Artificial Intelligence and Machine Learning

Project Title: CleanTech- Transforming Waste Management with Transfer Learning

Introduction:

In today's rapidly urbanizing world, effective waste management has become a critical environmental and public health concern. One of the key challenges lies in the accurate segregation of waste into biodegradable and non-biodegradable categories — a process that is often manual, inefficient, and prone to error.

CleanTech – Transforming Waste Management with Transfer Learning is an innovative project that leverages the power of deep learning and transfer learning to automate the classification of waste. By utilizing pre-trained convolutional neural networks (CNNs), this project significantly reduces the time and resources required to build an intelligent waste classification model from scratch.

The system is integrated into a user-friendly web application where users can upload images of waste materials and instantly receive predictions regarding their category. This not only streamlines waste segregation but also supports smarter and more sustainable waste disposal practices.

Through CleanTech, we aim to bridge the gap between AI technology and environmental sustainability, contributing to smarter cities and a cleaner planet.

Team Members:4

S.NO	Team Member Names	Roles
1.	Rangana Baby Pavani Durga	Model Developer
2.	Matta Venkata Laxmi Jayanthi	Frontend Developer
3.	Mithina Santhosh Kumar	Backend Developer
4.	K.Mounika	Documentation

Project Overview:

Purpose:

The purpose of the CleanTech – Transforming Waste Management with Transfer Learning project is to develop an intelligent system that automates the classification of municipal waste into biodegradable and non-biodegradable categories using machine learning. By leveraging transfer learning, the project aims to reduce the time and computational resources needed for training a high-accuracy image classification model.

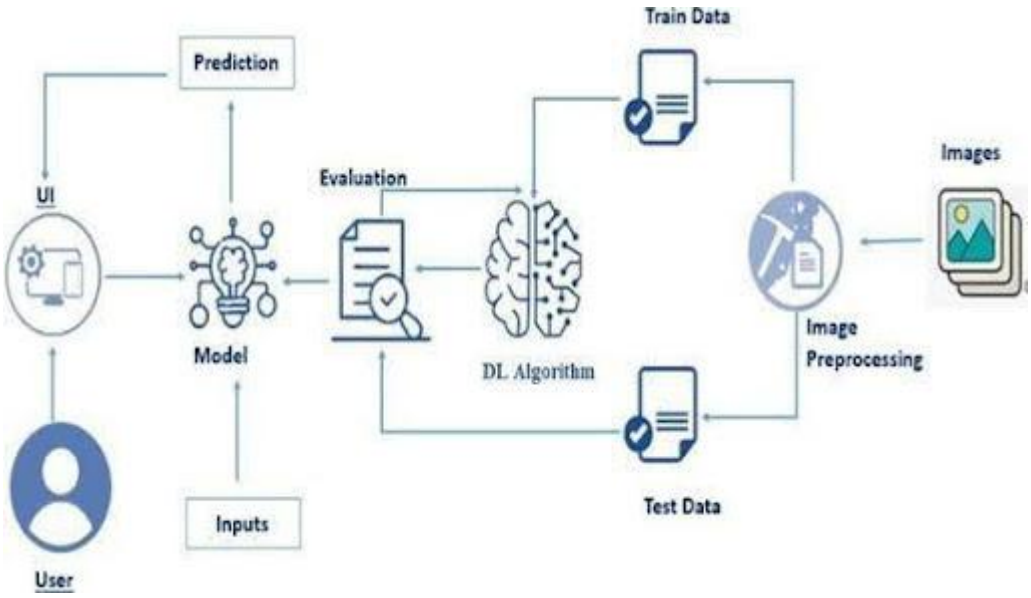
The key goals of the project are:

- To build an efficient deep learning model using transfer learning techniques.
- To create a user-friendly web application for real-time waste classification.
- To contribute toward smart, sustainable, and eco-friendly waste management practices.
- To raise awareness and encourage proper waste segregation through technology.

Key Features:

- 🧠 **Transfer Learning-Based Waste Classifier**
  Utilizes a pre-trained deep learning model (e.g., vgg16) fine-tuned for classifying waste images into biodegradable and non-biodegradable categories.
- 📤 **Image Upload Functionality**
  Users can easily upload waste images through the web interface for real-time classification.
- ⚡ **Fast & Accurate Predictions**
  Delivers quick and reliable classification results powered by an optimized deep learning pipeline.
- 🌐 **Interactive Web Application**
  Clean and responsive UI built using HTML/CSS integrated with Flask for backend processing.
- 📊 **Prediction Result Display**
  Displays clear output with predicted class and confidence score to inform the user.
- 🗂️ **Organized Dataset Management**
  Includes preprocessed, labeled datasets with appropriate train/test splits for effective model training.

Architecture:



The architecture diagram illustrates the end-to-end workflow of the CleanTech: Waste Classification System using Transfer Learning. Here's a breakdown of each component:

1. **User Interface (UI):**
    - The entry point where users interact with the system.
    - Users upload waste images through a simple web interface.
2. **Inputs:**
    - The images submitted by users serve as inputs for classification.
3. **Model:**
    - A pre-trained Deep Learning (DL) model (via Transfer Learning) processes the inputs.
    - It generates predictions to classify waste as either biodegradable or non-biodegradable.
4. **Prediction:**
    - The output of the model is a class label and confidence score indicating the waste category.
5. **Evaluation:**
    - The model's performance is assessed using standard metrics (e.g., accuracy, precision, recall).
    - This step ensures the model's effectiveness before deployment

6. **DL Algorithm:**
    - The core learning engine that has been trained using transfer learning.
    - It takes in training and testing data for model learning and validation.
7. **Image Preprocessing:**
    - Input images are resized, normalized, and augmented to enhance model performance.
    - Cleaned and formatted data is then split into training and test sets.
8. **Train Data & Test Data:**
    - Preprocessed images are divided into training data (for learning) and test data (for evaluation).
9. **Images:**
    - The raw image dataset forms the foundational input for the entire model development process.

Setup Instructions:

Prerequisites:

🔧 Technical Skills

- **Python Programming:** Understanding of Python for scripting and model development.
- **Deep Learning Basics:** Knowledge of CNNs and how transfer learning works.
- **Flask Framework:** For building the backend of the web application.
- **HTML/CSS:** For designing the frontend UI of the web app**.**

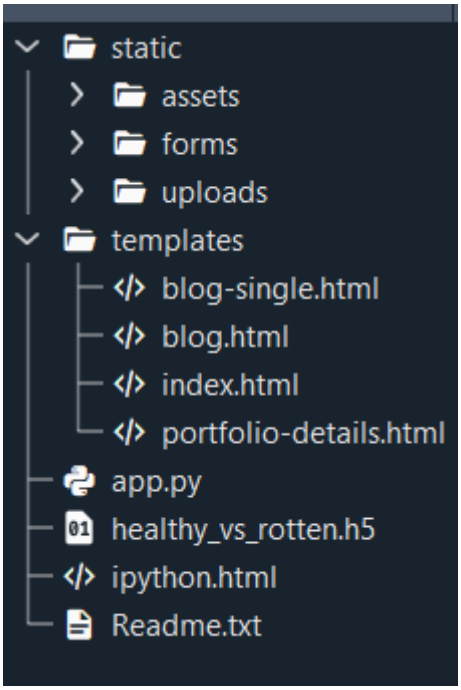
📦 Software & Libraries

- Python (3.6 or above)
- Jupyter Notebook or any Python IDE (e.g., VSCode, PyCharm)
- Required Python Libraries:
    - **tensorflow or keras –** for deep learning and transfer learning
    - **numpy, pandas –** for data handling
    - **matplotlib, seaborn –** for data visualization
    - **flask –** to build the web app
    - **Pillow, opencv-python –** for image preprocessing
    - **scikit-learn –** for evaluation metrics

Installations:

1. **Install Python**
2. **Create a Virtual Environment**
3. **Install Required Libraries**
    - pip install tensorflow
    - pip install keras
    - pip install numpy
    - pip install pandas
    - pip install matplotlib
    - pip install seaborn
    - pip install opencv-python
    - pip install scikit-learn
    - pip install flask
    - pip install pillow
4. **Install Flask**
    - pip install flask

Folder Structure:



📁 static/

Contains static assets such as images, stylesheets, and other frontend resources.

- **assets/** – Holds CSS, JavaScript, or other design-related assets.
- **forms/** – Used for form-related resources if applicable.
- **uploads/** – Stores user-uploaded images for classification.

📁 templates/

Houses all HTML files used by Flask for rendering dynamic web pages.

- **index.html** – The main landing page of the web application.
- **blog.html** – A blog overview page for articles or project content.
- **blog-single.html** – Displays detailed view of a single blog post.
- **portfolio-details.html** – Showcases project details or predictions.

📄 app.py

The main Flask application file.
It includes all backend logic such as routing, model loading, image processing, and prediction handling.

---

healthy_vs_rotten.h5

The trained deep learning model file using **Transfer Learning**, responsible for classifying waste images into categories.

---

ipython.html

Used for rendering output from Jupyter or IPython environments (optional, could be an analysis summary).

Readme.txt

A text file providing an overview of the project, instructions, and setup guidance (may include goals, features, and usage).

Running the application:

Step 1: Set Up a Virtual Environment

- python -m venv venv
- source venv/Scripts/activate

Step 2: Install Dependencies

- pip install -r requirements.txt

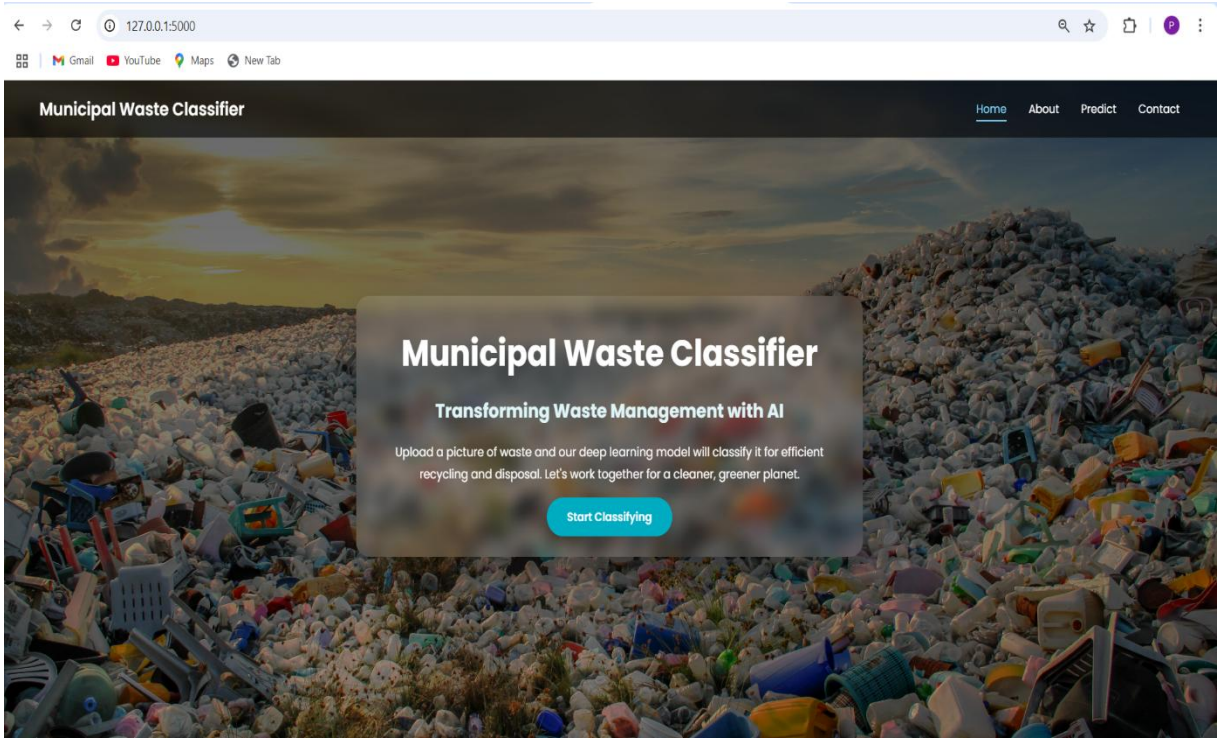
Step 3: Verify Model File

Step 4: Run the Flask Application

- python app.py

Step 5: Access the Web Application

- http://127.0.0.1:5000/



API Documentation

The backend exposes several endpoints to handle the image upload, prediction, and rendering of pages in the web application. Below is a detailed description of each endpoint, including request methods, expected parameters, and sample responses.

GET /

- Description: Loads the homepage of the web application.
- Method: GET
- Request Params: None
- Response: Renders index.html
- Example: GET http://localhost:5000/

POST /predict

- Description: Accepts a user-uploaded image, processes it, and returns the predicted waste category.
- Method: POST
- Request Params:
    - file (form-data): The image file uploaded by the user (e.g., .jpg, .png)
- Response: Renders portfolio-details.html with the prediction result.
- Example Request (using Postman or HTML form):POST http://localhost:5000/predict

Form Data:

file = sample_waste_image.jpg

- Example Response: Prediction: Non-Biodegradable Waste (Confidence: 92.67%)

Internally, the image is passed to the loaded model (healthy_vs_rotten.h5) for classification.

GET /blog

- Description: Loads the blog listing page.
- Method: GET
- Response: Renders blog.html
- Example:GET http://localhost:5000/blog

GET /blog-single

- Description: Loads a single blog article page.
- Method: GET
- Response: Renders blog-single.html
- Example:GET http://localhost:5000/blog-single

GET /portfolio-details

- Description: Displays the prediction result page (called after /predict).
- Method: GET
- Response: Renders portfolio-details.html
- Example:GET http://localhost:5000/portfolio-details

User Interface

The CleanTech: Waste Classification Web Application provides a simple and intuitive UI that enables users to easily interact with the waste classification system.

1.Home Page (index.html)

2. Image Upload Section

3.Prediction page

Demo Link: Just Click on it

Municipal_waste_classifier_demo.mp4

Future Enhancements

As the current version of CleanTech is a prototype for waste image classification, there are several ways to expand and improve the system in the future:

---

1. Multi-Class Waste Classification

- Extend the model to classify **more than two categories**, such as:
    - Biodegradable
    - Non-biodegradable
    - Recyclable
    - Hazardous
- Improves the system's usefulness in real-world waste management.

---

2. Authentication & User Management

- Implement **login/signup** using Flask-Login or JWT.
- Allow users to track their past predictions or maintain waste logs.

---

3. Mobile-Friendly UI / PWA Support

- Make the app fully responsive and mobile-optimized.
- Convert it into a **Progressive Web App (PWA)** for offline and installable usage.

---

4. Real-Time Camera Integration

- Enable **real-time classification** using a webcam or phone camera feed.
- Useful for live waste sorting setups.

Links:

Github Link:

https://github.com/PAVANIRANGANA/Municipal_waste_classification

Demo Video Link:

https://drive.google.com/file/d/1mlbgCVl_DWnBuqn8M7kIv5pWr8kNHPUF/view?usp=drive_link