Artificial Intelligence and Machine Learning

Project Documentation

1. Introduction

- **Project Title:** Enchanted Wings: Marvels of Butterfly Species - AI/ML Image Classification
- **Team Members:**
  - Mudda Narendra
  - Katta Rajeswari
  - Jessie Evans Nannam
  - Kode Yogitha

2. Project Overview

- **Purpose:** The primary purpose of the "Enchanted Wings" project is to develop a highly accurate and efficient image classification model for identifying diverse butterfly species. This aims to automate and streamline the identification process, reducing reliance on manual expert knowledge and accelerating data collection for biodiversity monitoring, ecological research, and citizen science initiatives.
- **Features:**
  - **Automated Butterfly Species Classification:** Identifies 75 distinct butterfly species

from images.
    - **Transfer Learning Approach:** Utilizes pre-trained Convolutional Neural Networks (CNNs) for efficient model training and high accuracy.
    - **Robust Performance:** Designed to handle variations in image quality, lighting, and angles.
    - **Real-time Potential:** Optimized for fast inference, enabling near real-time identification.
    - **Data-driven Conservation:** Provides a tool to aid in species inventory, population studies, and habitat management.
    - **Educational Engagement:** Supports citizen science projects by providing instant species identification for educational purposes.

3. Architecture

- **Model Architecture:** The core of the system is a deep learning model based on **Transfer Learning**. It utilizes a pre-trained Convolutional Neural Network (CNN) backbone (e.g., [Specific Pre-trained Model like ResNet50, MobileNetV2, InceptionV3 - *specify if known*]) as a feature extractor. The initial layers of this pre-trained model are frozen to preserve learned generic features. A custom classification head, consisting of fully connected (dense) layers, is appended to the backbone. This head is trained specifically on the butterfly dataset to classify the extracted features into 75 distinct species categories.
- **Data Pipeline:** The data pipeline involves several stages:
    - **Ingestion:** Raw butterfly images are ingested into the system.
    - **Preprocessing:** Images undergo resizing, normalization, and potential data augmentation (e.g., rotations, flips, zooms) to enhance model robustness.
    - **Training:** Preprocessed images are fed into the transfer learning model for training and fine-tuning.
    - **Inference:** For classification, new images are preprocessed and then passed through the trained model to obtain predictions.
- **Dataset:** The dataset comprises 6499 images across 75 unique butterfly species, partitioned into training, validation, and test sets to ensure proper model development and evaluation.

4. Setup Instructions

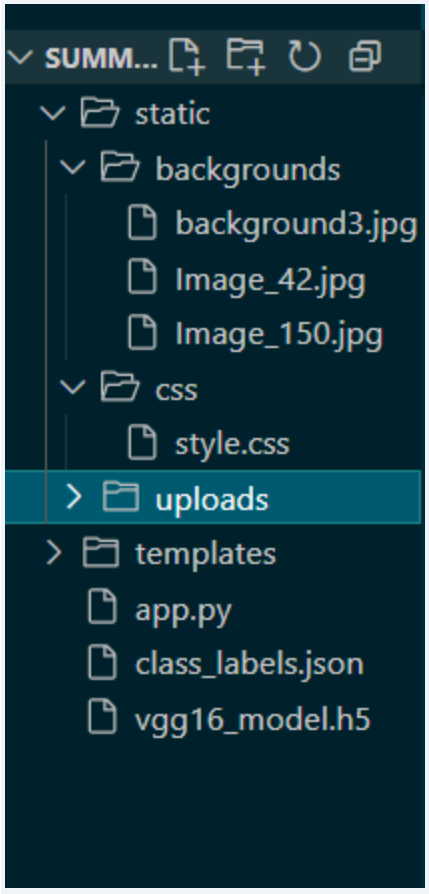
- **Prerequisites:**
  - Python (3.8+)
  - pip (Python package installer)
  - Access to a GPU (highly recommended for faster training, but CPU can be used for inference)
  - Sufficient disk space for the dataset and model checkpoints.
- **Installation:**
  1. **Clone the repository:**
     Bash
     git clone https://github.com/Narendranaid/enchanted-wings-marvels-of-butterfly-species/tree/main

  2. cd enchanted-wings-project

  3. **Create a virtual environment (recommended):**
     Bash
     python -m venv venv
     source venv/bin/activate # On Windows: .\venv\Scripts\activate

  4. **Install required Python packages:**
     Bash
     pip install -r requirements.txt

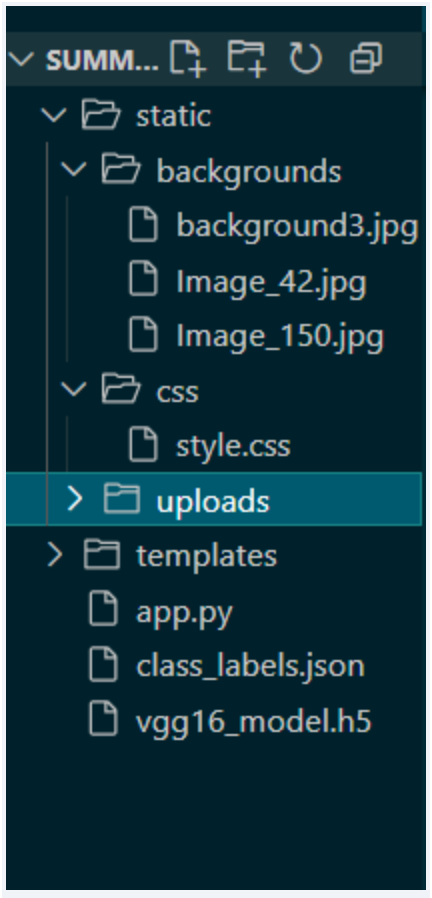
     (Assuming requirements.txt contains tensorflow or torch, numpy, pandas, scikit-learn, Pillow, matplotlib, etc.)
  5. **Download and organize the dataset:**
     - The dataset (6499 images, 75 classes) should be organized into train, validation, and test directories, with subdirectories for each species.
     - [Provide specific instructions on how to download or access the dataset, e.g., a link to a Kaggle dataset or a cloud storage bucket, and its expected directory structure.]
     - Folder structure:



6. **Set up environment variables (if any):**
   - [Specify any environment variables needed, e.g., DATA_PATH=/path/to/your/dataset]

5. Folder Structure

The project typically follows a structured layout for managing code, data, and models:



6. Running the Application (Training & Inference)

- **Training the Model:**
  - Navigate to the project root directory.
  - Run the training script (ensure your dataset is correctly set up as per Section 4.2):
    Bash
    python src/training/trainer.py --epochs 20 --batch_size 32 --model_name resnet50

    (Adjust parameters like epochs, batch size, and model name as needed. Refer to trainer.py for full options.)
- **Performing Inference/Prediction:**
  - To classify a new image using a trained model:
    Bash
    python src/prediction/predictor.py --image_path /path/to/your/image.jpg --model_path trained_models/best_model.h5

    (Replace /path/to/your/image.jpg with the actual path to the image you want to

classify and trained_models/best_model.h5 with your saved model.)
    - For a simple demo using a specific image, you might have a dedicated script:
      Bash
      python app.py

7. API Documentation (Conceptual for Deployment)

If the model were to be deployed as a service, it would typically expose an API. Below is a conceptual example:

- **Endpoint:** /predict/butterfly
    - **Method:** POST
    - **Description:** Accepts an image and returns the predicted butterfly species.
    - **Request Body (JSON):**
      JSON
      ```
      {
        "image_base64": "JVBERi0xLjQKJ..." # Base64 encoded image string
      }
      ```

    - **Example Response (JSON - Success):**
      JSON
      ```
      {
        "prediction": "Monarch Butterfly",
        "confidence": 0.987,
        "top_k_predictions": [
          {"species": "Monarch Butterfly", "confidence": 0.987},
          {"species": "Painted Lady", "confidence": 0.009},
          {"species": "Red Admiral", "confidence": 0.003}
        ]
      }
      ```

    - **Example Response (JSON - Error):**
      JSON
      ```
      {
        "error": "Invalid image format or processing error.",
        "status_code": 400
      ```

```
}
```

8. Authentication

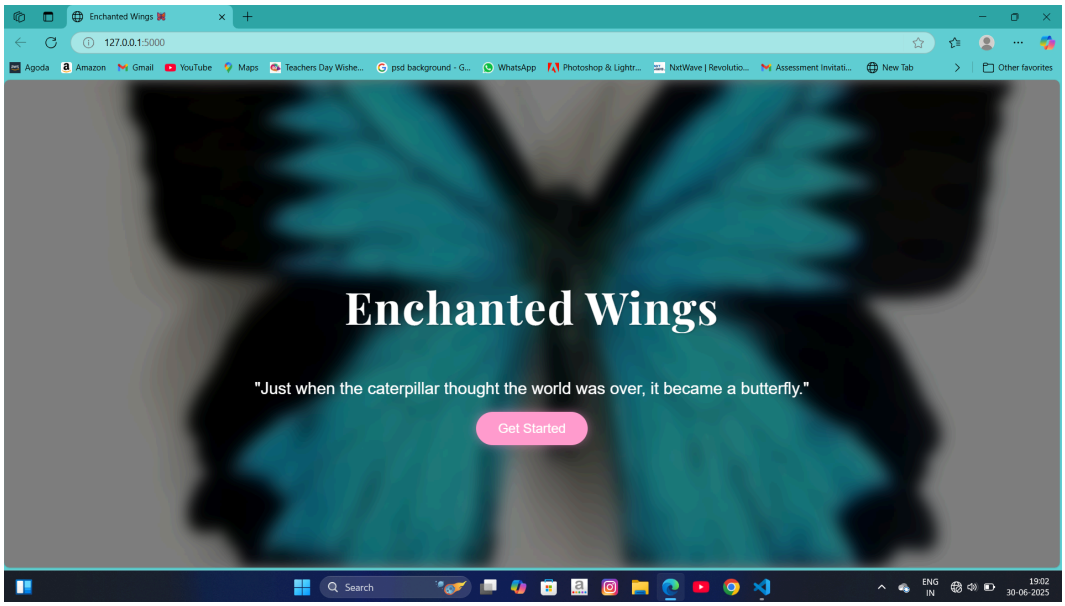
For a standalone AI/ML model, explicit user authentication (like in web applications) is often not inherent to the model itself. However, if this model were integrated into a larger application (e.g., a web portal, a mobile app, or an API service), authentication and authorization mechanisms would be handled at the application layer.

- **Conceptual Authentication (if part of an application):**
  - **API Key based:** Users accessing the prediction API would need to provide a valid API key in the request header or query parameters.
  - **Token-based (e.g., JWT):** For web or mobile applications, users would log in, receive a JSON Web Token (JWT), and include this token in subsequent requests to authenticate and authorize their access to the model's services.
  - **OAuth2:** For third-party integrations, OAuth2 could be used to grant delegated access to the prediction service.
- **No Authentication (for standalone model):** For the direct execution of the model scripts (training and inference) as described in Section 6, no explicit authentication mechanism is typically required as it runs within the user's local or cloud environment. Access is implicitly managed by system permissions.

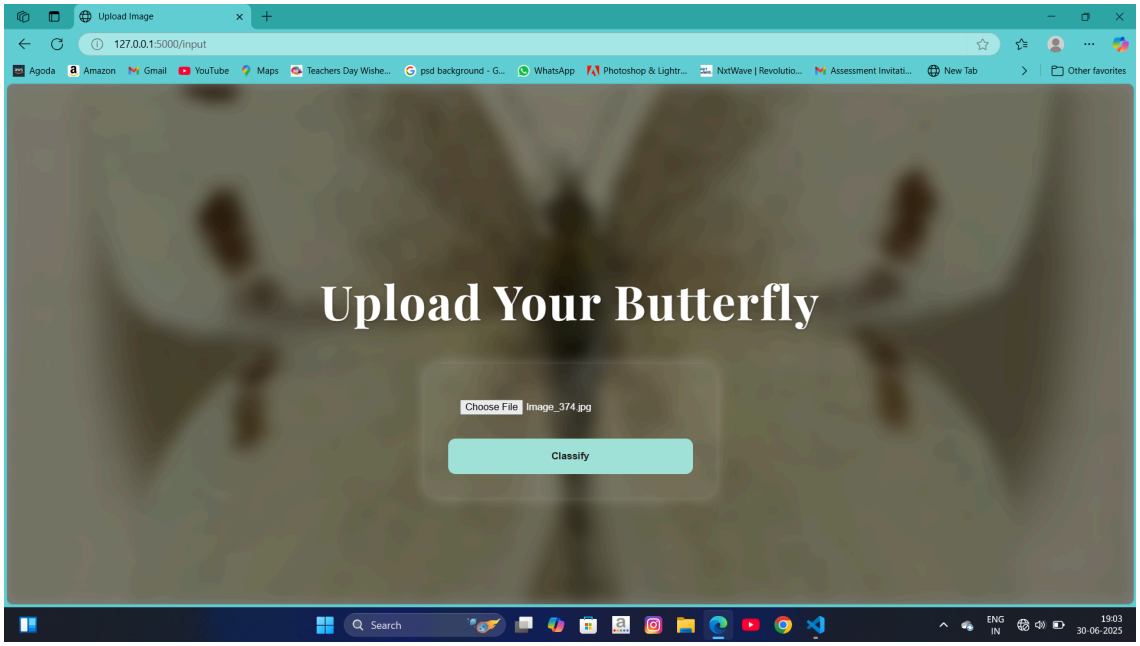
9. User Interface (Conceptual for a Demo/Application)

While the core project is the AI/ML model, for demonstration or deployment, a user interface would be essential.

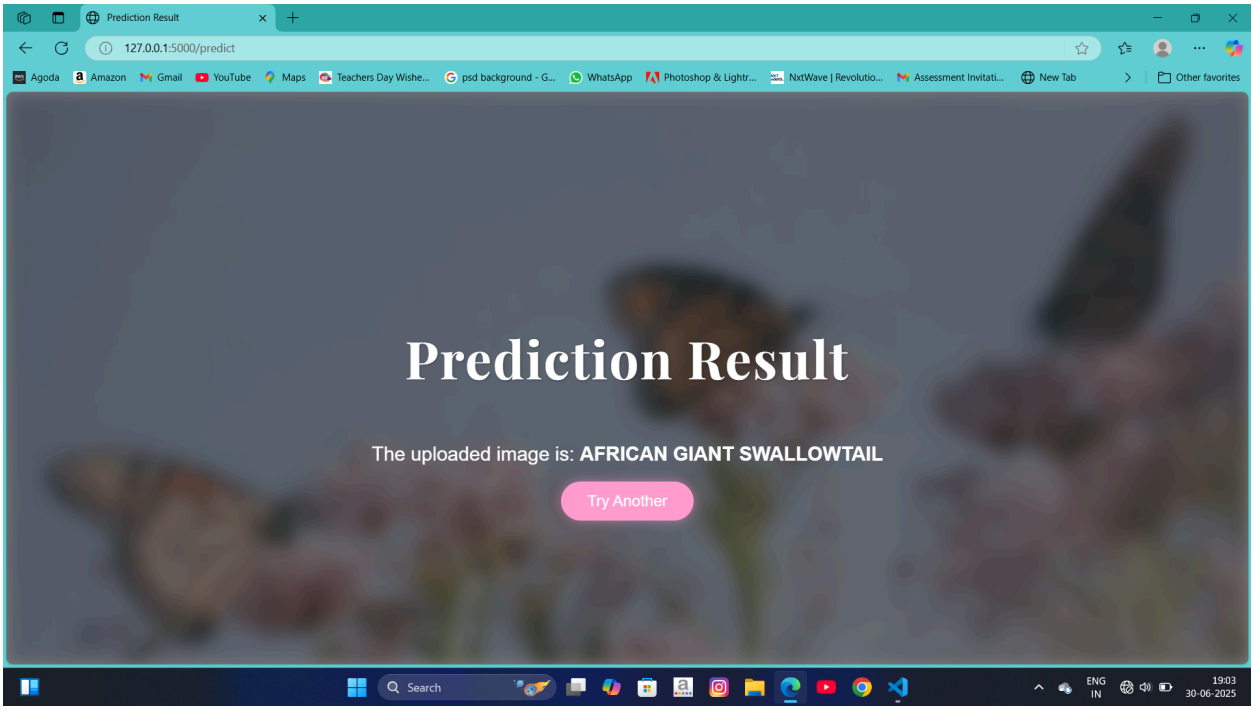
- **Web-based Interface (Example)**



UI Screenshot- 1



UI screenshot-2



UI Screenshot-3

10. Testing

The testing strategy for the "Enchanted Wings" project focuses on ensuring the model's accuracy, robustness, and generalization capabilities.

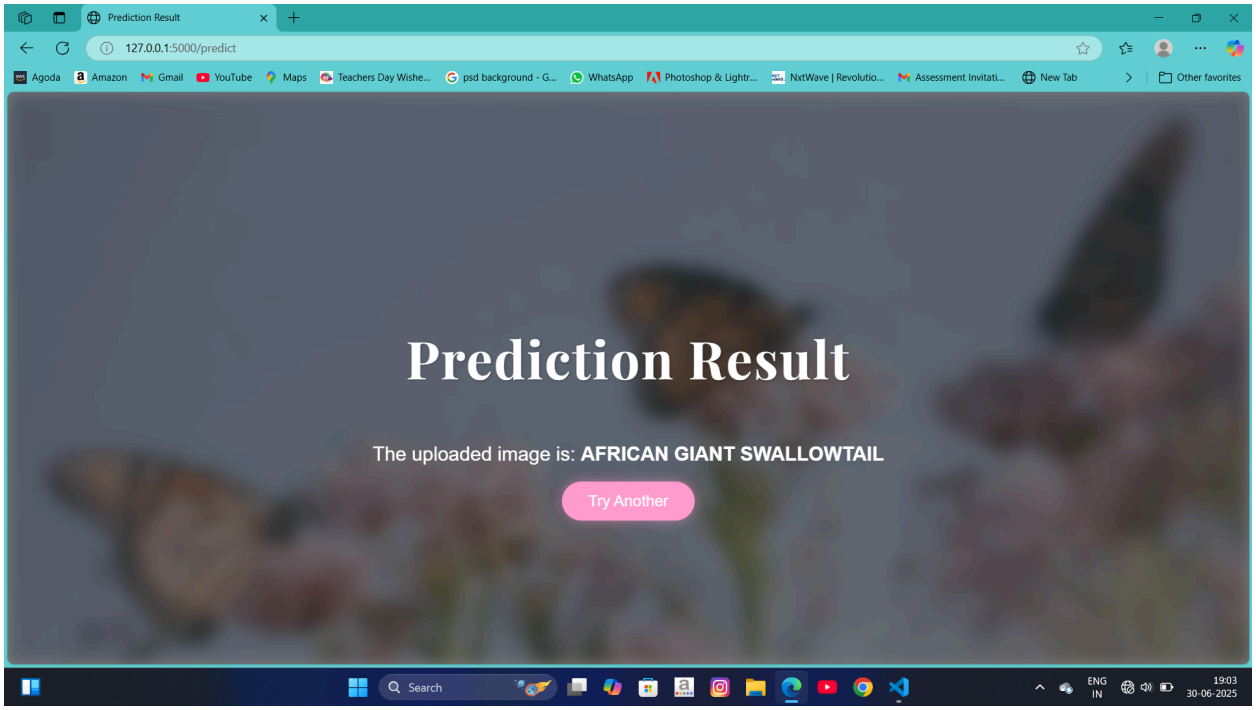
- **Unit Testing:**
  - **Data Preprocessing:** Verify individual data augmentation and preprocessing functions (e.g., image resizing, normalization) work as expected.
  - **Model Components:** Test custom layers or helper functions within the model architecture.
- **Integration Testing:**
  - **Data Pipeline:** Ensure images flow correctly from loading through preprocessing to model input.
  - **Training Loop:** Validate the training and validation steps, ensuring loss decreases and metrics improve as expected.
- **Model Evaluation:**
  - **Test Set Evaluation:** The primary method for assessing overall performance. The model's accuracy, precision, recall, and F1-score are calculated on the unseen test dataset.
  - **Confusion Matrix Analysis:** To understand which species are frequently misclassified and identify potential issues with specific classes.
  - **Robustness Testing:** Evaluating model performance on images with varying noise levels, blur, different lighting conditions, or partial occlusions.
- **Tools Used:**
  - Python's built-in unittest or pytest for unit and integration tests.
  - scikit-learn for calculating classification metrics.
  - matplotlib and seaborn for visualizing results (e.g., confusion matrices, training history plots).
  - TensorFlow/Keras or PyTorch built-in evaluation utilities.

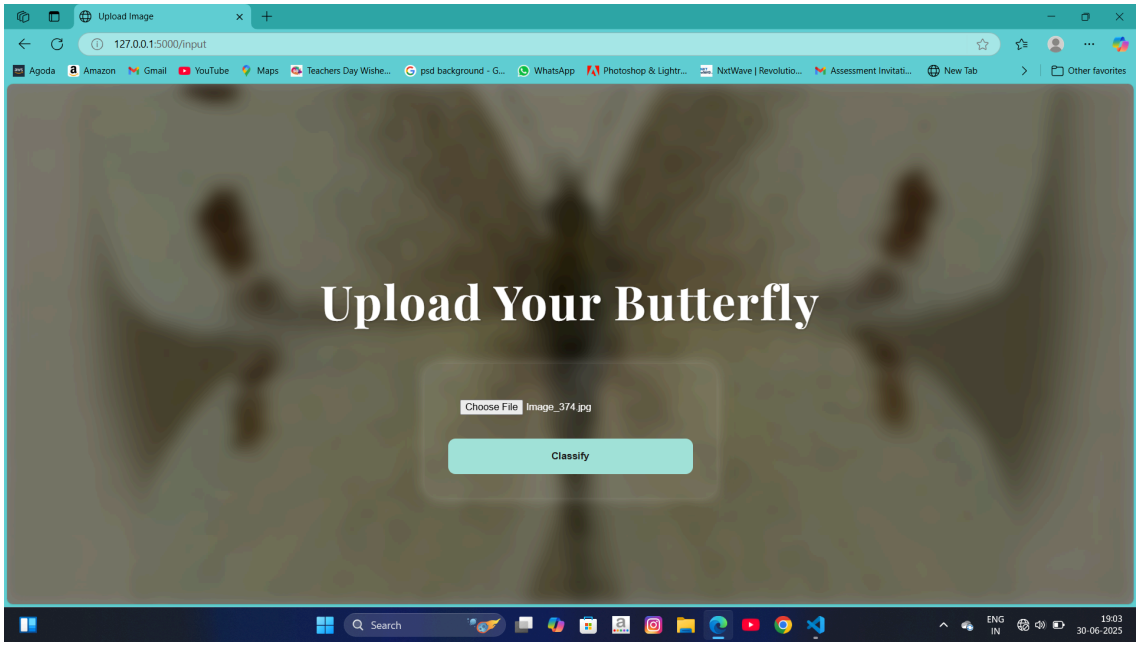
11. Screenshots or Demo

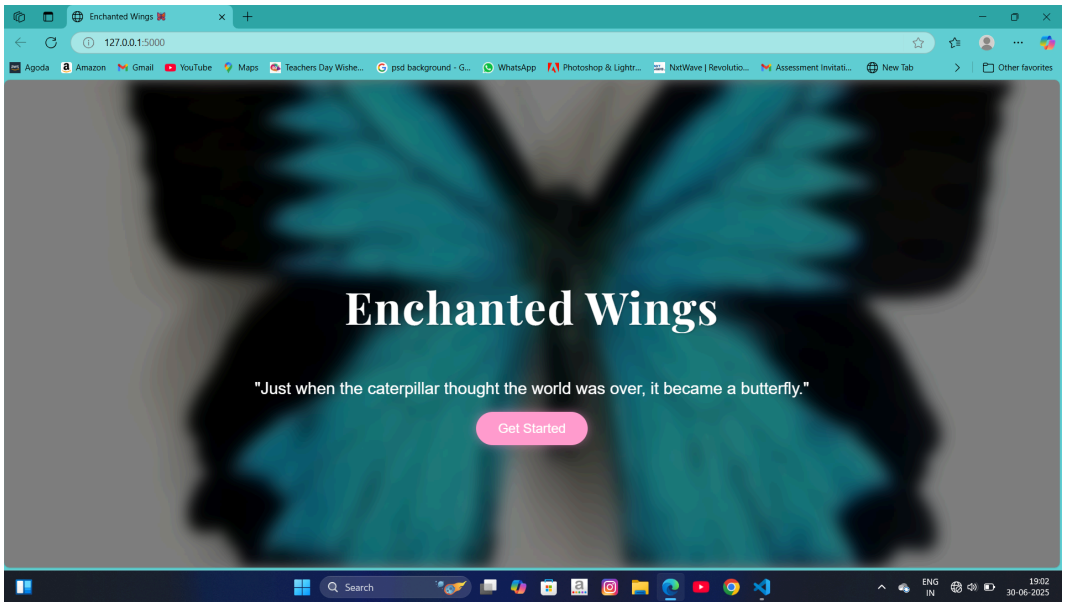
- **Model Training Progress:** A screenshot of a training log showing epochs, training loss, validation loss, training accuracy, and validation accuracy over time.
- **Example Prediction Output:** Screenshots demonstrating the input image and the model's prediction, including the species name and confidence score (as described in Section 7.1 of the previous report).
- **Classification Report:** A textual output showing per-class precision, recall, F1-score, and support for the test set.
- **Confusion Matrix:** A visual representation showing correct vs. incorrect classifications across all 75 species, highlighting areas where the model might struggle.
- **Link to Demo:** https://drive.google.com/drive/u/0/folders/1F2Fogk_I2yDjrm_CNe1elpyl4V9yCQxq

12. Known Issues

- **Fine-grained Distinctions:** Some closely related butterfly species might still pose a challenge for precise differentiation, especially if visual differences are subtle or the dataset contains limited examples for those specific species.
- **Occlusion Sensitivity:** Performance may degrade significantly if a large portion of the butterfly in the image is obscured.
- **Lighting and Background Variation:** While data augmentation helps, extreme variations in lighting conditions or complex, cluttered backgrounds can still impact accuracy.
- **Deployment Challenges:** Optimizing the large pre-trained models for real-time inference on very resource-constrained mobile or edge devices requires additional work (e.g., model quantization, pruning).
- **Data Imbalance:** If some species have significantly fewer images in the dataset, the model might perform less robustly on those rare classes.


Prediction Result
127.0.0.1:5000/predict
Prediction Result
The uploaded image is: AFRICAN GIANT SWALLOWTAIL
Try Another


Upload Image
127.0.0.1:5000/input
Upload Your Butterfly
Choose File
Image_374.jpg
Classify



13. Future Enhancements

- **Expand Dataset:** Continuously collect and integrate more diverse images for existing species, and add new butterfly/moth species to further improve robustness and coverage.
- **Advanced Data Augmentation:** Explore more sophisticated augmentation techniques (e.g., Generative Adversarial Networks - GANs for synthetic data) to improve generalization.
- **Model Optimization:** Implement techniques like model quantization, pruning, or knowledge distillation to reduce model size and inference time for deployment on edge devices.
- **Explainable AI (XAI):** Integrate XAI techniques (e.g., LIME, SHAP, Grad-CAM) to provide insights into *why* the model made a particular prediction, increasing trust and interpretability.
- **Multi-modal Input:** Explore combining image data with other contextual information

(e.g., geolocation, time of year, habitat type) to further enhance identification accuracy.
- **Interactive Learning/Feedback Loop:** Implement a mechanism within a deployed application for users to provide feedback on incorrect predictions, which can be used to retrain and improve the model over time.
- **Species Tracking and Monitoring Dashboards:** Develop dashboards that visualize identified species over time and location, providing valuable insights for conservation efforts.