Final Project Report\ Project Title: Waste Classification Using Transfer Learning

---

1. INTRODUCTION

1.1 Project Overview

This project aims to automate waste classification using transfer learning. Proper waste management starts with correct segregation, and our solution classifies waste images into Biodegradable, Recyclable, and Trash. A Flask-based web application enables real-time classification and eco-friendly guidance.

1.2 Purpose

- Develop a smart system to categorize waste efficiently.
- Use transfer learning (VGG16) to achieve high accuracy.
- Offer educational and disposal suggestions.

---

2. IDEATION PHASE

2.1 Problem Statement

Unsegregated waste leads to improper disposal, increasing pollution and recycling inefficiency. Manual sorting is not scalable. There is a need for an accessible, image-based waste classification solution.

2.2 Empathy Map Canvas

End-users face confusion about where to discard waste. The app provides immediate feedback and classification assistance.

2.3 Brainstorming

- Use of pre-trained models to reduce training cost.
- Real-time classification with confidence levels.
- Responsive UI with upload preview and recycling tips.

---

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. User lands on home page.
2. Clicks 'Get Started' → Predict page.
3. Uploads waste image → Receives prediction and instructions.
4. Optionally uploads another image.

3.2 Solution Requirements

- Image upload interface.
- Backend model serving.
- Feedback on confidence and tips.

3.3 Data Flow Diagram

User → Upload Image → Preprocess → Predict → Display Results

3.4 Technology Stack

- Python, Flask, HTML, CSS, TailwindCSS
- TensorFlow, Keras, OpenCV
- Anaconda, Jupyter Notebook

4. PROJECT DESIGN

4.1 Problem-Solution Fit

Transfer learning offers a fast, efficient way to adapt pre-trained vision models for new tasks. It reduces training time and improves accuracy with limited data.

4.2 Proposed Solution

1. Preprocess uploaded image.
2. Use VGG16-based model to predict category.
3. Display result with confidence and recycling instructions.

4.3 Solution Architecture

Frontend (HTML/CSS) → Flask Server → TensorFlow Model → Output + Tips

5. PROJECT PLANNING

- Week 1: Dataset preparation and research
- Week 2: Model training with VGG16
- Week 3: Flask backend and API integration
- Week 4: Frontend design and deployment

6. FUNCTIONAL AND PERFORMANCE TESTING

- Functional tests on image upload and prediction flow.
- Edge case tests with unsupported or blank images.

- Performance: average response time < 2s.

7. RESULTS

- Accuracy: \~90% on validation set
- High prediction confidence on clean inputs
- Real-time UI with drag-and-drop upload

7.1 Output Screenshots

(Attached separately in demo folder or appendix)

8. ADVANTAGES & DISADVANTAGES

Advantages:

- Reduces manual waste segregation effort
- Quick setup via transfer learning
- Easily deployable on low-resource systems

Disadvantages:

- May misclassify overlapping objects
- Performance may degrade with poor lighting

9. CONCLUSION

This project demonstrates how transfer learning can solve real-world environmental problems. The system promotes better waste management and encourages awareness about proper disposal methods.

10. FUTURE SCOPE

- Add more waste classes: hazardous, e-waste, metal
- Mobile app integration
- Real-time webcam prediction
- Cloud deployment with user analytics

11. APPENDIX

Folder Structure:

```
W_FLASK/
├── app.py
├── vgg16.h5
├── data/
│   ├── raw/
│   │   ├── Biodegradable Images/
│   │   ├── Recyclable Images/
│   │   └── Trash Images/
│   ├── output_dataset/
│   │   ├── train/
│   │   ├── val/
│   │   └── test/
├── static/
│   ├── assets/
│   ├── forms/
│   └── uploads/
├── templates/
│   ├── index.html
│   ├── predict.html
│   ├── portfolio.html
│   ├── contact.html
│   └── blog.html
├── notebooks/
│   ├── train_model.ipynb
│   └── test_model.ipynb
```

Run Instructions:

```
conda activate waste_classification
cd waste_classification_project/w_flask
python app.py
```

API Endpoints:

- `/predict` → Web upload route
- `/api/predict` → JSON-based API for file input

Dataset Source: Kaggle (linked externally) **Model File:** `vgg16.h5` (trained and saved locally)