Project Name:

To develop a CNN Model To Classify Images of Plastic Waste into Different Categories

Learning Objectives

-Understand CNNs for image classification
-Apply deep learning to waste segregation
- Improve waste management efficiency

Source : www.freepik.com/

Tools and Technology used

- Python
- TensorFlow/Keras
- OpenCV
- NumPy & Pandas
- MatplotlibCNN Architecture:- Input Layer: Image (224x224)- 3 Convolutional Layers with ReLU Activation- Max Pooling after each Conv Layer- Fully Connected Dense Layer- Output Layer with Softmax Activation

Methodology

- 1.Data Collection
  2. Preprocessing (Resizing, Normalization)
  3. Model Training using CNN
  4. Evaluation & Accuracy Calculation
  5. Deployment & Real-World ApplicationTraining Details:- Optimizer: Adam- Loss Function: Categorical Crossentropy- Batch Size: 32- Epochs: 25- Data Augmentation Applied

Problem Statement:

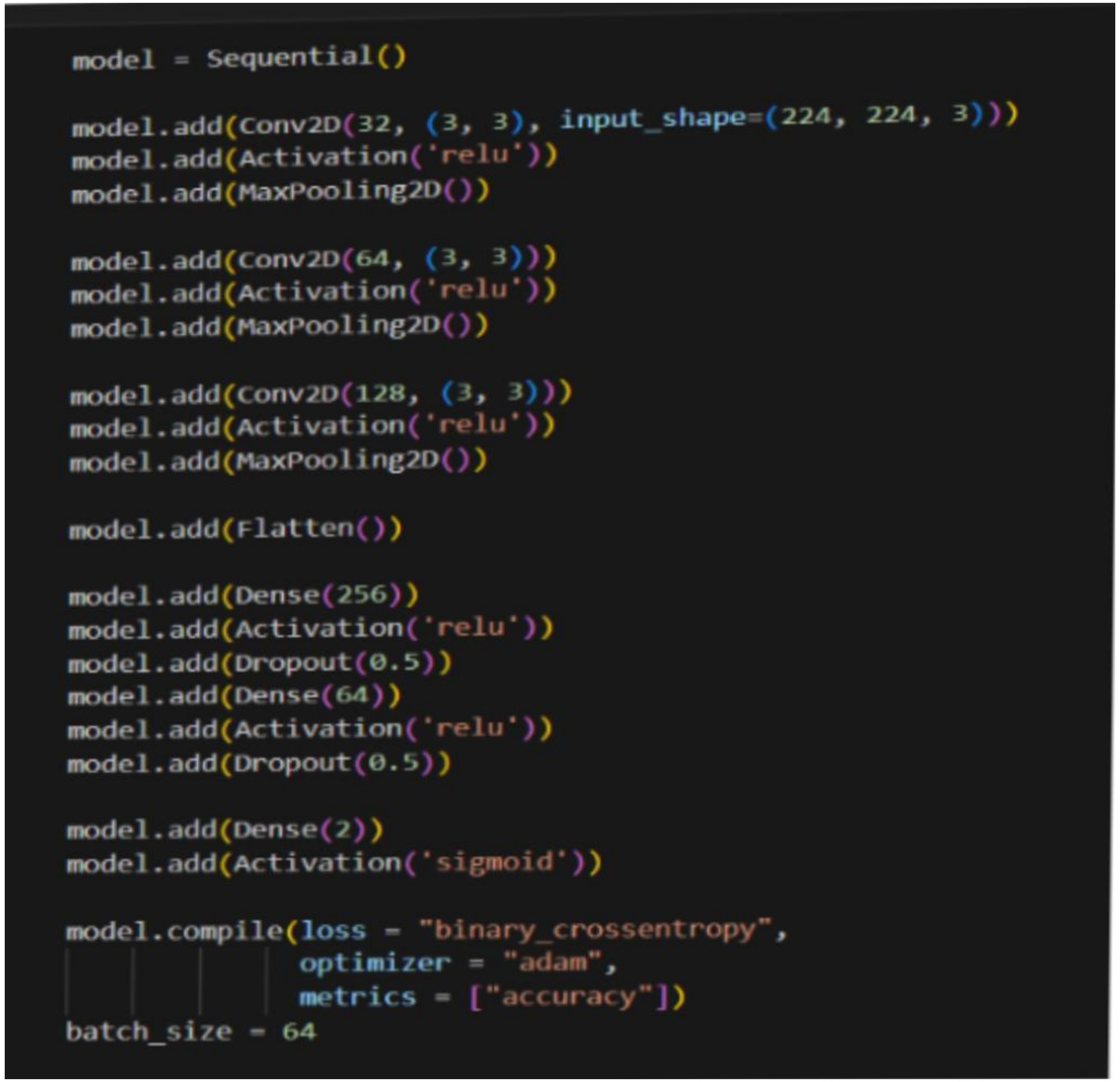
Manual waste classification is inefficient and error-prone. Automating the process using deep learning can significantly improve recycling efforts

.Why is this important?
- Inefficient waste management leads to pollution-
Manual sorting is time-consuming and error-prone-
AI-based automation can improve efficiency and sustainability

Solution:

"A Convolutional Neural Network (CNN)-based model is developed to classify waste into different categories automatically. Using deep learning techniques, this model processes images of waste and predicts the appropriate category. The system can be integrated into smart waste management solutions to improve recycling efficiency and reduce environmental impact."

Algorithm

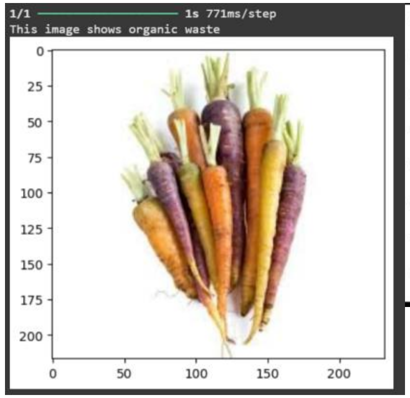


```
model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape=(224, 224, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D())

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D())

model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D())

model.add(Flatten())

model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(2))
model.add(Activation('sigmoid'))

model.compile(loss = "binary_crossentropy",
              optimizer = "adam",
              metrics = ["accuracy"])
batch_size = 64
```

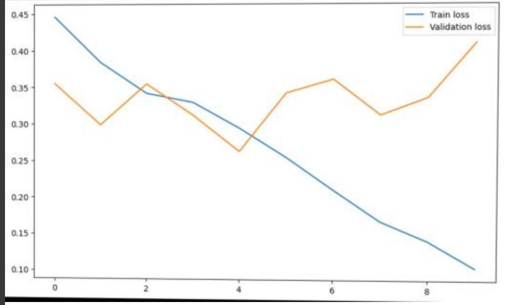
Github link:

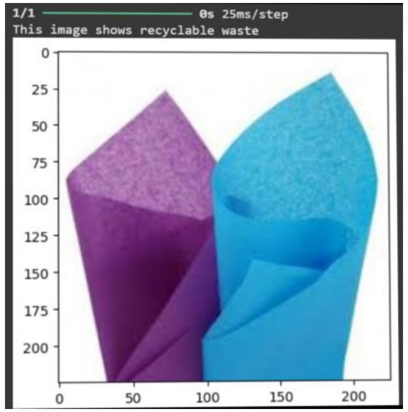
https://github.com/Srija9059/WasteClassification-Using-CNN/

Screenshot of Output:

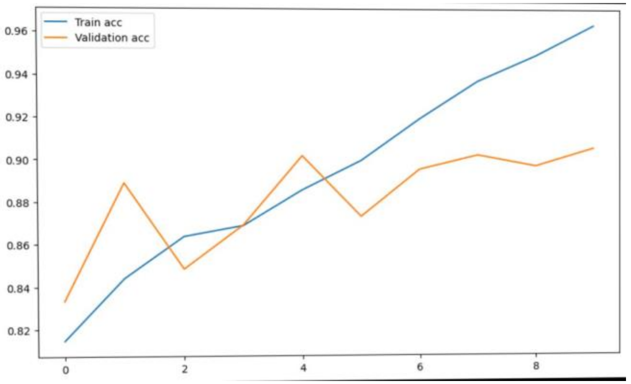


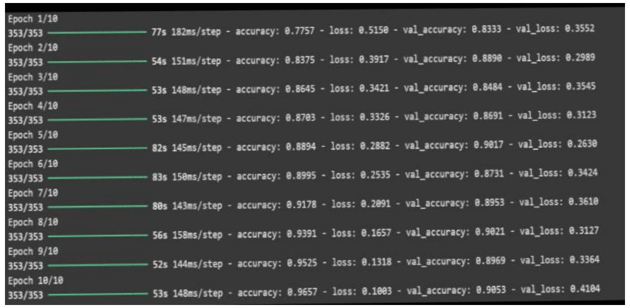
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 771ms/step
This image shows organic waste
```



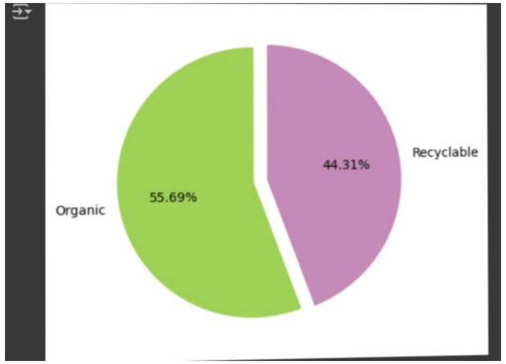


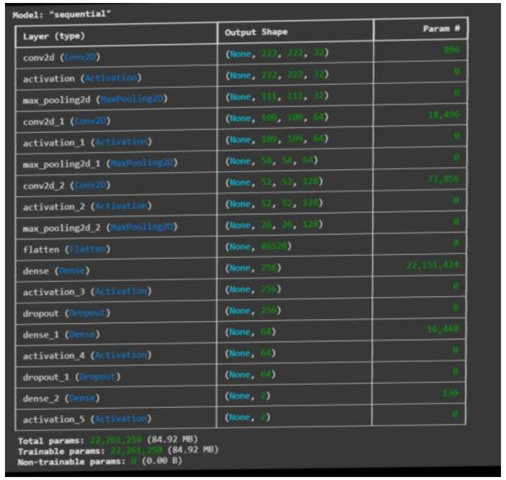
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
This image shows recyclable waste
```



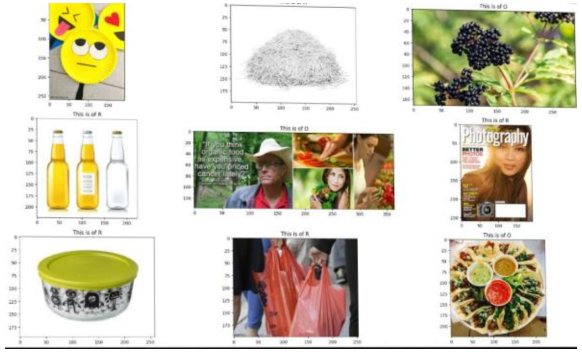


```
Epoch 1/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 77s 182ms/step - accuracy: 0.7757 - loss: 0.5150 - val_accuracy: 0.8333 - val_loss: 0.3552
Epoch 2/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 54s 151ms/step - accuracy: 0.8375 - loss: 0.3917 - val_accuracy: 0.8890 - val_loss: 0.2989
Epoch 3/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 53s 148ms/step - accuracy: 0.8645 - loss: 0.3421 - val_accuracy: 0.8484 - val_loss: 0.3545
Epoch 4/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 53s 147ms/step - accuracy: 0.8703 - loss: 0.3326 - val_accuracy: 0.8691 - val_loss: 0.3123
Epoch 5/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 82s 145ms/step - accuracy: 0.8894 - loss: 0.2882 - val_accuracy: 0.9017 - val_loss: 0.2630
Epoch 6/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 83s 150ms/step - accuracy: 0.8995 - loss: 0.2535 - val_accuracy: 0.8731 - val_loss: 0.3424
Epoch 7/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 80s 143ms/step - accuracy: 0.9178 - loss: 0.2091 - val_accuracy: 0.8953 - val_loss: 0.3610
Epoch 8/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 56s 158ms/step - accuracy: 0.9391 - loss: 0.1657 - val_accuracy: 0.9021 - val_loss: 0.3127
Epoch 9/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 52s 144ms/step - accuracy: 0.9525 - loss: 0.1318 - val_accuracy: 0.8969 - val_loss: 0.3364
Epoch 10/10
353/353 ━━━━━━━━━━━━━━━━━━━━ 53s 148ms/step - accuracy: 0.9657 - loss: 0.1003 - val_accuracy: 0.9053 - val_loss: 0.4104
```





```
Model: "sequential"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d (Conv2D)                 │ (None, 222, 222, 32)   │           896 │
│ activation (Activation)         │ (None, 222, 222, 32)   │             0 │
│ max_pooling2d (MaxPooling2D)    │ (None, 111, 111, 32)   │             0 │
│ conv2d_1 (Conv2D)               │ (None, 109, 109, 64)   │        18,496 │
│ activation_1 (Activation)       │ (None, 109, 109, 64)   │             0 │
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 54, 54, 64)     │             0 │
│ conv2d_2 (Conv2D)               │ (None, 52, 52, 128)    │        73,856 │
│ activation_2 (Activation)       │ (None, 52, 52, 128)    │             0 │
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 26, 26, 128)    │             0 │
│ flatten (Flatten)               │ (None, 86528)          │             0 │
│ dense (Dense)                   │ (None, 256)            │    22,151,424 │
│ activation_3 (Activation)       │ (None, 256)            │             0 │
│ dropout (Dropout)               │ (None, 256)            │             0 │
│ dense_1 (Dense)                 │ (None, 64)             │        16,448 │
│ activation_4 (Activation)       │ (None, 64)             │             0 │
│ dropout_1 (Dropout)             │ (None, 64)             │             0 │
│ dense_2 (Dense)                 │ (None, 2)              │           130 │
│ activation_5 (Activation)       │ (None, 2)              │             0 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
Total params: 22,261,250 (84.92 MB)
Trainable params: 22,261,250 (84.92 MB)
Non-trainable params: 0 (0.00 B)
```



Conclusion:

CNN are effective for waste classification.

Future work includes model optimization and real-world deployment .

The CNN model successfully classifies waste images. Future improvements include real-time implementation using mobile apps and IoT devices.