Product Requirements & Specification Document

Project Name

EcoClassify - Wildlife Image Classifier

Description

EcoClassify is a Streamlit-based application leveraging transfer learning (ResNet) to classify wildlife images into multiple species. The app supports data augmentation, model fine-tuning, and provides an explanation dashboard for model predictions. Target users include researchers and educators.

---

1. Goals & Objectives

Goal	Description
Accurate Wildlife Classification	Classify uploaded images into predefined wildlife species
Model Explainability	Provide visual explanations for predictions
User-Friendly Interface	Simple, intuitive Streamlit UI for non-technical users
Research & Education Utility	Support learning and research in wildlife identification

---

2. Core Features

Feature	Description
Image Upload	Users upload single/multiple wildlife images
Image Preprocessing	Resize, normalize, and augment images using OpenCV and torchvision
Transfer Learning	Use pre-trained ResNet (PyTorch) with fine-tuning on wildlife dataset
Data Augmentation	Apply random flips, rotations, color jitter, etc.
Model Fine-Tuning	Allow users to trigger fine-tuning on custom datasets
Prediction Output	Display top-3 predicted species with confidence scores
Explanation Dashboard	Visualize prediction explanations (e.g., Grad-CAM heatmaps)
Download Results	Export predictions and explanations as CSV

---

3. User Stories

As a…	I want to…	So that…
Researcher	Upload and classify wildlife images	I can analyze species distribution
Educator	View model explanations for predictions	I can teach students about model behavior

User	Fine-tune the model with my own dataset	I can improve accuracy for my use case

4. Functional Requirements

ID	Requirement
FR1	The app shall allow users to upload images (JPG, PNG) via the Streamlit interface
FR2	The app shall preprocess and augment images before inference
FR3	The app shall use a pre-trained ResNet model, fine-tuned on a wildlife dataset
FR4	The app shall display top-3 predicted species and confidence scores for each image
FR5	The app shall provide visual explanations (e.g., Grad-CAM) for each prediction
FR6	The app shall allow users to fine-tune the model with their own labeled image datasets
FR7	The app shall allow users to download prediction results and explanations as CSV

5. Non-Functional Requirements

ID	Requirement
NFR1	The app shall respond to image uploads within 5s
NFR2	The app shall support batch processing (≤10 images)
NFR3	The app shall run on standard CPU hardware
NFR4	The app shall be compatible with Python 3.8+

6. Technical Specifications

Component	Technology / Library	Notes
Frontend	Streamlit	UI, file upload, results display
Model	PyTorch, torchvision	ResNet (e.g., ResNet-50), transfer learning
Data Augmentation	torchvision, OpenCV	Random transforms, normalization
Explanations	torchcam, matplotlib	Grad-CAM or similar methods
Data Handling	pandas, numpy	CSV export, data manipulation

7. Data Flow Overview

```
graph TD
A[User Uploads Image] --> B[Preprocessing & Augmentation]
B --> C[ResNet Inference]
C --> D[Prediction Output]
```

```
C --> E[Explanation Generation]
D & E --> F[Results Display & Download]
```

---

8. UI/UX Requirements

- Simple, single-page Streamlit layout
- Image upload widget (multi-file support)
- Display of uploaded images, predictions, and explanations
- Button to trigger model fine-tuning (with dataset upload)
- Download button for results

---

9. Acceptance Criteria

ID	Criteria
AC1	Users can upload and classify up to 10 images at once
AC2	Top-3 predictions and confidence scores are shown for each image
AC3	Visual explanation (heatmap) is displayed for each prediction
AC4	Users can fine-tune the model with a custom dataset and see improved results
AC5	Results and explanations can be downloaded as CSV

---

10. Out of Scope

- Real-time video classification
- Mobile app version
- Unsupervised or semi-supervised learning

---

11. Milestones

Milestone	Description	Target Date
Prototype UI	Basic Streamlit interface	Week 1
Model Integration	ResNet inference and augmentation	Week 2
Explanation Dashboard	Grad-CAM integration	Week 3
Fine-Tuning Feature	Custom dataset training	Week 4
Testing & Documentation	QA, user guide, deployment	Week 5

---

12. Appendix

Sample Species List:

- Lion, Elephant, Zebra, Giraffe, Leopard, Buffalo, Rhino, Hyena, Cheetah, Wildebeest

Sample Model Loading Pseudocode:

```
import torch
from torchvision import models

model = models.resnet50(pretrained=True)
# Replace final layer for N species
model.fc = torch.nn.Linear(model.fc.in_features, N_SPECIES)
```

End of Document