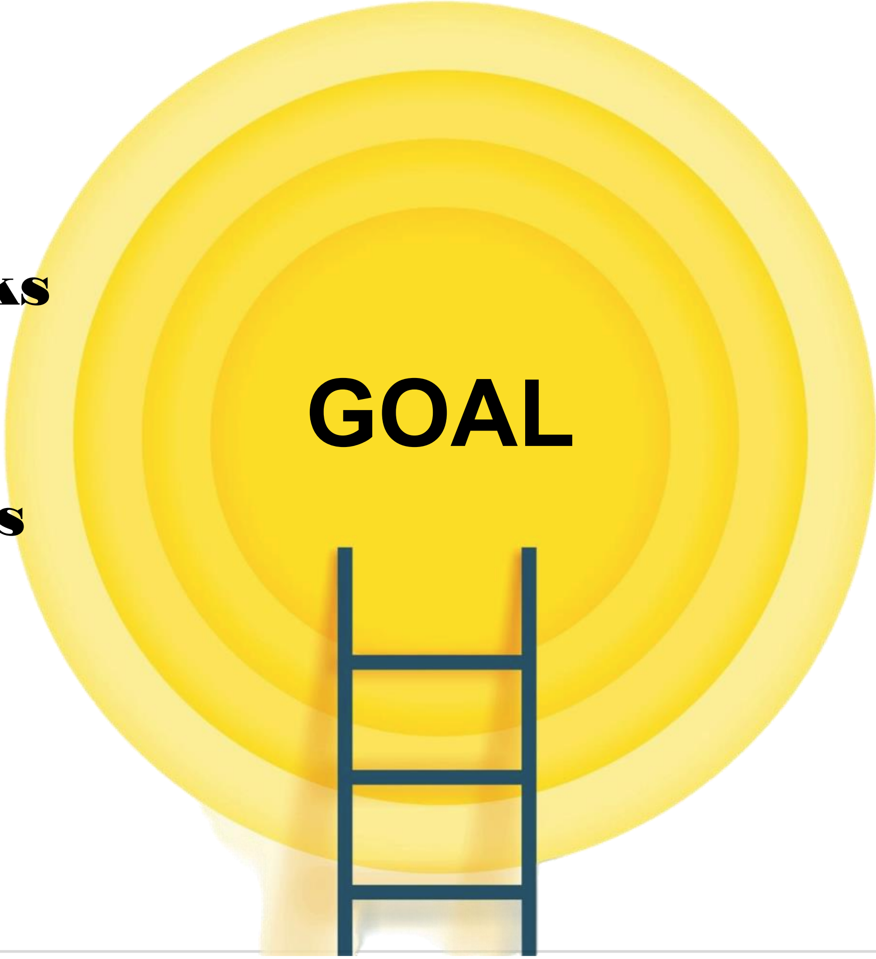
Garbage Classification with EfficientNetV2B2

PRESENTED BY: ADITYA SHARMA

edunet foundation

Learning Objectives :

- Understand image classification in waste management
- Apply EfficientNetV2B2 for classification tasks
- Preprocess and augment datasets
- Evaluate model performance
- Deploy trained models in real-world scenarios



Tools and Technology used :

- Python
- TensorFlow / Keras
- EfficientNetV2B2
- Garbage Classification Dataset (Kaggle)
- Jupyter Notebook / Google Colab
- NumPy, Pandas, Matplotlib, OpenCV

Problem Statement:

To develop an accurate and efficient garbage classification model using EfficientNetV2B2 and transfer learning for automated waste sorting.

edunet foundation

Solution:

1. Leveraging Pre-trained EfficientNetV2B2

•Utilize **EfficientNetV2B2**, a state-of-the-art CNN architecture pre-trained on ImageNet.

•Benefits :

- High accuracy with fewer parameters.
- Faster training and inference.
- Scales well for various datasets.

2. Transfer Learning Approach

•**Freeze initial layers** of EfficientNetV2B2 to retain learned features.

•**Customize final classification layers** (Dense + Softmax) for garbage categories (e.g., Plastic, Metal, Organic, etc.).

•**Fine-tune** on a garbage image dataset for higher accuracy.

3. Garbage Image Dataset

•Train on a labeled dataset of waste images:

- •Categories: **Cardboard , Glass , Metal , Paper , Plastic , Trash**.
- •Data Augmentation: Rotation, Zoom, Flip to handle variability.

4. Model Training and Evaluation

•Use **Adam optimizer**, **categorical crossentropy loss**, and **early stopping** for stable training.

•Evaluate using **accuracy**, **precision**, **recall**, and **F1-score**.

•Achieved accuracy

5. Deployment Readiness

•Export trained model to TensorFlow Lite or ONNX format for edge deployment (e.g., on mobile or smart bins).

•Integrate with a camera-based detection system for **real-time classification** and sorting.

Methodology :

- 1. Dataset Collection
- 2. Data Preprocessing (resize, normalize, augment)
- 3. Model Building (EfficientNetV2B2 + custom layers)
- 4. Training (Adam optimizer, categorical loss)
- 5. Evaluation (accuracy, confusion matrix)

Model Architecture (Diagram) :

- EfficientNetV2B2 (base)
- Global Average Pooling
- Dense (ReLU)
- Dropout
- Dense Softmax (Output Classes)

CODE SCREENSHOT:

```
In [2]:  import numpy as np  # Importing NumPy for numerical operations and array manipulations
         import matplotlib.pyplot as plt  # Importing Matplotlib for plotting graphs and visualizations
         import seaborn as sns  # Importing Seaborn for statistical data visualization, built on top of Matplotlib
         import tensorflow as tf  # Importing TensorFlow for building and training machine learning models
         from tensorflow import keras  # Importing Keras, a high-level API for TensorFlow, to simplify model building
         from tensorflow.keras import Layer  # Importing Layer class for creating custom layers in Keras
         from tensorflow.keras.models import Sequential  # Importing Sequential model for building neural networks layer-by-layer
         from tensorflow.keras.layers import Rescaling , GlobalAveragePooling2D
         from tensorflow.keras import layers, optimizers, callbacks  # Importing various modules for layers, optimizers, and callback
         from sklearn.utils.class_weight import compute_class_weight  # Importing function to compute class weights for imbalanced d
         from tensorflow.keras.applications import EfficientNetV2B2  # Importing EfficientNetV2S model for transfer learning
         from sklearn.metrics import confusion_matrix, classification_report  # Importing functions to evaluate model performance
         import gradio as gr  # Importing Gradio for creating interactive web interfaces for machine learning models
```

```
In [3]:  dataset_dir= r"C:\Users\adity\Downloads\TrashType_Image_Dataset"
         image_size = (124, 124)
         batch_size = 32
         seed = 42
```

```
In [4]:  train_ds = tf.keras.utils.image_dataset_from_directory(
             dataset_dir,
             validation_split=0.2,
             subset="training",
             seed=seed,
             shuffle = True,
             image_size=image_size,
             batch_size=batch_size
         )
```

```
Found 2527 files belonging to 6 classes.
Using 2022 files for training.
```

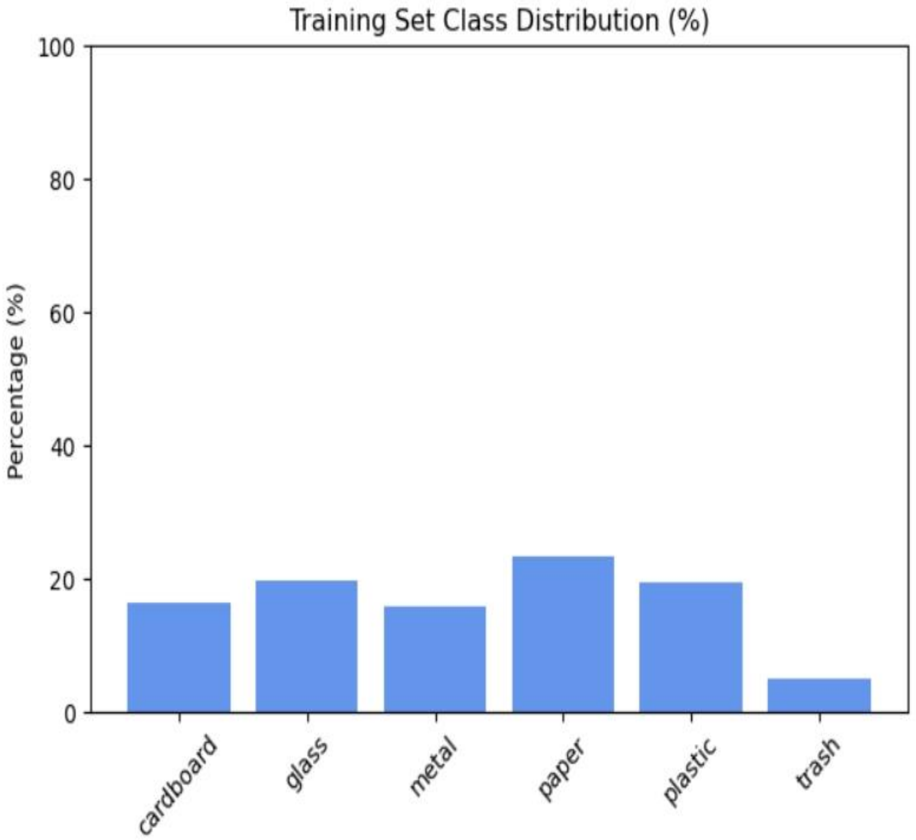
CODE SCREENSHOT:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(12):
    ax = plt.subplot(4, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(train_ds.class_names[labels[i]])
    plt.axis("off")
```

In [12]:

```
# Show visualizations
simple_bar_plot(train_dist, "Training Set Class Distribution (%)")
simple_bar_plot(val_dist, "Validation Set Class Distribution (%)")
simple_bar_plot(test_dist, "Test Set Class Distribution (%)")
simple_bar_plot(overall_dist, "Overall Class Distribution (%)")
```



CODE SCREENSHOT:

```
epochs = 15  # Number of times the model will go through the entire dataset

# Train the model using the fit function
history = model.fit(
    train_ds,                  # Training dataset used to adjust model weights
    validation_data=val_ds,    # Validation dataset to monitor performance on unseen data
    epochs=epochs,             # Number of training cycles, referencing the variable set earlier
    class_weight=class_weights,  # Handles class imbalances by assigning appropriate weights
    batch_size=32,             # Number of samples processed in each training step
    callbacks=[early]          # Implements early stopping to prevent unnecessary training
)
```

```
Epoch 1/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 70s 476ms/step - accuracy: 0.3224 - loss: 1.6997 - val_accuracy: 0.6673 - val_loss: 1.1279
Epoch 2/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 27s 414ms/step - accuracy: 0.6878 - loss: 1.0827 - val_accuracy: 0.7980 - val_loss: 0.7614
Epoch 3/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 27s 415ms/step - accuracy: 0.7973 - loss: 0.7131 - val_accuracy: 0.8475 - val_loss: 0.5521
Epoch 4/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 27s 423ms/step - accuracy: 0.8308 - loss: 0.5213 - val_accuracy: 0.8535 - val_loss: 0.4630
Epoch 5/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 28s 435ms/step - accuracy: 0.8860 - loss: 0.3735 - val_accuracy: 0.8673 - val_loss: 0.3865
Epoch 6/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 28s 438ms/step - accuracy: 0.9076 - loss: 0.2947 - val_accuracy: 0.8733 - val_loss: 0.3565
Epoch 7/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 28s 443ms/step - accuracy: 0.9247 - loss: 0.2286 - val_accuracy: 0.8733 - val_loss: 0.3481
Epoch 8/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 29s 450ms/step - accuracy: 0.9332 - loss: 0.1895 - val_accuracy: 0.8911 - val_loss: 0.3133
Epoch 9/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 29s 446ms/step - accuracy: 0.9470 - loss: 0.1918 - val_accuracy: 0.8812 - val_loss: 0.3146
Epoch 10/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 31s 476ms/step - accuracy: 0.9588 - loss: 0.1269 - val_accuracy: 0.8891 - val_loss: 0.3071
Epoch 11/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 30s 469ms/step - accuracy: 0.9566 - loss: 0.1175 - val_accuracy: 0.8871 - val_loss: 0.2984
Epoch 12/15
64/64 ━━━━━━━━━━━━━━━━━━━━ 30s 465ms/step - accuracy: 0.9697 - loss: 0.1059 - val_accuracy: 0.8911 - val_loss: 0.2781
Epoch 13/15
```

```
]:  # 📝 Summary (optional but useful)
    model.summary()
```

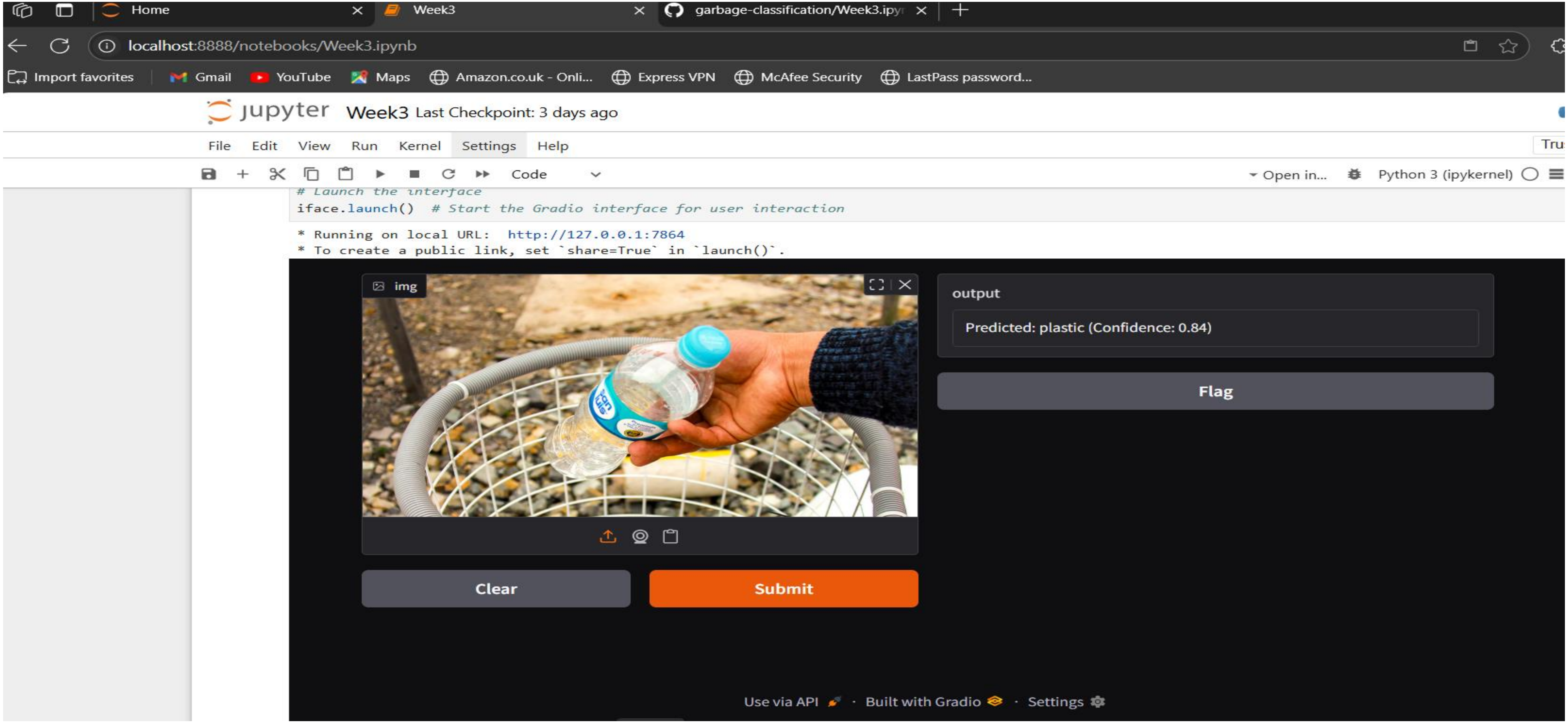
Model: "sequential_1"

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 124, 124, 3)	0
efficientnetv2-b2 (Functional)	(None, 4, 4, 1408)	8,769,374
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1408)	0
dropout (Dropout)	(None, 1408)	0
dense (Dense)	(None, 6)	8,454

Total params: 24,727,114 (94.33 MB)
Trainable params: 7,974,642 (30.42 MB)
Non-trainable params: 803,186 (3.06 MB)
Optimizer params: 15,949,286 (60.84 MB)

FOR FULL CODE VISIT MY GITHUB LINK : https://github.com/AdityaSharma-12/garbage-classification.git

Screenshot of Output:



Conclusion:

- Waste classification can be automated
- EfficientNetV2B2 is accurate and scalable
- Integrates into smart waste management systems

Future Scope :

- Real-time detection on edge/mobile devices
- Classify sub-types of recyclables
- Field testing with local authorities