Enchanted Wings Project Documentation

1.Introduction

- Project Title**:** Enchanted Wings: Marvels of Butterfly Species
- Team Members:
    - ➢ Team Leader: Dharmavarapu Tejomurthy
    - ➢ Team Member: Gedela Revathi
    - ➢ Team Member: Kuchi Ramani

2. Project Overview

• Purpose:

The purpose of this project is to build a deep learning-based butterfly species classification system using transfer learning. The main goal is to accurately identify butterfly species from images by leveraging pre-trained convolutional neural networks like VGG16. This reduces training time and improves model efficiency. The system is designed to assist researchers, educators, and conservationists by providing real-time predictions, supporting biodiversity monitoring, ecological research, and educational outreach.

• Features:

- ➢ Classification of 75 butterfly species using image data.
- ➢ Utilizes transfer learning with pre-trained CNN models like VGG16 or ResNet50.
- ➢ High accuracy with reduced training time and computational cost.
- ➢ Simple user interface to upload images and view predictions instantly.
- ➢ Real-time butterfly species identification based on image input.
- ➢ Supports use cases in biodiversity monitoring, ecological research, and education.
- ➢ Easily extendable to include more species or deployment options in future.
- ➢ Promotes environmental awareness through AI-based technology.

3. Architecture

• Frontend:

The frontend of the application is built using Streamlit, a lightweight Python library for creating interactive web apps. It allows users to upload butterfly images and displays the predicted species name and confidence score in real-time. The interface is user-friendly and requires no prior technical knowledge, making it suitable for students, researchers, and the general public.

• Backend:

The backend is developed using Python with TensorFlow/Keras for deep learning. A pre-trained CNN model like VGG16 is used for transfer learning. The model is trained on a dataset of 6,499 butterfly images across 75 species. It handles image preprocessing, feature extraction, classification, and prediction logic.

• Database:

No traditional database system (like MongoDB or SQL) is used. Instead, the butterfly dataset is organized in a folder-based structure where each class (species) has its own directory. This structure is used for training, validation, and testing.

4. Setup Instructions

• Prerequisites:

- Python – Programming language used for development
- TensorFlow / Keras – For building and training the butterfly classification model using transfer learning
- Streamlit – To create a user-friendly web interface for image upload and prediction
- Jupyter Notebook or Google Colab – For training and testing the model during development

• Installation:

Clone or Download the Project Repository

```
git clone https://github.com/your-username/butterfly-classification.git
```

```
cd butterfly-classification
```

(Optional) Create and Activate a Virtual Environment

```
python -m venv venv
```

```
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

Install Required Packages

```
pip install tensorflow flask
```

Run the Flask Application

```
python app.py
```

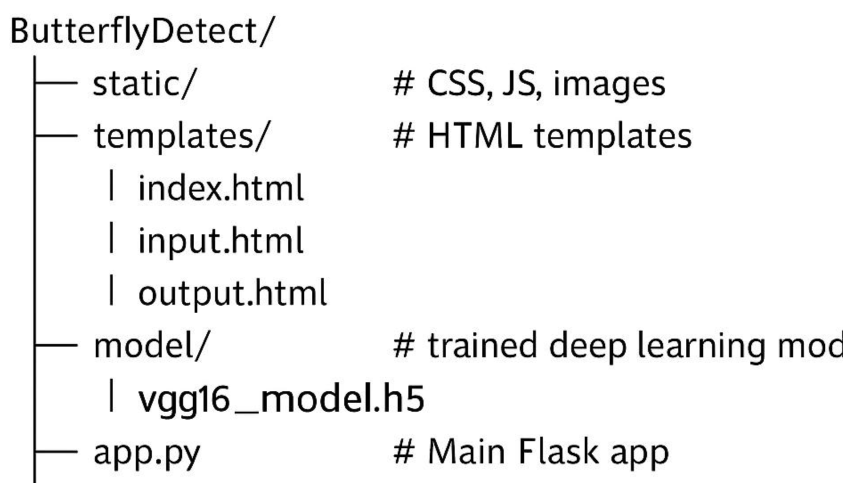
5. Folder Structure

• Client:

The frontend is created using Flask with HTML templates stored inside the templates folder. These templates handle user input and display output.

• Server:

The backend is developed using Flask . The logic for routing, file upload handling, and prediction is written in app.py. The model file vgg16_model. is loaded here to make predictions based on uploaded butterfly images.



```
ButterflyDetect/
├── static/              # CSS, JS, images
├── templates/           # HTML templates
│     | index.html
│     | input.html
│     | output.html
├── model/               # trained deep learning mod
│     | vgg16_model.h5
├── app.py               # Main Flask app
```

6. Running the Application

• Provide commands to start the frontend and backend servers locally.

This project runs as a single Flask application, where both the frontend (HTML templates) and backend (model prediction) are served from the same app.py file.

Command to Start the Flask Server:

python app.py

After running the above command, the application will start locally and can be accessed in your browser.

The frontend is rendered from the /templates/ directory.

The backend handles the uploaded image and returns the prediction result using the trained model.

There is no need to start separate frontend or backend servers, as everything is integrated within Flask.

7. API Documentation

The Flask backend of this butterfly classification project exposes two core endpoints that power the web interface shown in the application.

i. **/ – Homepage Endpoint**

- Method: GET
- Purpose: Loads the main interface where users can upload butterfly images.
- Parameters: None
- Response:
  Renders index.html, which contains:
  - A file input (Choose File)
  - A submit button labeled "Predict Butterfly"
  - Project title and team member details displayed on screen

ii. **/predict – Prediction Endpoint**

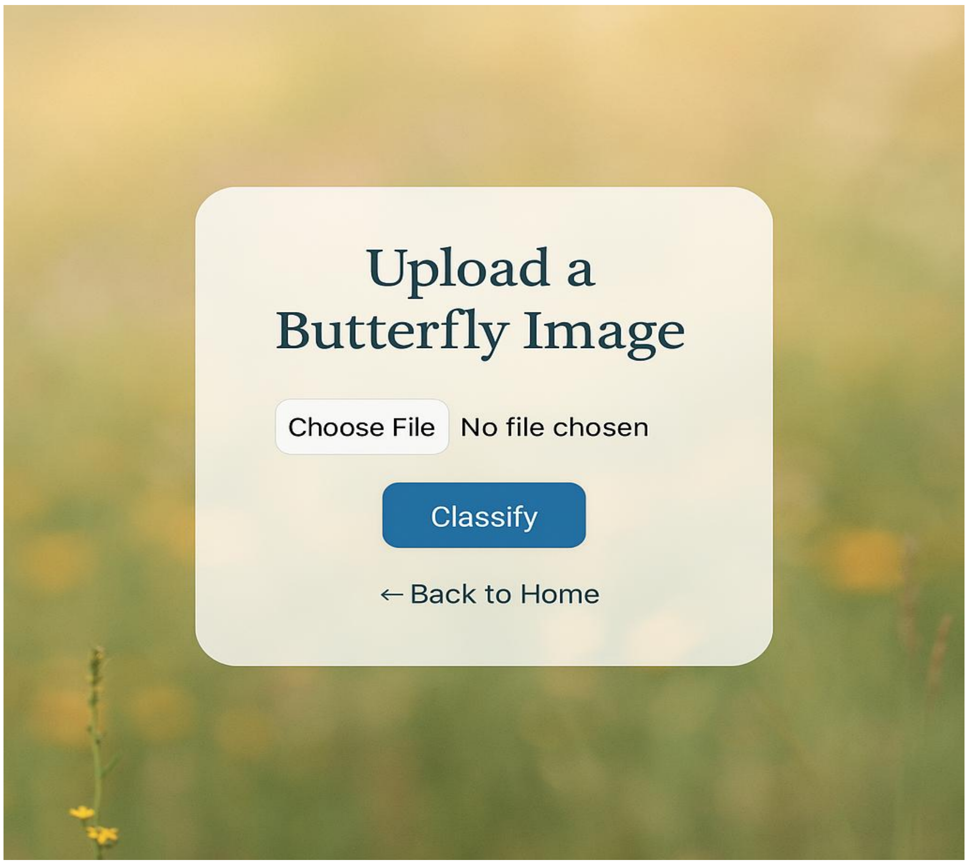
- Method: POST
- Purpose: Receives the uploaded image, performs preprocessing, runs the trained VGG16 model, and predicts the butterfly species.
- Parameters:
  - image: Image file uploaded by the user (from the HTML form)
- Response:
  Renders output.html displaying the predicted species.

8. Authentication

This project does not include authentication or authorization features, as it is designed for open, single-user access. The focus of the system is on classifying butterfly species from uploaded images without requiring user login or account management.

9. User Interface

The user interface is built using Flask and HTML. It allows users to upload butterfly images and view predictions. The interface includes a title, file upload button, prediction button, and displays the result along with team information and a thank-you message.



10. Testing

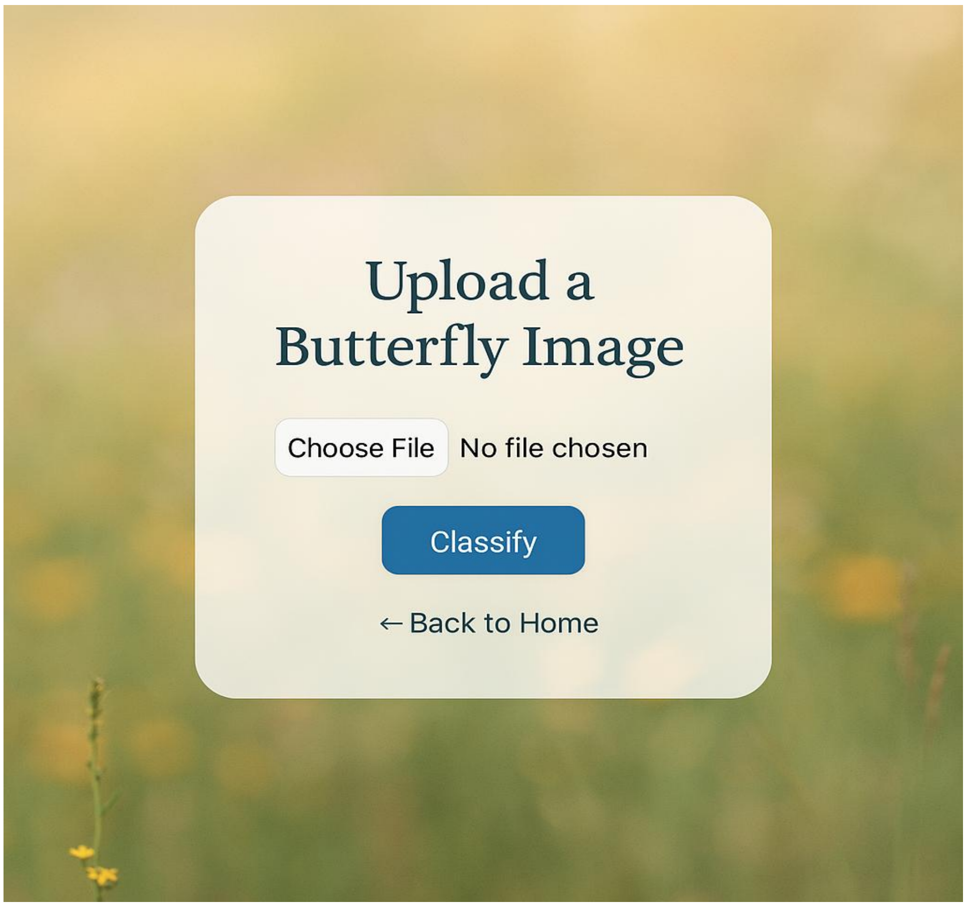
The testing for this project was focused on verifying the accuracy and functionality of the butterfly classification model and its integration with the Flask web interface.

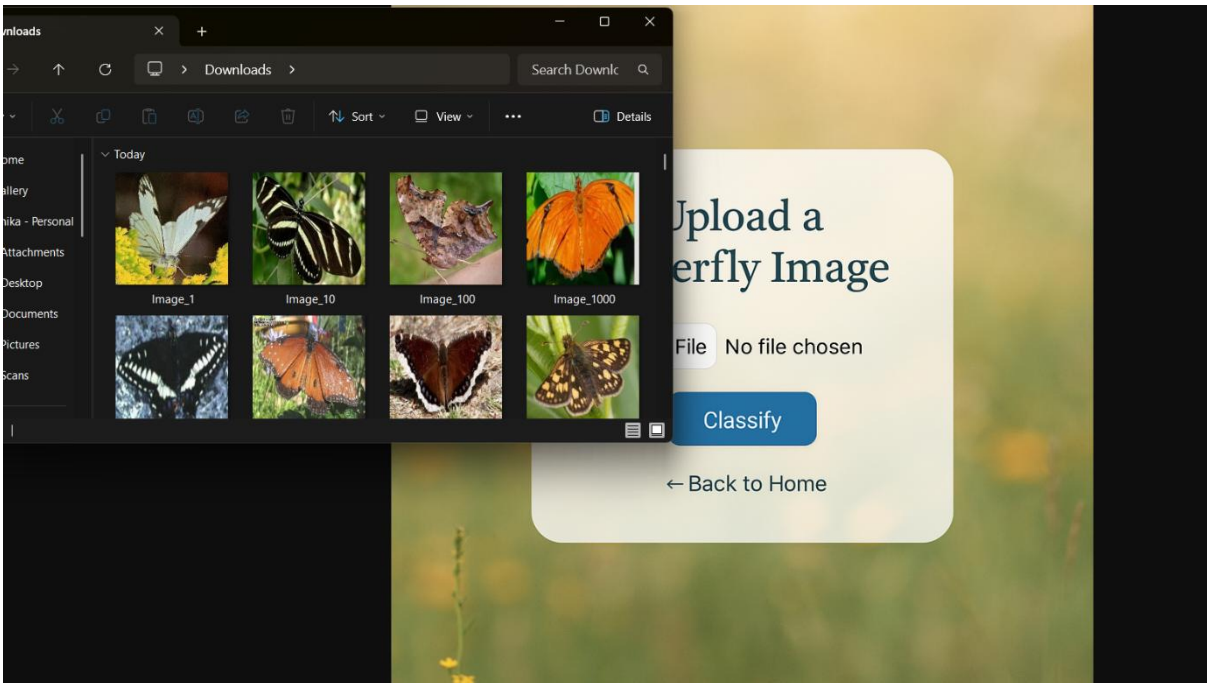
Testing Strategy:

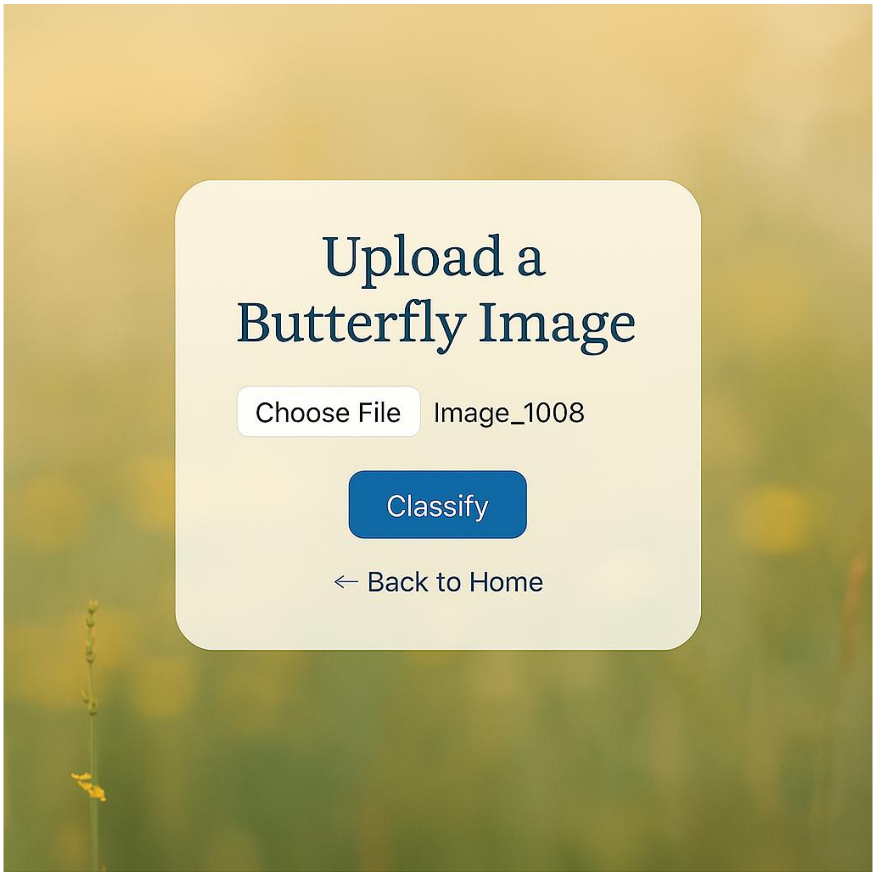
- **Model Testing**:
  - The dataset was split into training and testing sets.
  - The trained model was evaluated using the test images to check prediction accuracy.
  - Sample images were manually tested by uploading through the UI and verifying the predicted butterfly species.
- **Interface Testing**:

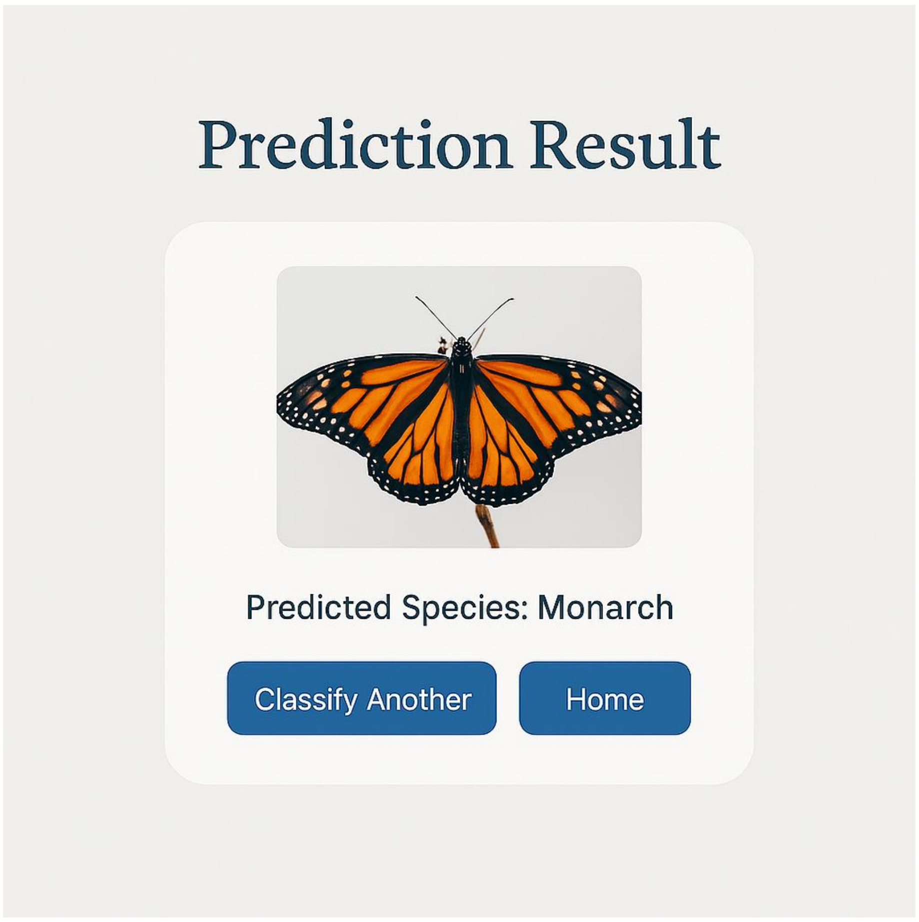
- The Flask web interface was tested by uploading various butterfly images.
- Verified whether the UI correctly accepted files and displayed the predicted output.
- Checked behavior for invalid inputs (e.g., no file selected, wrong file type).

11. Screenshots



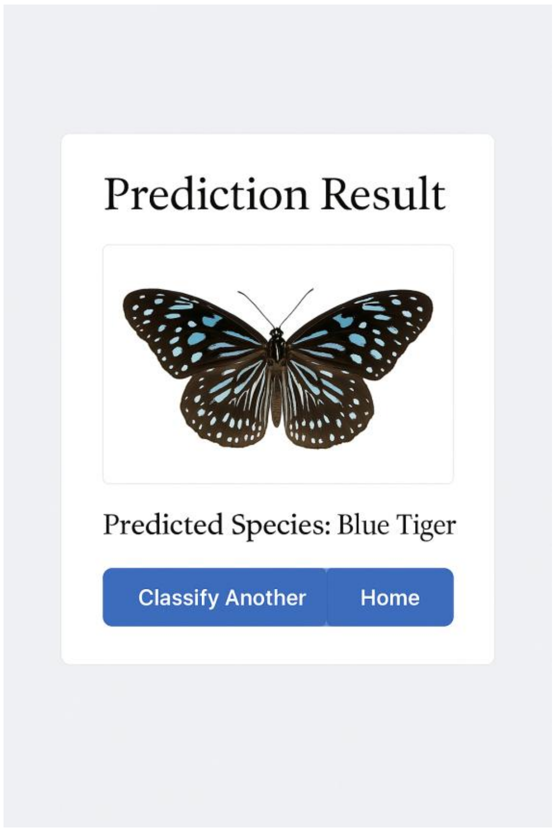

Downloads
Search Downlc
Sort
View
Details
Today
Image_1
Image_10
Image_100
Image_1000
Upload a
erfly Image
File No file chosen
Classify
← Back to Home


Upload a Butterfly Image
Choose File
Image_1008
Classify
← Back to Home



Prediction Result

Predicted Species: Monarch



Prediction Result

Predicted Species: Blue Tiger

12. Known Issues

While the butterfly classification system performs well overall, the following limitations and known issues were observed during testing:

- **Similar-Looking Species Confusion:**
  Some butterfly species that look very similar (in color or pattern) may be misclassified due to visual overlap.
- **Low-Quality Images Affect Accuracy:**
  Blurry, low-resolution, or poorly lit images may lead to incorrect predictions.

13. Future Enhancements

- **Host as a Web App or Mobile App** for easier public access.
- **Improve Model Accuracy** using more advanced deep learning architectures.
- **Add Support for More Image Formats** to avoid upload issues.
- **Enable Batch Image Upload** for multiple predictions at once.
- **Display Extra Info** about the predicted butterfly species for learning purposes.