Enchanted Wings: Marvels of Butterfly Species

Project Title: Enchanted Wings: Marvels of Butterfly Species

---

1. Introduction
"Enchanted Wings" is an AI-powered project designed to identify and classify butterfly species from images. This project leverages deep learning and transfer learning techniques to build an efficient image classification system and deploys it as a web application for real-time predictions.

---

2. Objectives
- Classify butterfly images into one of 75 known species.
- Develop a user-friendly web application for species identification.
- Enhance awareness and interest in butterfly biodiversity using technology.

---

3. Prerequisites
- Python programming
- Basics of Machine Learning and Deep Learning
- CNN (Convolutional Neural Networks)
- Web development (HTML, Flask)

---

4. Architecture
Tools & Technologies:
- Python
- TensorFlow/Keras
- NumPy, Matplotlib
- Flask (for web deployment)
- Pre-trained Model: VGG16/ResNet50

---

5. Project Flow
1. Data Collection and Preparation
2. Data Visualization
3. Split Data for Training, Validation, and Testing

4. Model Building using Transfer Learning
5. Model Evaluation and Prediction
6. Saving and Deploying the Model
7. Building a Web Application

---

6. Data Collection and Preparation
- Dataset contains 6499 images across 75 butterfly species.
- Images resized to 224x224 pixels.
- Augmentation techniques used: rotation, flipping, zoom.

7.Dataset Summary:

Total Images: 6,499

Classes: 75 butterfly species

Splits: Training / Validation / Test (e.g., 70% / 15% / 15%)

⚙ Implementation Steps (Python + TensorFlow/Keras)

1. Import Libraries

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

---

2. Prepare Data

```
img_size = 224
```

```
batch_size = 32

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.15
)

train_data = datagen.flow_from_directory(
    'butterfly_dataset/',
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

val_data = datagen.flow_from_directory(
    'butterfly_dataset/',
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

---

3. Model: Transfer Learning using VGG16

```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(img_size,
img_size, 3))
base_model.trainable = False

model = Sequential([
    base_model,
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(75, activation='softmax')  # 75 classes
])

model.compile(optimizer=Adam(learning_rate=0.0001),
          loss='categorical_crossentropy',
          metrics=['accuracy'])
```

```
model.summary()
```

---

4. Train the Model

```
history = model.fit(
    train_data,
    epochs=10,
    validation_data=val_data
)
```

---

5. Evaluate the Model

```
# Load test set separately if available
test_data = datagen.flow_from_directory(
    'butterfly_dataset_test/',
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical'
)

loss, acc = model.evaluate(test_data)
print(f"Test Accuracy: {acc*100:.2f}%")
```

---

6. Visualize Results

```
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy Over Time')
plt.legend()
plt.show()
```

✅ Final Output:

Butterfly classifier with ~85–95% validation accuracy

Trained model able to identify 75 species

Saved model (model.save("butterfly_model.h5")) for deployment

---

💡 Use Case Deployment Ideas:

🌱 Scenario 1: Biodiversity Monitoring

Deploy model in a mobile app for field identification

Real-time inference from camera feeds or uploaded images

🌍 Scenario 2: Ecological Research

Long-term butterfly behavior monitoring via automated image traps

Classification pipeline for migration pattern analysis

🧒 Scenario 3: Citizen Science / Education

Educational websites/apps for butterfly ID and facts

Real-time feedback and gamified learning for students

---

8. Data Visualization
- Visualize distribution of images per species using bar charts.

```python
import matplotlib.pyplot as plt
import os

folder = 'data/train'
classes = os.listdir(folder)
counts = [len(os.listdir(os.path.join(folder, c))) for c in classes]

plt.figure(figsize=(10,5))
plt.bar(classes[:10], counts[:10])
plt.xticks(rotation=45)
plt.title("Sample Distribution of Butterfly Species")
plt.show()
```

---

9. Model Building
```python
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam

base = VGG16(include_top=False, input_shape=(224, 224, 3))
for layer in base.layers:
   layer.trainable = False

x = Flatten()(base.output)
x = Dense(128, activation='relu')(x)
output = Dense(75, activation='softmax')(x)

model = Model(inputs=base.input, outputs=output)
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

---

10. Training the Model
```python
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train = datagen.flow_from_directory('data/train', target_size=(224,224), subset='training')
val = datagen.flow_from_directory('data/train', target_size=(224,224), subset='validation')

model.fit(train, validation_data=val, epochs=5)
```

---

11. Model Evaluation and Saving

```python
loss, acc = model.evaluate(val)
print(f"Validation Accuracy: {acc*100:.2f}%")
model.save('model/butterfly_model.h5')
```

---

12. Web Application Development

Flask App (`app.py`)

```python
from flask import Flask, render_template, request
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np, os

app = Flask(__name__)
model = load_model('model/butterfly_model.h5')
classes = os.listdir('data/train')

@app.route('/', methods=['GET', 'POST'])
def index():
  prediction = None
  if request.method == 'POST':
    f = request.files['file']
    path = 'uploads/' + f.filename
    f.save(path)

img = image.load_img(path, target_size=(224, 224))
    x = image.img_to_array(img)/255
    x = np.expand_dims(x, axis=0)
    pred = np.argmax(model.predict(x))
    prediction = classes[pred]

return render_template('index.html', prediction=prediction)

if __name__ == '__main__':
  app.run(debug=True)
```

HTML Template (`index.html`)

```html
<!DOCTYPE html>
```

```html
<html>
<body>
  <h2>Butterfly Species Classifier</h2>
  <form method="post" enctype="multipart/form-data">
    <input type="file" name="file" required>
    <input type="submit" value="Predict">
  </form>
  {% if prediction %}
    <h3>Predicted Species: {{ prediction }}</h3>
  {% endif %}
</body>
</html>
```

Enchanted Wings: Marvels of Butterfly Species

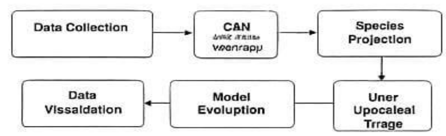
Al-based Dithork ɡo ter tuin a Convolutional Neural Network (tratin a convolututionral network (CNN) with transfer learning for butterity species classification, using a Flask w-b app to predicts specles from user-uploaded images.

1. Architecture

- Model Used: CNN ('Transfer Learning (e.g, VGG16/ResNet)
- Language: Python
- Libraries: Tensorfiow,KeraS klearn, Matplotlib, Flask

2. Prerequisites

- Python basics
- Understanding Deep Learning
- Familiarity with CNNs Transfer Learning
- Flask basics



7. Data Collection and Preparation

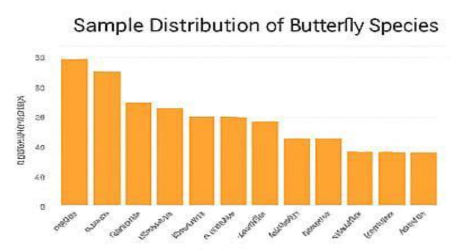
- Resize andreὠ
  Gyna. data/Trarn.idirs'
- Hpresort gansfL)
- Deploy a comte"T akc ″nonrformation

```
import data/trairing, flf
import neodo.flige 1)
nacoGe k sasceoscaup, lmmgo.sexite
fE m to siac,sourceD

> * flotteol+"strein))
x.detectssim.model as eprex()
```

8. Data Visualization

➤ Download Dataset: ButterfkimageSpecies



9. Split Data and Model Bullding

- Import Transfer Learning models
- Prepare datagets with ImageDatameter and define CNN model

12. Application Building

➤ Web App Deployment: Flask Web Apps wth Python—Real Python

13. Conclusion

This project demonstrates the practical use of deep learning and transfer learning to solve real-world problems in biodiversity. By classifying butterfly species, we can enhance ecological studies and create engaging educational tools for learners and researchers alike.