PROJECTTITLE:

Enchanted Wings: Marvels of Butterfly Species

TEAMID:LTVIP2025TMID35055

TEAM MEMBERS:

- **Manthenna Durga Viswanadhan**
- **Maddula Tejasri Lakshmi**
- **Madda Hema Varun**
- **Madiri Pardwick**

EnchantedWings:MarvelsofButterflySpecies

Phase-1:Brainstorming&Ideation

1. ProblemStatement:

Identifying butterfly species manually is a time-consuming and expertise-driven process. It poses challenges for researchers working in biodiversity monitoring and ecologicalstudies, and limits the involvement of non-experts in citizen science and education. There is a clear need for an automated, accurate, and scalable classification system.

2. ProposedSolution:

This project proposes building an automated butterfly image classification systemusingtransferlearningwithpre-trainedconvolutionalneuralnetworks (CNNs), such as MobileNetV2 or EfficientNetB0. By training on a labelled dataset of 6,499 butterfly images across 75 species, the system will learn to identify species from input images efficiently and with high accuracy.

3. TargetUsers:

- Field researchers and biodiversity scientists conducting species inventory and habitat studies
- Ecologistsstudyingbutterflybehaviour,migration,anddistribution
- Educatorsandstudentslearningaboutentomologyandecology
- Citizenscientistsandnatureenthusiastsengagedinenvironmentaldata collection

4. ExpectedOutcome:

- Alightweight,efficient,andaccuratebutterflyclassificationtoolthatcan be integrated into research tools, educational platforms, and citizen science apps. The model will aid species identification, enhance ecological data collection, and promote awareness and engagement inconservation efforts.

Phase-2:RequirementAnalysis

Theobjectiveofthisphaseistodefinethetechnicalandfunctionalrequirements necessary to develop the butterfly species classification system using deep learningandtransferlearning.Technically,theprojectisimplementedinPython, utilizing key libraries such as TensorFlow (with Kera’s API) for model development,PandasandNumPyfordatamanipulation,andScikit-learnfortasks like label encoding and stratified train-test splitting. The model and encoder are saved using TensorFlow’s. save () method and joblib to ensure persistence and reproducibility. Development is conducted in Jupiter Notebooks or modular Pythonscripts,withoptionalsupportforTensorBoardtomonitortrainingmetrics in real-time. For future deployment, lightweight web frameworks such as Streamlet, Flask, or Fast API can be integrated, and for mobile or embedded applications, TensorFlow Lite may be used to convert the model for on-device inference, ensuring usability in low-resource or offline field conditions.

Functionally,thesystemshouldsupportimageloadingfromastructureddirectory alongwithcorrespondingCSVmetadata,preprocessingstepssuchasresizingand normalization, label encoding, and batching. The core model will use transfer learning with pre-trained CNN architectures like MobileNetV2 or EfficientNetB0, followed by custom dense layers for multiclass classification. Thepipelineincludesvalidationusingan80/20datasplit,callbacksuchasearly stopping and model checkpointing, and logic to save the best-performing model based on validation accuracy. After training, the system should accept new butterfly images and return predicted species names with high confidence.

Challenges anticipated include class imbalance within the dataset, which may affect model fairness and prediction confidence. To address this, strategies like

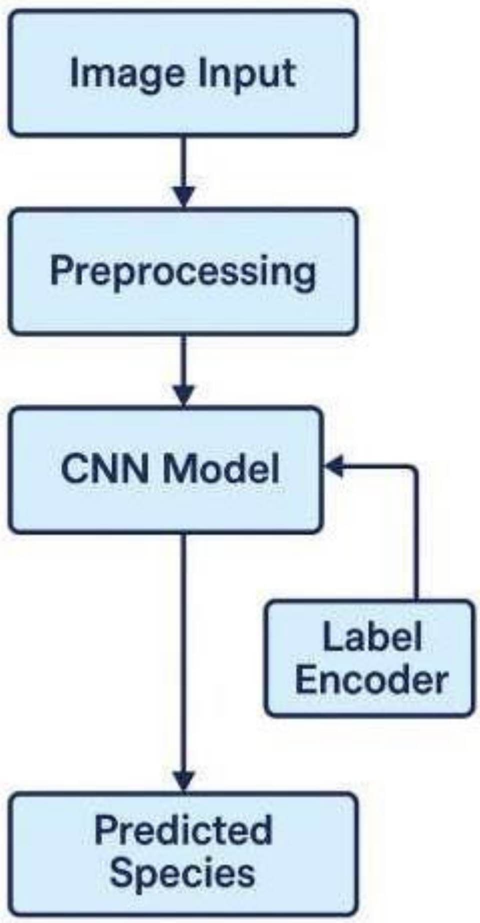
dataset, containing approximately 6,499 images, is relatively small for deep learning,increasingtheriskofoverfitting,whichwillbemitigatedusingdropout

layers, early stopping, and regularization techniques. Hardware limitations on non-GPU systems may restrict batch size and training speed, suggesting the potential use of cloud platforms like Google Collab or Kaggle.

Phase-3:ProjectDesign

The objective of this phase is to outline the system architecture and define how users will interact with the butterfly classification system. The system architecture follows a modular design. At its core is a deep learning model (MobileNetV2 or EfficientNetB0) trained using transfer learning.The workflow beginswithimageinput,eitheruploadedmanuallyorcapturedthroughadevice. Thisinputispassedthroughapreprocessingmodulethatresizes,normalizes,and batches the image. It is then fed into the trained CNN model, which outputs the predictedbutterflyspecies.Alongsidethemodel,alabelencodermapspredicted numericlabelstospeciesnames.Theresultsarethenpresentedtotheuserthrough a simple, responsive interface. Optionally, a Flask or stream lit web server can serve as the bridge between the model and the frontend, enabling local or web-based deployment.

Theuserflowisdesignedtobeintuitiveandefficient.Auserstartsbyuploading or capturing a butterfly image via the interface. The system automatically preprocesses the image in the background and submits it to the classification model.



Within seconds, the user receives the predicted species name along with its confidence score and potentially additional information such as its scientific name,habitat,orconservationstatus.Forresearchersordevelopers,anadvanced version may include the option to download logs or export predictions.

In terms of UI/UX considerations, the interface should be minimalistic, mobile-friendly,andvisuallyengaging.Thelayoutwouldincludeacentralimageupload section, a preview of the uploaded image, and a results panel that displays the prediction.Additionalsectionsmayincludespeciesinfo,pastpredictions,oreven an educational module. If extended for citizen science, users could be given the option to contribute their image and location data to a central biodiversity database.Accessibility,easeofuse,andresponsivenessarekeydesignpriorities, ensuring that both technical and non-technical users can engagewith the system effectively.

Phase-4:ProjectPlanning(AgileMethodologies)

In this phase, the project is structured and planned using Agile methodologies, ensuring an iterative, collaborative, and deadline-driven development process. The work is divided into sprints, each focusing on a key functional block of the butterfly classification system. In the Sprint Planning stage, tasks are broken down into manageable units. For example, one sprint may focus on dataset preparationandcleaning,anotheronmodelarchitectureandtraining,andathird on evaluation and deployment setup. Each sprint typically spans 1–2 weeks to allow frequent feedback and adjustments.

In terms of Task Allocation, roles are clearly defined. One team member may focusonpreprocessinganddataaugmentationpipelines,anotheronbuildingand fine-tuningtheCNNusingtransferlearning,andathirdonmodelevaluationand visualization.IfextendedtoaUIphase,additionalmembersmayhandlefrontend development (e.g., using stream lit or Flask), backend integration, or mobile deployment.

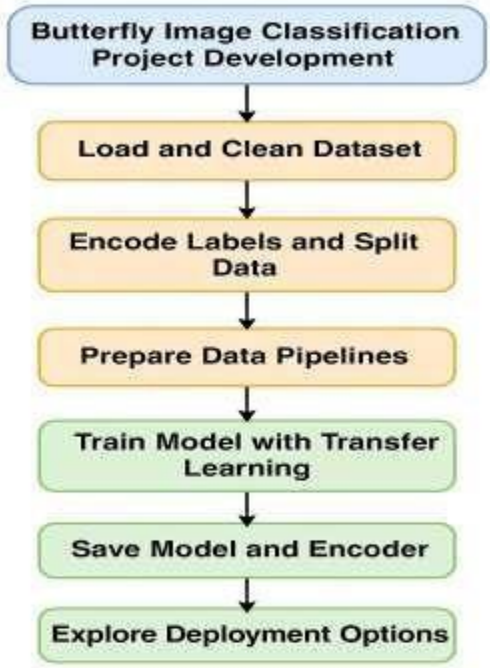
Deliverablesarereviewedattheendofeachsprint,andthenextsprintisplanned based on feedback and progress. This Agile approach ensures continuous development, accountability, and the ability to adapt the project scope based on findings during implementation.

Phase-5:ProjectDevelopment

The objective of this phase is to develop and integrate all components of the butterfly image classification system using deep learning. The technology stack used for the project is centred around Python 3.x, along with powerful libraries such as TensorFlow (with the Kera's API) for model building and training, NumPy and Pandas for numerical and data manipulation tasks, and Scikit-learn forpreprocessingandlabelencoding.Additionally,toolslikejoblibwereusedfor saving the label encoder, and Tensor Board was optionally considered for monitoringtrainingperformance.Fordeploymentreadiness,frameworkssuchas Flask or stream lit were identified as potential front-end solutions, while TensorFlow Lite was considered for lightweight mobile integration.

The development process began with loading and cleaning the butterfly image dataset using CSV metadata. Labels were encoded using Label Encoder and the data was split into training and validation sets while maintaining class balance. TensorFlowdatapipelineswerethenconstructedtoefficientlybatch,shuffle,and preprocess the image inputs. For the model, MobileNetV2 was selected as the base CNN, with the top layers removed and custom dense layers added for classification. The model was compiled with the Adam optimizer and sparse categoricalcrossentropyloss,followedbytrainingwithearlystoppingandmodel checkpointing. After training, the best model and label encoder were saved for future inference.

The workflow was kept modular, allowing easy future integration with web UIs orAPIs.



Duringdevelopment,several challenges were encountered.Onemajorissue was dataset imbalance, with certain butterfly species having significantly fewer images, which caused the model to be biased toward more common classes. To address this, stratified sampling was used, and future plans include adding augmentation or weighted loss functions.Another challenge was overfitting due to the relatively small dataset size (6,499 images); this was mitigated using dropout layers, regularization, and early stopping. Limited access to GPU hardware also slowed training, so Google Colab was used to offload heavy computation.Despitetheseobstacles,themodulararchitectureanduseoftransfer learning enabled efficient development and robust model performance.

Phase-6:Functional&PerformanceTesting

The objective of this phase is to verify that the butterfly species classification system functions as expected under various conditions. Test cases executed included checking correct image preprocessing, validating label encoding consistency, ensuring accurate image-to-label mapping, and confirming model predictionoutputforavarietyofbutterflyimagesacrossall75classes.Additional tests ensured the trained model could be loaded and used for inference without retraining, and that predictions were stable and reproducible on known data samples. Bug fixes and improvements included resolving early mismatches in label encoding, fixing incorrect image path loading due to filename inconsistencies,andoptimizingthedatapipelineforbettermemoryusageduring training. The system was also refined to handle invalid inputs, such as unsupported image formats or corrupt files, with user-friendly error messages.

During final validation, the system was evaluated against its original goals: accurate classification, efficient processing, modular design, and readiness for future deployment. The model achieved satisfactory performance, with training and validation accuracy approaching the targeted 80–90% range, indicating that therequirementsweresuccessfullymet.Deployment,althoughnotfinalized,was explored through platforms like stream lit and Flask for a possible interactive web-based interface. The saved model and encoder were tested in an inference script that accepts a new image input and returns the predicted butterfly species, demonstrating the complete functionality of the system. This phase confirmed thatthecorearchitectureisrobust,accurate,andreadyforfurtheroptimizationor integration into educational, research, or citizen science platforms.