Enchanted Wings: Marvels of Butterfly Species

Team information

Team ID : LTVIP2025TMID60178

Team Size : 4

Team Leader : Sakamuri Harika

Team member : Samarouthu Bala Gangadhara Sai Ram

Team member : Sankula Lakshmi Kalyani

Team member : Satyala Joseph Jeevan Paul

Executive Summary

This project focuses on creating a robust butterfly image classification model using transfer learning techniques. Leveraging a dataset comprising diverse butterfly species, including 75 classes with a total of 6499 images, the dataset is partitioned into training, validation, and test sets. Transfer learning utilizes pre-trained convolutional neural networks (CNNs) to accelerate model training by extracting relevant features from butterfly images. This method enhances classification accuracy while reducing computational resources and training time, ensuring efficient and effective species identification.

Scenarios

Scenario 1: Biodiversity Monitoring

In the context of biodiversity monitoring, a butterfly image classification system based on transfer learning can contribute significantly. Field researchers and conservationists can use this system to quickly identify butterfly species in diverse habitats. By capturing images in the field, the system identifies butterflies in real-time, aiding in species inventory, population studies, and habitat management efforts. This facilitates data-driven conservation strategies and promotes ecosystem health monitoring.

Scenario 2: Ecological Research

For ecological research, especially studies on butterfly behavior and distribution patterns, automated image classification systems are invaluable. Researchers can deploy cameras equipped with the classification system to monitor butterfly activities over extended periods. This enables tracking of migratory patterns, habitat preferences, and responses to environmental changes. The system's ability to accurately classify butterflies supports scientific discoveries and informs conservation practices aimed at preserving vulnerable species.

Scenario 3: Citizen Science and Education

Educational initiatives and citizen science projects benefit from interactive butterfly classification tools. These tools engage enthusiasts and students in butterfly identification and data collection. Users can capture butterfly images using mobile devices, and the classification system instantly provides species identification and relevant educational information. Such tools promote environmental awareness, citizen participation in scientific research, and foster a deeper understanding of butterfly ecology and conservation.

By applying transfer learning to butterfly image classification, this project not only advances scientific research and conservation efforts but also enhances public engagement and educational outreach in the field of biodiversity conservation.

Architecture

Prerequisites

To complete this project, you must require the following software, concepts, and packages:

- **Anaconda Navigator**: Refer to the link below to download Anaconda Navigator.
- **Python packages**: Open Anaconda prompt as administrator and install the following packages:

bash

RunCopy code

1pip install numpy

2pip install pandas

3pip install scikit-learn

4pip install matplotlib

5pip install scipy

6pip install seaborn

7pip install tensorflow

8pip install Flask

Project Structure

Create the Project folder which contains files as shown below:

- **Flask application** with HTML pages stored in the **templates** folder and a Python script **app.py** for scripting.
- **Vgg16_model.h5**: Our saved model for Flask integration.

Data Collection and Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. This section allows you to download the required dataset.

Collect the Dataset

There are many popular open sources for collecting the data, e.g., kaggle.com, UCI repository, etc. In this project, we have used 53 classes of playing cards data. This data is downloaded from kaggle.com or can be connected by using API. Please refer to the link given below to download the dataset.

LINK

As the dataset is downloaded, let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the Libraries

Import the necessary libraries as shown in the image.

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, or zip files, etc. We can read the dataset with the help of pandas.

At first, unzip the data and convert it into a pandas data frame.

Data Visualization

The provided Python code imports necessary libraries and modules for image manipulation. It selects a random image file from a specified folder path and displays the randomly selected image using IPython's Image module. This code is useful for showcasing random images from a directory for various purposes like data exploration or testing image processing algorithms.

This code snippet performs several tasks related to handling image files within a specified directory (**folder_path**). Initially, it lists all files in the directory that have file extensions commonly associated with image files (.jpg, .png, .jpeg). It then randomly selects one image file from the list of files retrieved. Finally, it constructs the full path to the randomly selected image file and displays it using the display function, assuming an environment where this function can render the image directly in the output.

Split Data and Model Building

Train-Test-Split

In this project, we have already separated data for training and testing.

Model Building: Vgg16 Transfer-Learning Model

The VGG16-based neural network is created using a pre-trained VGG16 architecture with frozen weights. The model is built sequentially, incorporating the VGG16 base, a flattening layer, dropout for regularization, and a dense layer with SoftMax activation for classification into five categories. The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss. During training, which spans 15 epochs, a generator is employed for the training data, and validation is conducted, incorporating callbacks such as Model Checkpoint and Early Stopping. The best-performing model is saved as "vgg16_model.h5" for potential future use. The model summary provides an overview of the architecture, showcasing the layers and parameters involved.

Testing Model & Data Prediction

Testing the Model

Here we have tested with the Vgg16 Model with the help of the **predict()** function.

Saving the Model

Finally, we have chosen the best model and saved that model.

Application Building

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the users where they have to enter the values for predictions. The entered values are given to the saved model, and the prediction is showcased on the UI.

This section has the following tasks:

- Building HTML Pages
- Building server-side script

Building HTML Pages

For this project, create three HTML files namely:

- **index.html** and save them in the **templates** folder.

UI Image Preview: Let's see what our **index.html** page looks like.

Now when you click on the inspect button further in the top right corner, you will get redirected to **Inspect.html**. Let's look at what our **inner.html** file looks like and test the model.

Test Class: tomato rotten (27)

Now when you click on the predict button, you will get output down to the image itself. Let's look at what our output looks like.

Test Class: potato rotten (23)

Now when you click on the predict button, you will get output down to the image itself. Let's look at how our output looks like.

Test Class: tomato healthy (26)

Build Python Code

Import the libraries.

- Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (**__name__**) as an argument.
- Here we will be using the declared constructor to route to the HTML page which we have created earlier.
- In the above example, the '/' URL is bound with the **index.html** function. Hence, when the index page of the web server is opened in the browser, the HTML page will be rendered. Whenever you enter the values from the HTML page, the values can be retrieved using the POST Method.

Retrieves the Value from UI

Here we are routing our app to the **output()** function. This function retrieves all the values from the HTML page using a POST request. That is stored in an array. This array is passed to the model. The **predict()** function returns the prediction. This prediction value will be rendered to the text that we have mentioned in the **output.html** page earlier.

Main Function

- Open Anaconda prompt from the start menu.
- Navigate to the folder where your Python script is.
- Now type the **app.py** command.
- Navigate to the local host where you can view your web page.
- Click on the inspect button from the top right corner, enter the inputs, click on the predict button, and see the result/prediction on the web.

Now, go to the web browser and write the localhost URL (http://127.0.0.1:5000) to get the below results.

Run the Web Application

UI Image Preview: Let's see what our **index.html** page looks like.

By clicking on "Get Started," it will redirect us to the input page, i.e., the prediction page.

Input.html:

Output.html:

Uploading another Image:

Output:

Now when you click on the predict button, you will get output down to the image itself. Let's look at how our output looks like.

Test Class: strawberry_healthy (24)

Now when you click on the predict button, you will get output down to the image itself. Let's look at how our output looks like.

Test Class: Apple_healthy (0)

Now when you click on the predict button, you will get output down to the image itself. Let's look at how our output looks like.

Project timeline & Milestones

Week 1: Project Setup and Prerequisites

- **Objective**: Set up the project environment and install necessary software and packages.
- **Tasks**:
    - Download and install **Anaconda Navigator**.
    - Open Anaconda prompt as administrator.
    - Install required Python packages:
        - **numpy**
        - **pandas**
        - **scikit-learn**
        - **matplotlib**
        - **scipy**
        - **seaborn**
        - **tensorflow**
        - **Flask**

- **Deliverables**: A fully set up development environment ready for data collection and model building.

Week 2: Data Collection and Preparation

- **Objective**: Collect and prepare the dataset for training the model.
- **Tasks**:
    - Identify and download the butterfly dataset (75 classes, 6499 images).
    - Unzip the dataset and convert it into a pandas DataFrame.
    - Perform initial data exploration and visualization to understand the dataset.
- **Deliverables**: A cleaned and organized dataset ready for training.

Week 3: Data Visualization and Understanding

- **Objective**: Visualize the dataset to gain insights.
- **Tasks**:
    - Import necessary libraries for data visualization.
    - Write code to randomly select and display images from the dataset.
    - Use various visualization techniques to analyze the distribution of butterfly species.
- **Deliverables**: Visualizations that provide insights into the dataset, including species distribution and image examples.

Week 4: Model Building

- **Objective**: Build the VGG16 transfer-learning model.
- **Tasks**:
    - Create a VGG16-based neural network using a pre-trained architecture.
    - Freeze the weights of the VGG16 base and add custom layers (flattening, dropout, dense).
    - Compile the model using the Adam optimizer and sparse categorical cross-entropy loss.

    - Train the model for 15 epochs, using callbacks for model checkpointing and early stopping.
- **Deliverables**: A trained VGG16 model saved as "vgg16_model.h5".

Week 5: Model Testing and Prediction

- **Objective**: Test the model and make predictions.
- **Tasks**:
    - Use the **predict()** function to test the model with sample images.
    - Evaluate the model's performance on the test dataset.
    - Save the best-performing model for future use.
- **Deliverables**: A tested model with documented performance metrics.

Week 6: Application Development

- **Objective**: Develop a web application to integrate the model.
- **Tasks**:
    - Create HTML pages for the user interface (index.html, input.html, output.html).
    - Build the Flask server-side script to handle user inputs and predictions.
    - Implement routing to connect the HTML pages with the Flask application.
- **Deliverables**: A functional web application that allows users to upload images and receive predictions.

Week 7: Testing and Deployment

- **Objective**: Test the web application and prepare for deployment.
- **Tasks**:
    - Conduct thorough testing of the web application to ensure all functionalities work as expected.
    - Gather feedback from potential users and make necessary adjustments.

    - Prepare documentation for the application, including installation and usage instructions.
- **Deliverables**: A fully functional web application ready for deployment.

Week 8: Final Review and Presentation

- **Objective**: Review the entire project and prepare for presentation.
- **Tasks**:
    - Compile all project documentation, including code, results, and insights.
    - Prepare a presentation summarizing the project objectives, methods, results.
    - Conduct a final review of the project with stakeholders or peers.
- **Deliverables**: A comprehensive project report and presentation materials.

Conclusion

The "Enchanted Wings: Marvels of Butterfly Species" project successfully demonstrates the application of transfer learning techniques in the field of image classification, specifically for identifying diverse butterfly species. By leveraging a robust dataset of 6499 images across 75 classes, the project not only advances the capabilities of automated species identification but also contributes significantly to biodiversity monitoring, ecological research, and educational outreach.