Low Level Design Document

Introduction

This Low Level Design (LLD) document details the implementation plan for **EcoClassify - Wildlife Image Classifier**. EcoClassify is a Streamlit-based application that leverages transfer learning (ResNet) to classify wildlife images into multiple species, featuring data augmentation, model fine-tuning, and an explanation dashboard for predictions.

---

1. System Components

Component	Description	Key Responsibilities
UI Layer (Streamlit)	Web interface for user interaction	Image upload, display results, dashboard
Data Pipeline	Handles image preprocessing and augmentation	Load, transform, augment images
Model Handler	Manages model loading, inference, and fine-tuning	Load ResNet, predict, train, save/load
Explanation Engine	Generates prediction explanations	Saliency maps, class probabilities
Data Storage	Stores models, logs, and metadata	Save/load model weights, logs

---

2. Class/Interface Overview

Class/Module	Key Methods/Attributes	Relationships
`ImageProcessor`	`preprocess()`, `augment()`, `load_image()`	Used by ModelHandler
`ModelHandler`	`load_model()`, `predict()`, `fine_tune()`, `save_model()`	Uses ImageProcessor, DataStorage
`ExplanationEngine`	`generate_saliency()`, `get_probabilities()`	Uses ModelHandler
`DataStorage`	`save()`, `load()`, `log()`	Used by ModelHandler
`AppUI`	`main()`, `display_results()`, `upload_image()`	Uses all above modules

Key Methods Example:

```
class ModelHandler:
    def load_model(self, path): ...
    def predict(self, image): ...
```

```
    def fine_tune(self, data_loader): ...
    def save_model(self, path): ...
```

---

3. Data Structure Overview

Data Model	Fields/Schema	Purpose
Image Input	`np.ndarray` (H x W x C), file path	Raw/processed image data
Prediction Output	`{ 'species': str, 'probability': float }`	Model prediction result
Explanation Output	`{ 'saliency_map': np.ndarray, 'top_classes': list }`	Explanation dashboard
Model Metadata	`{ 'model_name': str, 'accuracy': float, ... }`	Model info for dashboard/logs

---

4. Algorithms/Logic

Image Classification Flow:

```
def classify_image(image_path):
    image = ImageProcessor.preprocess(image_path)
    pred = ModelHandler.predict(image)
    explanation = ExplanationEngine.generate_saliency(image)
    return pred, explanation
```

Model Fine-Tuning:

- Load pre-trained ResNet
- Replace final layer with species classifier
- Apply data augmentation
- Train on labeled dataset
- Save best model

---

5. Error Handling

Scenario	Handling Approach
Invalid image upload	Show error message, request re-upload
Model load failure	Log error, display fallback message
Prediction failure	Return error response, log details
Explanation generation failure	Warn user, skip explanation, log error
Data storage/read error	Retry, log, notify user if persistent

End of Document