CleanTech: Transforming Waste Management With Transfer Learning

# 🎯 Aim

To build a smart waste classification system using transfer learning that classifies waste into categories (organic, recyclable, hazardous, etc.) to aid automated waste management.

# 📝 Problem Statement

Manual waste segregation is inefficient and prone to errors. Using computer vision and transfer learning, we aim to develop an AI model that automatically classifies waste images into appropriate categories, improving waste disposal and recycling processes.

# 🧱 Architecture

1. Pretrained CNN (e.g., MobileNetV2, ResNet50) as feature extractor  
2. Custom classifier head on top  
3. Training on labeled waste image dataset  
4. Deployment in a web/mobile application

# ✅ Prerequisites

- Python  
- TensorFlow/Keras  
- NumPy, Pandas, Matplotlib  
- Pretrained models (MobileNetV2, ResNet50)  
- Dataset of waste images (e.g., TrashNet or custom)

# 🗂️ Project Structure

CleanTech/  
│  
├── data/ # Dataset of waste images  
├── model/ # Trained model saved here  
├── app/ # Flask or Streamlit app  
├── notebooks/ # Jupyter notebooks for model training  
├── utils/ # Helper functions  
└── main.py # Entry point

# 📥 Data Collection and Preparation

Use TrashNet dataset or similar.  
  
Categories:  
- Cardboard  
- Glass  
- Metal  
- Paper  
- Plastic  
- Trash

Code:

import os  
import tensorflow as tf  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
  
train\_datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)  
  
train\_data = train\_datagen.flow\_from\_directory(  
 'data/',  
 target\_size=(224, 224),  
 batch\_size=32,  
 class\_mode='categorical',  
 subset='training'  
)  
  
val\_data = train\_datagen.flow\_from\_directory(  
 'data/',  
 target\_size=(224, 224),  
 batch\_size=32,  
 class\_mode='categorical',  
 subset='validation'  
)

# 🧠 Split Data & Model Building

Code:

from tensorflow.keras.applications import MobileNetV2  
from tensorflow.keras.models import Model  
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D  
  
base\_model = MobileNetV2(input\_shape=(224, 224, 3), include\_top=False, weights='imagenet')  
base\_model.trainable = False  
  
x = base\_model.output  
x = GlobalAveragePooling2D()(x)  
x = Dense(128, activation='relu')(x)  
output = Dense(6, activation='softmax')(x)  
  
model = Model(inputs=base\_model.input, outputs=output)  
model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# 🧪 Testing Model & Data Prediction

Code:

history = model.fit(train\_data, validation\_data=val\_data, epochs=5)  
  
# Save the model  
model.save('model/waste\_classifier.h5')

# 🌐 Application Building (Flask)

Flask App Code (app/app.py):

from flask import Flask, request, render\_template  
from tensorflow.keras.models import load\_model  
from tensorflow.keras.preprocessing import image  
import numpy as np  
  
app = Flask(\_\_name\_\_)  
model = load\_model('model/waste\_classifier.h5')  
class\_names = ['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']  
  
@app.route('/', methods=['GET', 'POST'])  
def index():  
 if request.method == 'POST':  
 img = image.load\_img(request.files['file'], target\_size=(224, 224))  
 img\_array = image.img\_to\_array(img) / 255.0  
 prediction = model.predict(np.expand\_dims(img\_array, axis=0))  
 label = class\_names[np.argmax(prediction)]  
 return f"Predicted Category: {label}"  
 return '''  
 <form method="POST" enctype="multipart/form-data">  
 Upload Waste Image: <input type="file" name="file"><br><br>  
 <input type="submit">  
 </form>  
 '''  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

# 📊 Output

- A trained model classifies uploaded waste images into correct categories.  
- Application predicts class and displays it to the user.