# Capstone Project Data Preparation Feature Engineering and Model Exploration

## Project Title: Multi-Level Waste Classification System

## Team Members


1. **Khin Yadanar Hlaing** — khinyadanarhlaing.kt@gmail.com
2. **Myo Myat Htun** — myomyatskywalker@gmail.com
3. **Aye Nandar Bo** — ayenandarbo24@gmail.com

# Data Preparation/Feature Engineering

## 1. Overview

The data preparation and feature engineering phase is critical in this machine learning project to ensure the model receives high-quality, standardized inputs. The goal of this project is to classify waste images into categories (e.g., glass, metal, paper, plastic, trash, organic) or binary categories (Recycle vs. Non-recycle) to automate waste sorting. This phase involves loading raw image data, cleaning it, augmenting it to increase dataset diversity, and transforming it into a format suitable for deep learning models like VGG16.

## 2. Data Collection

The dataset was collected and stored in Google Drive. It consists of image files representing different types of waste..

- **Dataset Structure:** The data consists of over 4000 images.
    - **Multi-class Task:** Images are categorized into 6 classes: glass, metal, organic, paper, plastic, and trash.
    - **Binary Task:** Images are categorized into 2 classes: Recycle and Non-recycle.

## 3. Data Cleaning

To clean the raw data, the following steps were taken:

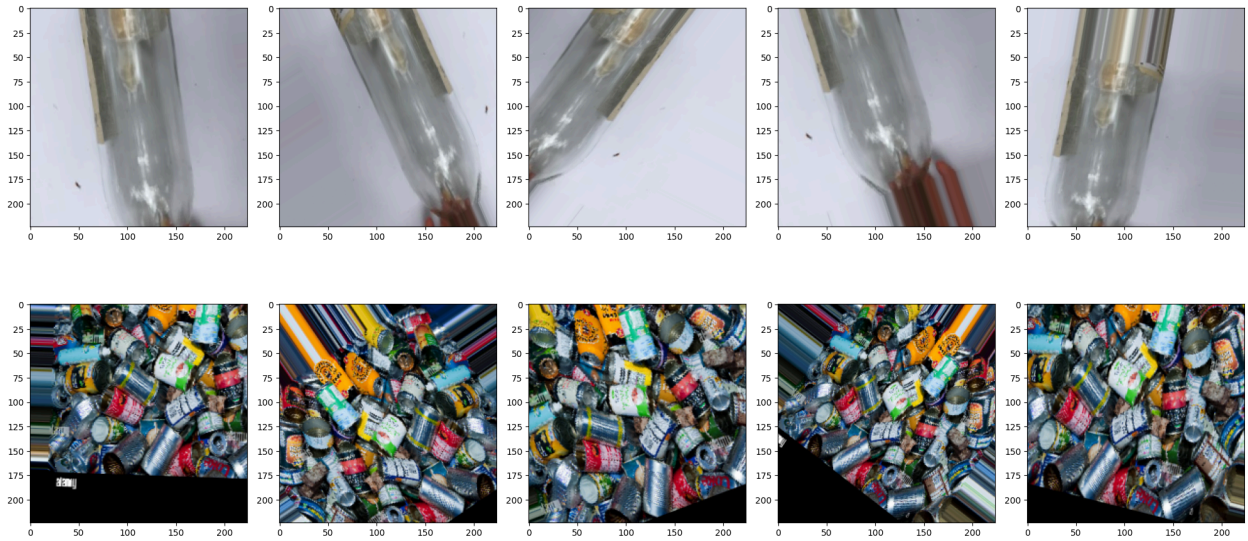
- **Corrupt File Handling:** The image loading process included checks to ensure files were valid images. If an image could not be read (e.g., using cv2.imread), it was skipped and a warning was logged.
- **Resizing:** All images were resized to a uniform dimension of **224 x 224 pixels** to match the input requirements of the pre-trained VGG16 and ResNet50 models.
- **Interpolation:** Nearest-neighbor interpolation was used during resizing to preserve image details

## 4. Exploratory Data Analysis (EDA)

Exploratory analysis was performed to understand the class distribution and verify data quality.

- Visualization: Sample images from each category were plotted to verify correct labeling and image quality.
- Class Mapping for multi-class classification: A dictionary was created to map class names to integer labels (e.g., `{'glass': 0, 'metal': 1, 'organic': 2, 'paper': 3, 'plastic': 4, 'trash': 5}`)
- Class Mapping for binary classification: A dictionary was created to map class names to integer labels (eg. {‘recycle’:1 , ‘non-recycle’ : 0 } )

***Fig. A grid display of random sample images from different classes as generated by `plotImages` function***



## 5. Feature Engineering

Feature engineering for this image data primarily relied on **Data Augmentation** to create new "features" and variations from existing data, preventing overfitting:

- **Technique:** ImageDataGenerator from Keras was utilized.
- **Augmentation Parameters:**
    - rescale=1./255: Normalization of pixel values.
    - rotation_range=40: Random rotations to handle orientation variance.
    - width_shift_range=0.2 & height_shift_range=0.2: Handling off-center objects.
    - shear_range=0.2 & zoom_range=0.2: Handling perspective and scale differences.

- horizontal_flip=True: Handling mirror images

### 6. Data Transformation

- **Scaling:** Pixel values were rescaled from the range [0, 255] to [0, 1] to aid model convergence.
- **Encoding:** Target labels (class names) were encoded into categorical vectors (One-Hot Encoding) using class_mode='categorical' in the data generator.
- **Array Conversion:** Image lists were converted into NumPy arrays for efficient computation.

# Model Exploration

## 1. Model Selection

Two deep learning models based on Convolutional Neural Networks (CNNs) were selected for this project using **Transfer Learning**: **VGG16**

**Rationale:**

- These models have been pre-trained on the massive ImageNet dataset, allowing them to extract rich feature representations (edges, textures, shapes) without training from scratch.
- **Strengths:** High accuracy on image recognition tasks; established architectures with proven performance.
- **Weaknesses:** Can be computationally expensive and large in size (especially VGG16).

## 2. Model Training

The pre-trained base (VGG16) was used as a feature extractor (layers frozen). A custom classification head was added:

- Flatten layer.
- Dense output layer with softmax activation for classification probabilities.

Hyperparameters:

- Optimizer: Adam.
- Loss Function: Categorical Crossentropy.
- Batch Size: 32 (augmented data) or 256.
- Epochs: Set to 100, but controlled by Early Stopping**.**

**Callbacks:**

- ModelCheckpoint: To save the best weights.
- EarlyStopping: To stop training if validation loss did not improve for 4 consecutive epochs, preventing overfitting.

## 3. Model Evaluation

The models were evaluated using the following metrics on the test dataset:

- Accuracy Score: To measure overall performance.
- Confusion Matrix: To visualize misclassifications between specific classes.
- Classification Report: To check Precision, Recall, and F1-Score for each waste category.
- Learning Curves: Plots of Training vs. Validation Loss and Accuracy were generated to diagnose overfitting or underfitting.

***Fig. The 'Training vs Validation Accuracy' and 'Loss' for Binary Classification (Recycle or Non-recycle)***

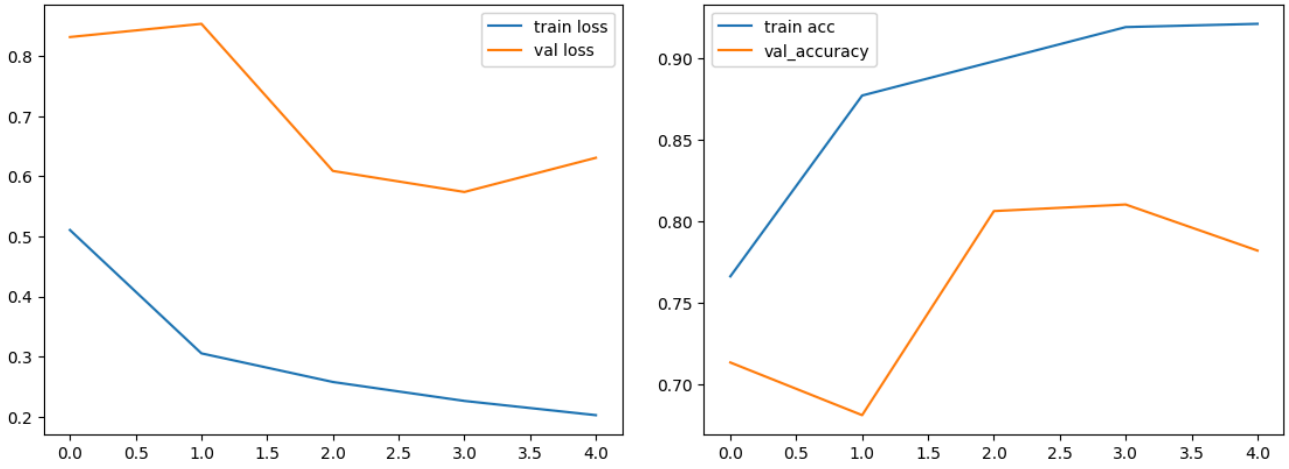



***Fig. The 'Training vs Validation Accuracy' and 'Loss' for Multi Classification***

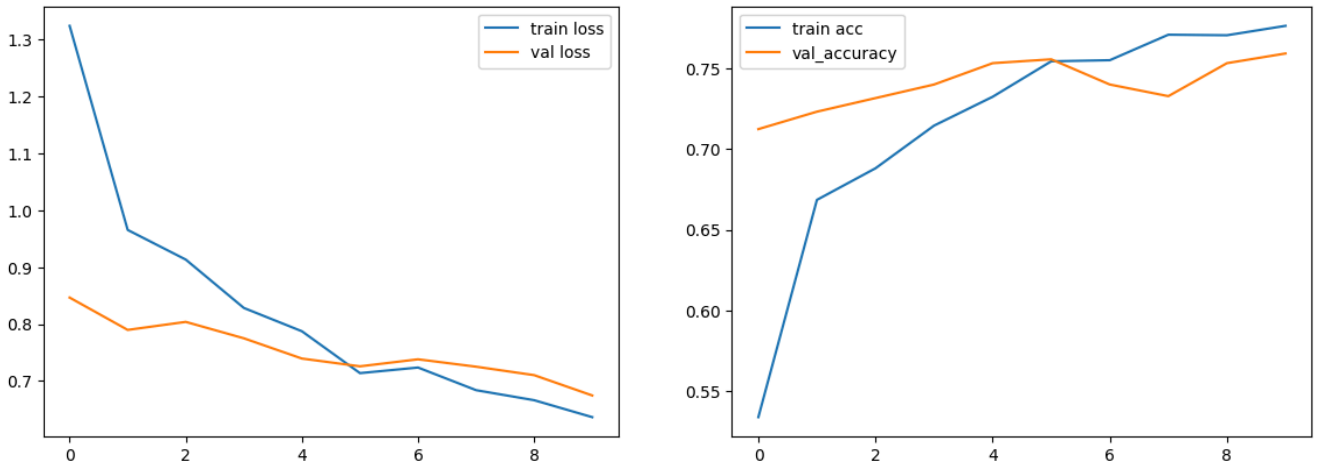


*Fig. The Confusion Matrix heatmap for Binary Classification*

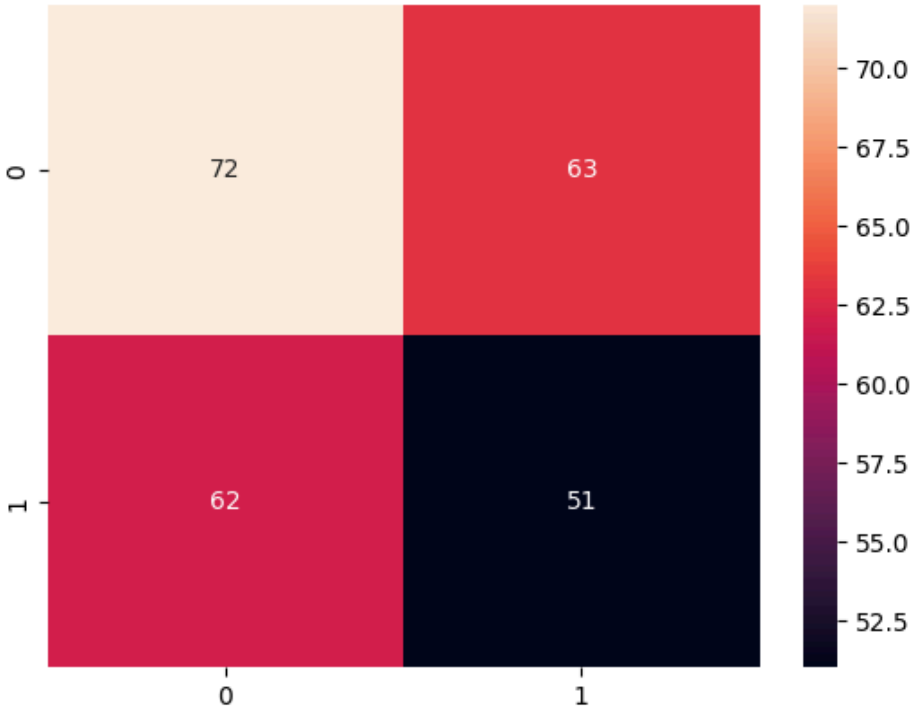
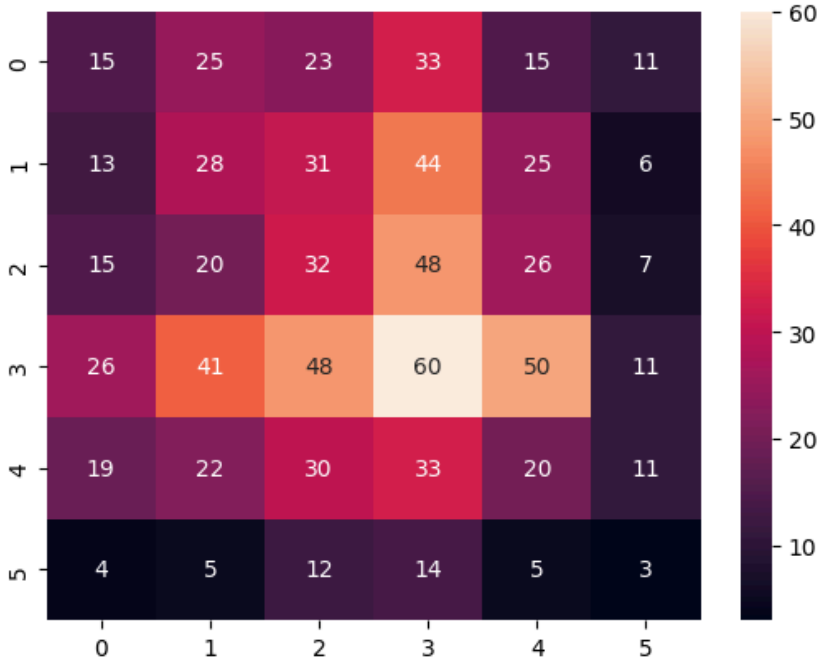



*Fig. The Confusion Matrix heatmap for Multi Classification*



## 4. Code Implementation

**Notebook files are committed to Github**
https://github.com/edasaruhan/FTL_Myanmar_Gr14