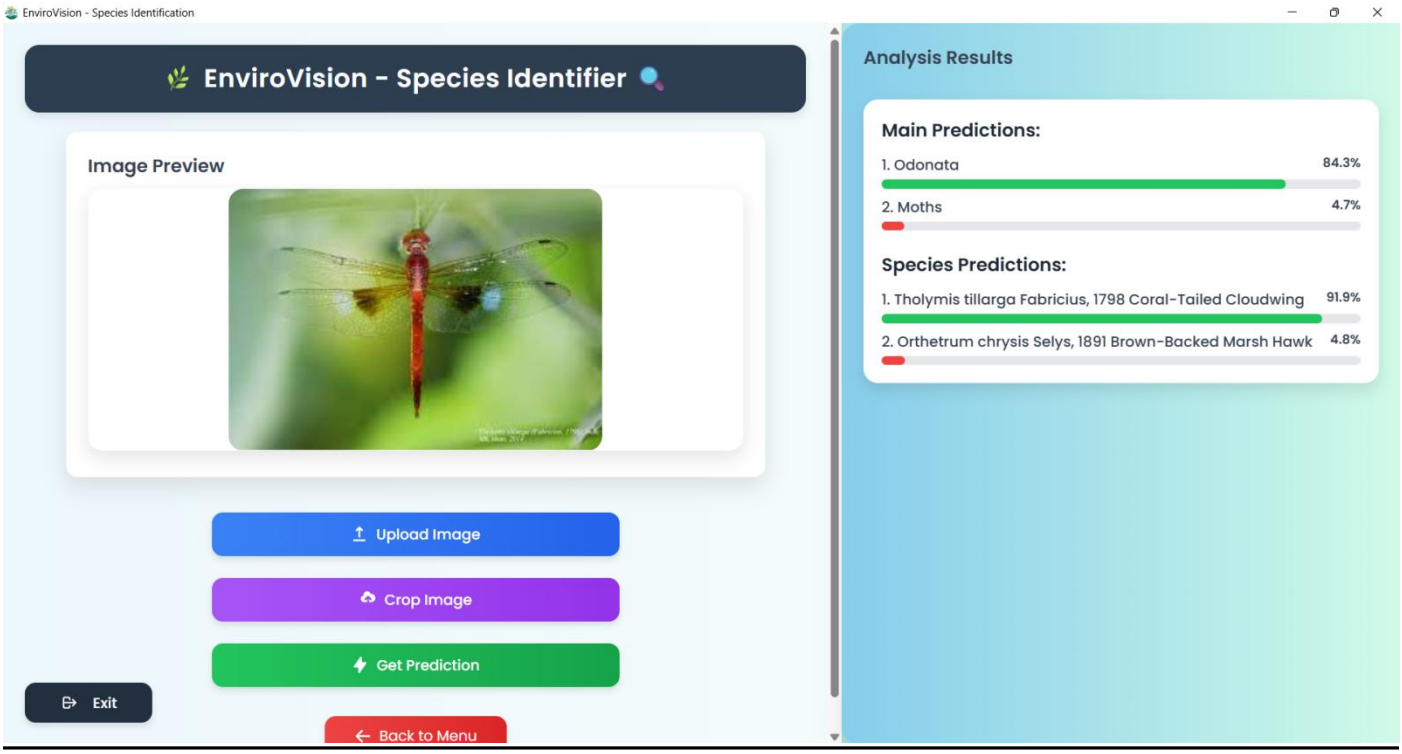

Fig 3.18 (i) Odonata Prediction Output

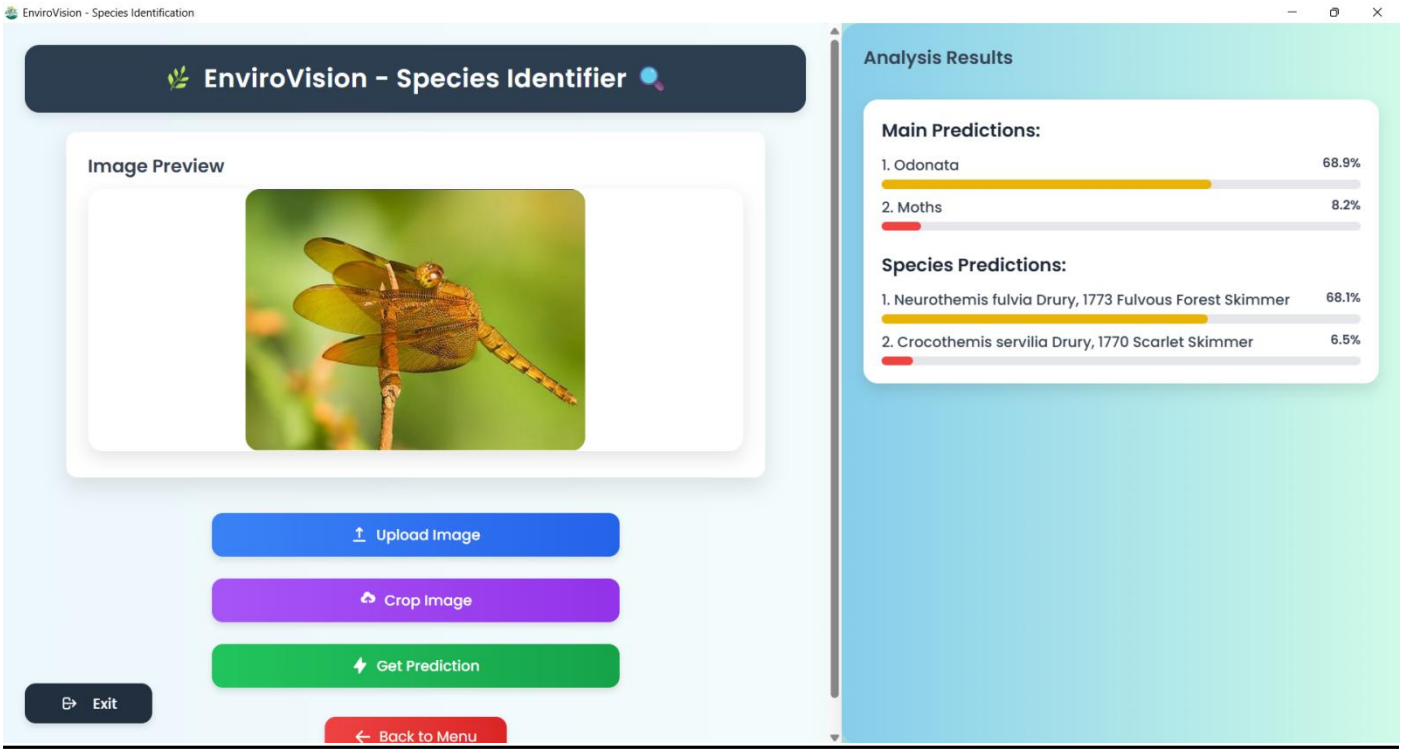

Fig 3.18 (ii) Odonata Prediction Output

CHAPTER – 5

CONCLUSION AND FUTURE ENHANCEMENTS

5.1CONCLUSION

The EnviroVision Species Identification Software provides an efficient, deep learning-based solution for real-time species classification in field research. Using a two-stage CNN architecture (XceptionNet for main categories and EfficientNetV2 Large for species-level identification), the system delivers accurate and fast results. It addresses the Zoological Survey of India's need for rapid, reliable, and offline-capable species identification, reducing manual effort and improving data accuracy.

This project demonstrates how deep learning can support biodiversity studies and streamline scientific workflows, especially in remote or resource-limited environments**.**

EfficientNetV2-Large was chosen for Sub CNNs not merely based on high observed accuracy, but due to its **state-of-the-art design**, **scalability**, and **proven effectiveness** in handling **fine-grained classification tasks on limited data**. This strategic selection has significantly enhanced the overall precision and reliability of our species identification software.

5.2 Future Enhancements

1. **Video-Based Identification** : Add support for identifying species from videos and live streams.
2. **Live Streaming Integration** : Enable real-time species detection during video streaming.
3. **Expand Species Coverage** : Train and integrate models for more species and categories.
4. **Mobile App Development** : Create a mobile version for better field usability.

REFERENCES

[1]. J. L. Allen and J. C. Lendemer, “An assessment of data accuracy and best practice recommendations for observations of lichens and other taxonomically difficult taxa on iNaturalist,” *Botany*, vol. 99, no. 6, pp. 345–352, 2021. [Online]. Available: https://doi.org/10.1139/cjb-2021-0160

[2]. H. Kim, J. Lee, and D. Park, “Assessing the Accuracy of Citizen Science Data Based on iNaturalist Data,” *Diversity*, vol. 14, no. 5, p. 316, 2022. [Online]. Available: https://doi.org/10.3390/d14050316

[3]. H. Goëau, P. Bonnet, and A. Joly, “Deep learning for plant identification: How the web can compete with human experts,” *Biodiversity Information Science and Standards*, vol. 1, p. e20530, 2017. [Online]. Available: https://doi.org/10.3897/tdwgproceedings.1.20530

[4]. S. Branson, “Facial recognition for birds is here,” *WIRED*, 2015. [Online]. Available: https://www.wired.com/story/facial-recognition-for-birds

[5]. Kahl, S., Wood, C. M., Eibl, M., & Klinck, H. (2021). BirdNET: A deep learning solution for avian diversity monitoring. Ecological Informatics, 61, 101236. https://doi.org/10.1016/j.ecoinf.2021.101236

[6]. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556. https://arxiv.org/abs/1409.1556

[7]. Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., ... & Belongie, S. (2018). The iNaturalist Species Classification and Detection Dataset. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 8769–8778. https://doi.org/10.1109/CVPR.2018.00914

[8].XceptionNet Architecture. Adapted from F. Chollet,”Xception: Deep Learning with Depthwise Separable Convolutions,”2017.https://images.app.goo.gl/vVx57

[9]. Fig 2.4.3 EfficientNetV2L Architecture. Adapted from M.Tan and Q.Le,”EfficientV2L:Smaller Models and Faster Training,”2021.https://images.app.goo.gl/PU89dCzjwzy7J7mD6