FINAL REPORT

Enchanted Wings: Marvels of Butterfly Species

1. INTRODUCTION

1.1 Project Overview

Enchanted Wings: Marvels of Butterfly Species is an artificial intelligence project that leverages deep learning and transfer learning techniques to classify butterfly species. The goal is to build a robust image classification model using a dataset of 6,499 images across 75 butterfly species. The solution enhances scientific research, biodiversity monitoring, and citizen science.

1.2 Purpose

The project aims to automate butterfly species identification using a pre-trained Convolutional Neural Network (CNN), thereby reducing training time and improving classification accuracy. This supports ecological research, educational outreach, and conservation initiatives.

2. IDEATION PHASE

2.1 Problem Statement

Manual butterfly species identification is time-consuming and error-prone, requiring expert knowledge. An automated, image-based identification system can assist researchers, educators, and the public.

2.2 Empathy Map Canvas

- Who?: Researchers, conservationists, educators, students, and citizen scientists
- Needs?: Fast, accurate butterfly identification
- Think & Feel?: Curious, committed to environmental conservation
- Hear?: Complaints about identification difficulties in fieldwork
- See?: Manual reference guides, labelling errors
- Do?: Capture and upload butterfly images for classification

2.3 Brainstorming

Ideas explored:

- Use of CNNs like VGG16 or ResNet

- Mobile-based classification app
- Educational integration for schools
- Real-time image capture and classification in field

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. Capture butterfly image
2. Upload to system
3. System processes image using pre-trained CNN
4. Display predicted species name
5. (Optional) Show species info for education

3.2 Solution Requirements

- Dataset with labeled butterfly images
- Image preprocessing pipeline
- Pre-trained CNN model (VGG16)
- Model training and evaluation logic
- User interface for image upload and prediction (optional)

3.3 Data Flow Diagram

1. Input: Butterfly image
2. Preprocessing (resizing, normalization)
3. VGG16 model (feature extraction)
4. Dense layer (classification)
5. Output: Predicted class

3.4 Technology Stack

- Language: Python
- Libraries: TensorFlow, Keras, Pandas, NumPy
- Model: VGG16 (Transfer Learning)
- Environment: Jupyter Notebook

4. PROJECT DESIGN

4.1 Problem-Solution Fit

Problem: Manual butterfly classification
Solution: Deep learning-based image classifier

4.2 Proposed Solution

Use VGG16 (pre-trained CNN) as a feature extractor, add custom dense layer for classifying 28 species (subset of 75 for this implementation), freeze convolution layers, and train classifier using augmented data.

4.3 Solution Architecture

1. Load butterfly dataset
2. Preprocess with ImageDataGenerator
3. Load VGG16 model (without top layer)
4. Add Flatten and Dense layer (28 outputs)
5. Compile and train
6. Save model for deployment

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

- Week 1: Dataset acquisition and preprocessing
- Week 2: Model design and implementation using VGG16
- Week 3: Model training and evaluation
- Week 4: UI design, documentation, testing

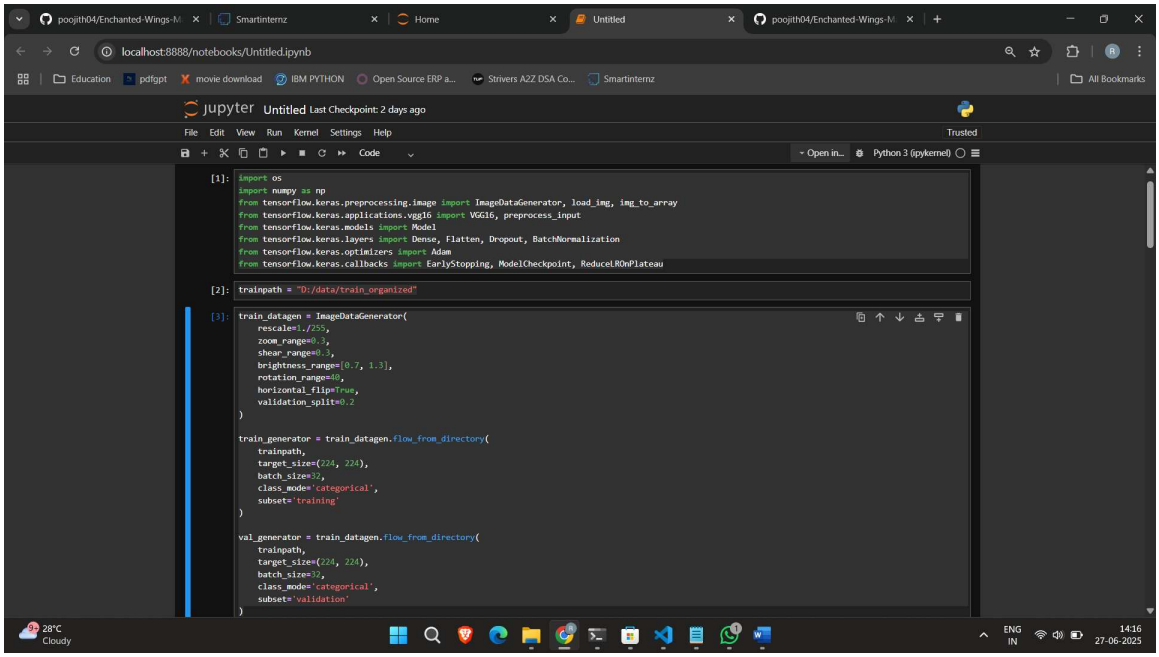
6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

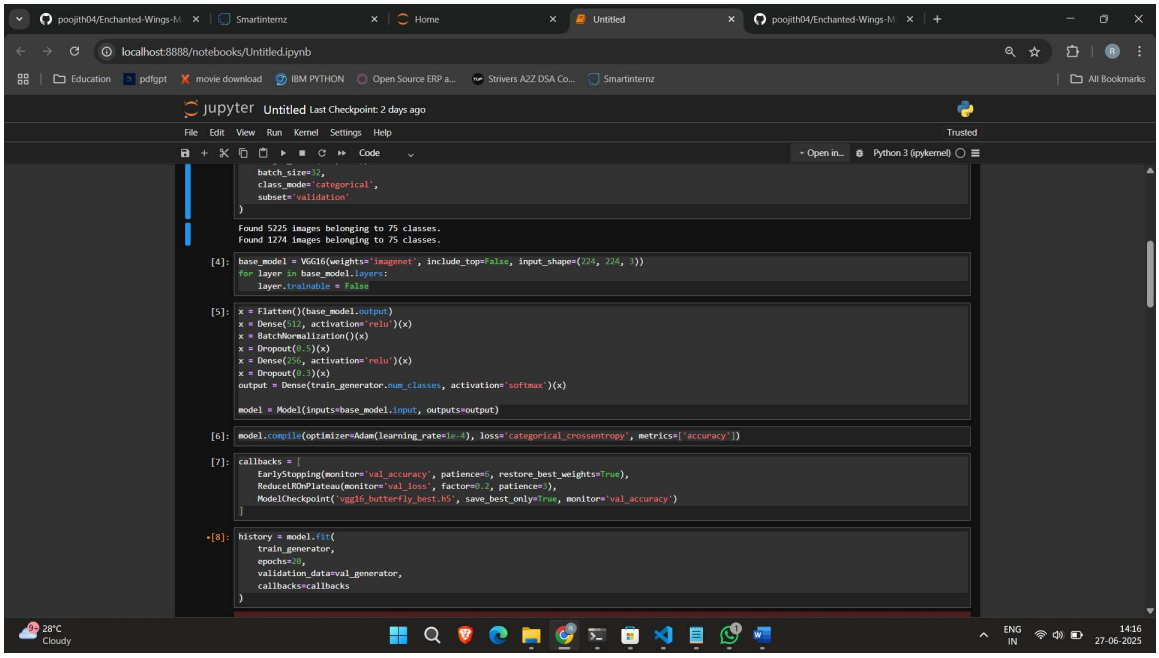
- Model trained on ~5,000 images
- Accuracy: Improved with data augmentation and dropout
- Image tested for classification shows correct label

7. RESULTS

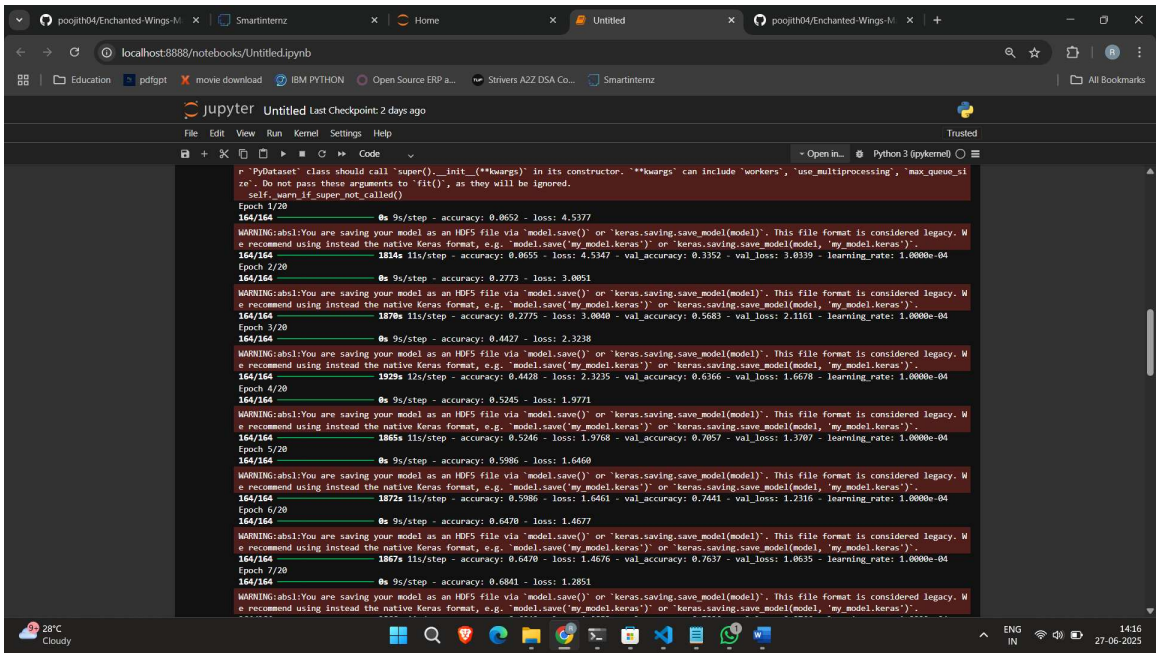
7.1 Output Screenshots:

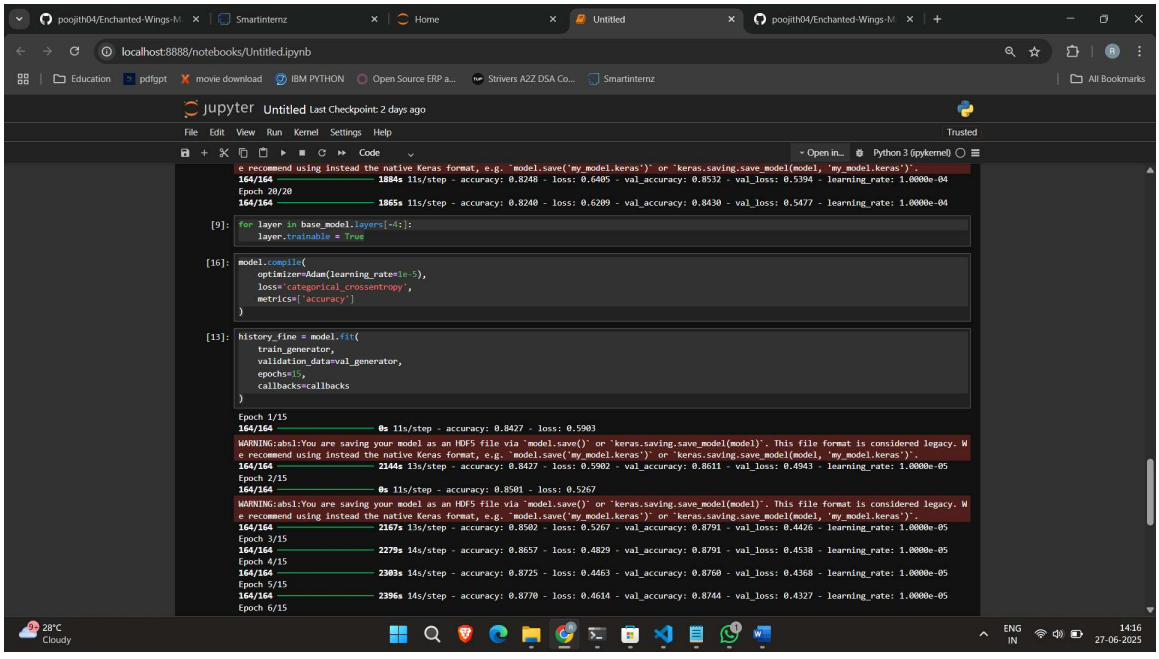


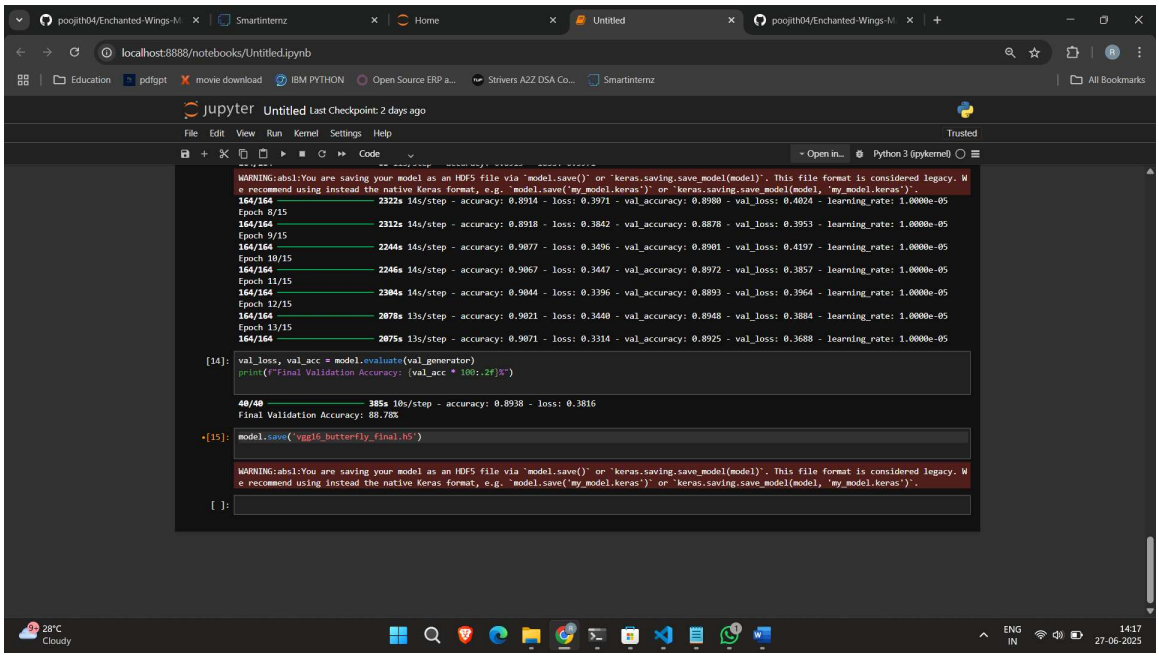
```
[1]: import os
     import numpy as np
     from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
     from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
     from tensorflow.keras.models import Model
     from tensorflow.keras.layers import Dense, Flatten, Dropout, BatchNormalization
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

[2]: trainpath = "D:/data/train_organized"

[3]: train_datagen = ImageDataGenerator(
         rescale=1./255,
         zoom_range=0.3,
         shear_range=0.3,
         brightness_range=[0.7, 1.3],
         rotation_range=40,
         horizontal_flip=True,
         validation_split=0.2
     )

     train_generator = train_datagen.flow_from_directory(
         trainpath,
         target_size=(224, 224),
         batch_size=32,
         class_mode='categorical',
         subset='training'
     )

     val_generator = train_datagen.flow_from_directory(
         trainpath,
         target_size=(224, 224),
         batch_size=32,
         class_mode='categorical',
         subset='validation'
     )
```



```
         batch_size=32,
         class_mode='categorical',
         subset='validation'
     )

Found 5225 images belonging to 75 classes.
Found 1274 images belonging to 75 classes.

[4]: base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
     for layer in base_model.layers:
         layer.trainable = False

[5]: x = Flatten()(base_model.output)
     x = Dense(512, activation='relu')(x)
     x = BatchNormalization()(x)
     x = Dropout(0.5)(x)
     x = Dense(256, activation='relu')(x)
     x = Dropout(0.3)(x)
     output = Dense(train_generator.num_classes, activation='softmax')(x)

     model = Model(inputs=base_model.input, outputs=output)

[6]: model.compile(optimizer=Adam(learning_rate=1e-4), loss='categorical_crossentropy', metrics=['accuracy'])

[7]: callbacks = [
         EarlyStopping(monitor='val_accuracy', patience=5, restore_best_weights=True),
         ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3),
         ModelCheckpoint('vgg16_butterfly_best.h5', save_best_only=True, monitor='val_accuracy')
     ]

*[8]: history = model.fit(
         train_generator,
         epochs=20,
         validation_data=val_generator,
         callbacks=callbacks
     )
```


poojith04/Enchanted-Wings-M
Smartinternz
Home
Untitled
localhost:8888/notebooks/Untitled.ipynb
jupyter Untitled Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)
r `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
self._warn_if_super_not_called()
Epoch 1/20
164/164 0s 9s/step - accuracy: 0.0652 - loss: 4.5377
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
164/164 1814s 11s/step - accuracy: 0.0655 - loss: 4.5347 - val_accuracy: 0.3352 - val_loss: 3.0339 - learning_rate: 1.0000e-04
Epoch 2/20
164/164 0s 9s/step - accuracy: 0.2773 - loss: 3.0051
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
164/164 1870s 11s/step - accuracy: 0.2775 - loss: 3.0040 - val_accuracy: 0.5683 - val_loss: 2.1161 - learning_rate: 1.0000e-04
Epoch 3/20
164/164 0s 9s/step - accuracy: 0.4427 - loss: 2.3238
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
164/164 1929s 12s/step - accuracy: 0.4428 - loss: 2.3235 - val_accuracy: 0.6366 - val_loss: 1.6678 - learning_rate: 1.0000e-04
Epoch 4/20
164/164 0s 9s/step - accuracy: 0.5245 - loss: 1.9771
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
164/164 1865s 11s/step - accuracy: 0.5246 - loss: 1.9768 - val_accuracy: 0.7057 - val_loss: 1.3707 - learning_rate: 1.0000e-04
Epoch 5/20
164/164 0s 9s/step - accuracy: 0.5986 - loss: 1.6460
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
164/164 1872s 11s/step - accuracy: 0.5986 - loss: 1.6461 - val_accuracy: 0.7441 - val_loss: 1.2316 - learning_rate: 1.0000e-04
Epoch 6/20
164/164 0s 9s/step - accuracy: 0.6470 - loss: 1.4677
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
164/164 1867s 11s/step - accuracy: 0.6470 - loss: 1.4676 - val_accuracy: 0.7637 - val_loss: 1.0635 - learning_rate: 1.0000e-04
Epoch 7/20
164/164 0s 9s/step - accuracy: 0.6841 - loss: 1.2851
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
28°C Cloudy
ENG IN
14:16 27-06-2025


poojith04/Enchanted-Wings-M
Smartinternz
Home
Untitled
localhost:8888/notebooks/Untitled.ipynb
jupyter Untitled Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)
e recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
164/164 1884s 11s/step - accuracy: 0.8248 - loss: 0.6405 - val_accuracy: 0.8532 - val_loss: 0.5394 - learning_rate: 1.0000e-04
Epoch 20/20
164/164 1865s 11s/step - accuracy: 0.8240 - loss: 0.6209 - val_accuracy: 0.8430 - val_loss: 0.5477 - learning_rate: 1.0000e-04
[9]:
for layer in base_model.layers[-4:]:
layer.trainable = True
[16]:
model.compile(
optimizer=Adam(learning_rate=1e-5),
loss='categorical_crossentropy',
metrics=['accuracy']
)
[13]:
history_fine = model.fit(
train_generator,
validation_data=val_generator,
epochs=15,
callbacks=callbacks
)
Epoch 1/15
164/164 0s 11s/step - accuracy: 0.8427 - loss: 0.5903
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
164/164 2144s 13s/step - accuracy: 0.8427 - loss: 0.5902 - val_accuracy: 0.8611 - val_loss: 0.4943 - learning_rate: 1.0000e-05
Epoch 2/15
164/164 0s 11s/step - accuracy: 0.8501 - loss: 0.5267
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
164/164 2167s 13s/step - accuracy: 0.8502 - loss: 0.5267 - val_accuracy: 0.8791 - val_loss: 0.4426 - learning_rate: 1.0000e-05
Epoch 3/15
164/164 2279s 14s/step - accuracy: 0.8657 - loss: 0.4829 - val_accuracy: 0.8791 - val_loss: 0.4538 - learning_rate: 1.0000e-05
Epoch 4/15
164/164 2303s 14s/step - accuracy: 0.8725 - loss: 0.4463 - val_accuracy: 0.8760 - val_loss: 0.4368 - learning_rate: 1.0000e-05
Epoch 5/15
164/164 2396s 14s/step - accuracy: 0.8770 - loss: 0.4614 - val_accuracy: 0.8744 - val_loss: 0.4327 - learning_rate: 1.0000e-05
Epoch 6/15
28°C Cloudy
ENG IN
14:16 27-06-2025



8. ADVANTAGES & DISADVANTAGES

Advantages

- Faster training using pre-trained model
- Accurate classification with limited resources
- Supports conservation and research

Disadvantages

- Needs GPU for faster training
- Accuracy depends on image quality and balanced classes

9. CONCLUSION

This project successfully demonstrates the power of transfer learning in building an efficient butterfly classification system. It offers practical applications in biodiversity monitoring, ecological research, and education.

10. FUTURE SCOPE

- Expand model to all 75 species
- Build a mobile app for live butterfly recognition
- Integrate species database for ecological insights
- Improve accuracy with more data and ensemble models

11. APPENDIX

Source Code

```
import os

import numpy as np

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img,
img_to_array

from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, Flatten, Dropout, BatchNormalization

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,
ReduceLROnPlateau

trainpath = "D:/data/train_organized"

train_datagen = ImageDataGenerator(

   rescale=1./255,

   zoom_range=0.3,

   shear_range=0.3,

   brightness_range=[0.7, 1.3],

   rotation_range=40,

   horizontal_flip=True,

   validation_split=0.2

)


train_generator = train_datagen.flow_from_directory(

   trainpath,

   target_size=(224, 224),
```

```
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    trainpath,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
for layer in base_model.layers:
    layer.trainable = False
x = Flatten()(base_model.output)
x = Dense(512, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.3)(x)
output = Dense(train_generator.num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)
model.compile(optimizer=Adam(learning_rate=1e-4), loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
callbacks = [
    EarlyStopping(monitor='val_accuracy', patience=6, restore_best_weights=True),
    ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3),
    ModelCheckpoint('vgg16_butterfly_best.h5', save_best_only=True,
monitor='val_accuracy')
]
history = model.fit(
    train_generator,
    epochs=20,
    validation_data=val_generator,
    callbacks=callbacks
)
for layer in base_model.layers[-4:]:
    layer.trainable = True
model.compile(
    optimizer=Adam(learning_rate=1e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
history_fine = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=15,
    callbacks=callbacks
)
model.save('vgg16_butterfly_final.h5')
```

Dataset Link

https://www.kaggle.com/datasets/phucthaiv02/butterfly-image-classification

GitHub Link

https://github.com/poojith04/Enchanted-Wings-Marvels-of-Butterfly-Species.git