# Real-time Garbage Classification using Transfer Learning with NASNet & MobileNetV2

* Gowtham, IIIrd Year (VCET)

## Problem Statement

The project aims to develop a real-time garbage classification system capable of identifying different types of waste materials through computer vision using transfer learning techniques. This system will help automate waste sorting processes and improve recycling efficiency.

## Proposed Solution

The proposed solution uses transfer learning with NASNetMobile architecture pre-trained on ImageNet. It processes 224x224x3 RGB images to classify them into six categories: cardboard, glass, metal, paper, plastic, and trash. The architecture preserves NASNet's pre-trained layers while adding custom classification layers including GlobalAveragePooling2D, Dropout (0.5), and a final Dense layer with softmax activation. The model is trained with categorical crossentropy loss and the Adam optimizer, with fine-tuning of the last 15 layers for optimal waste classification performance.

## Materials Required

### Hardware Requirements:

* Webcam for real-time detection
* Minimum 8GB RAM
* GPU recommended for faster training or use Colab T4 GPU

### Software Requirements:

* Python 3.7+
* TensorFlow 2.x
* OpenCV
* NumPy
* Matplotlib
* PIL (Python Imaging Library)
* Seaborn
* Scikit-learn

## Dataset Details

The project uses the TrashNet dataset, which contains 2527 images of garbage items across six categories:

1. Dataset Source: [TrashNet on Kaggle](https://www.kaggle.com/datasets/feyzazkefe/trashnet)
2. Image Resolution: 300x300 pixels
3. Format: JPG
4. Categories: 6 (cardboard, glass, metal, paper, plastic, trash)

## Code Repository

The complete project code & model is available in this google drive folder :

[Trash Classification - Transfer Learning (Drive)](https://drive.google.com/drive/folders/1EkmroHCBwRnxI2g0oLck8n4n5aiEHtss?usp=drive_link)

*Folder Structure :*

- Jupyter Notebook used for training :

[(NasNetMobile) Transfer Learning - Trash Classification.ipynb](https://drive.google.com/open?id=1JhPZ0uXwNnUSke080jPM1ZHUyTAqS_eW&usp=drive_copy)

[(MobileNetV2) Transfer Learning - Trash Classification.ipynb](https://drive.google.com/open?id=1wtDxEcLi3RLl3o54y_Sawy6JLS0yu9Oc&usp=drive_copy)

- Model File : [models](https://drive.google.com/open?id=1ZUs1rlMBwTWcuQPuow2QEnyA0Xhqls1j&usp=drive_copy)  
- For running the model realtime : [realtime\_trash\_detection\_TRANSFER\_LEARNING.py](https://drive.google.com/open?id=1vJzM-IbHhNtO9hNlXHfQcClc0ZT2qdrg&usp=drive_copy)  
- Results : [test images & results](https://drive.google.com/open?id=16jxPfP40sVVWTDzOEx8idPRvLGVOcAen&usp=drive_copy)

## Implementation Procedure

#### *1. Preparing the Data*

**Dataset Organization:**

The dataset should be organized in a specific folder structure for proper training:

* + Create a main folder called 'garbage\_dataset'
  + Inside it, create six subfolders named: cardboard, glass, metal, paper, plastic, and trash
  + Place all relevant images in their respective category folders
  + Ensure all images are in standard formats like JPG or PNG

**Preprocessing:**

Each image goes through several preparation steps:

* + - 1. First, load each image from its category folder
  1. Convert each image to a standard size of **224x224 pixels**
     + This ensures all inputs to the model are consistent
     + Maintains image proportions while fitting the required dimensions
  2. Normalize the pixel values to a range of **0 to 1**
     + Divide all pixel values by 255 (original range is 0-255)
     + This normalization helps the model train more effectively
  3. Split the dataset into two portions:
     + 80% for training the model
     + 20% for validating the model's performance

**Data Generation Setup:**  
The data preparation process is automated using an image generator that:

* + Loads images in batches of 64 at a time
  + This batch size is chosen to:
    - Efficiently use computer memory
    - Provide stable learning updates
    - Balance between speed and accuracy
  + Automatically handles the training/validation split
  + Ensures images are properly formatted for the model

#### *2. Designing the Model*

The transfer learning model is built systematically:

1. **Base Model Preparation:**
   * Start with the NASNetMobile model
   * This model comes pre-trained on ImageNet (millions of images)
   * Configure it to accept our 224x224 pixel RGB images
   * Remove its original classification layers since we'll add our own
2. **Layer Configuration:**
   * Leave the early layers unchanged
     + These layers contain valuable general image features
     + They help identify basic elements like edges, textures, and shapes
   * Make the last 15 layers trainable
     + These layers will adapt to our specific waste classification task
     + Allows the model to learn waste-specific features
   * This balance ensures we keep useful features while adapting to our needs
3. **Classification Layer Setup:**
   * Add a Global Average Pooling layer
     + This reduces the feature maps to a manageable size
     + Helps prevent overfitting
   * Include a Dropout layer set to 50%
     + Randomly deactivates half the neurons during training
     + Makes the model more robust and prevents memorization
   * Add the final classification layer
     + Contains 6 neurons (one for each waste category)
     + Uses softmax activation for probability outputs

#### *3. Training the Model*

* **Loss Function:**
  + Use Categorical Crossentropy
  + Perfect for multi-class classification problems
  + Helps the model learn to distinguish between waste categories
* **Optimizer:**
  + Employ the Adam optimizer
  + Automatically adjusts learning rates
  + Provides efficient and stable training
* **Training Process:**
  + Run for 20 complete epochs
  + During each epoch:
    - Process all training images in batches
    - Check performance on validation data
    - Adjust model weights accordingly
    - Save training progress and metrics

#### *4. Saving the Model*

The trained model needs to be saved in two formats:

1. Full model in H5 format
   * Named "garbage\_classification\_tf\_nasnet.h5"
   * Used for regular computer-based applications
2. Exporting as compressed TFLite format
   * Named "converted\_model.tflite"
   * Optimized for mobile and embedded devices

#### *5. Making Predictions*

To classify a new image, the process involves:

1. Image Preparation:
   * Resize the new image to 224x224 pixels
   * Normalize its pixel values (divide by 255)
2. Classification Process:
   * Feed the prepared image through the model
   * The model processes it through all layers
   * Get probability scores for each category
   * Select the category with highest probability

Example Classification:  
When showing a plastic bottle image, you might get:

* Plastic: 85%
* Glass: 8%
* Metal: 4%
* Other categories: remaining 3%

### How This Model Works Technically

* The model uses pre-trained features from NASNet
* These features were learned from millions of ImageNet images
* The last 15 layers are fine-tuned specifically for waste
* This combination allows efficient learning with less training data
* The model balances general image understanding with waste-specific features

## Results :

**NASNet Mobile (30 epochs)**:

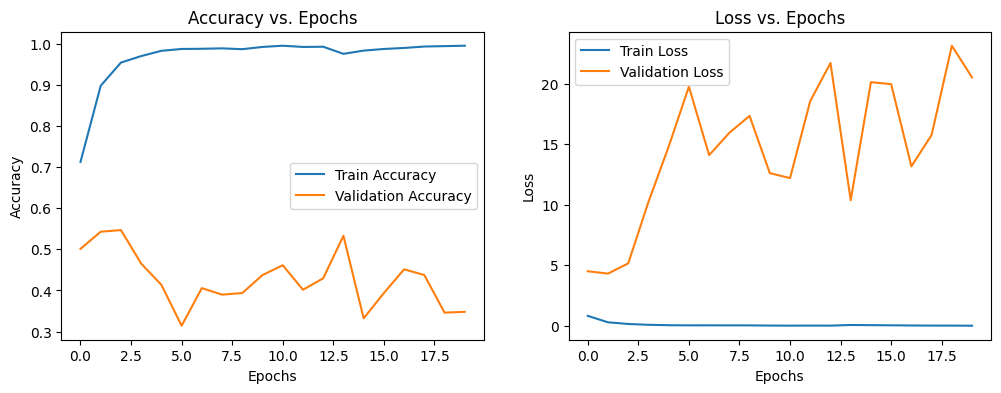
* Final Training Accuracy: 99.92%
* Final Validation Accuracy: 75.75%
* More balanced predictions across categories

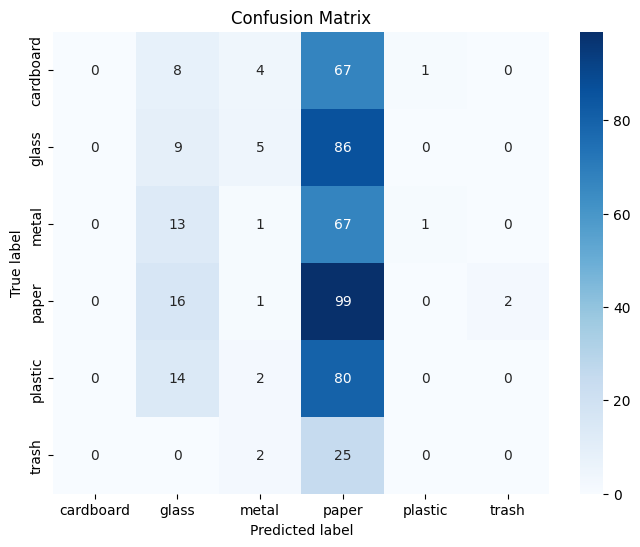
**MobileNetV2 (20 epochs)**:

* Final Training Accuracy: 99.63%
* Final Validation Accuracy: 34.79%
* Significant misclassification of most categories as 'paper'

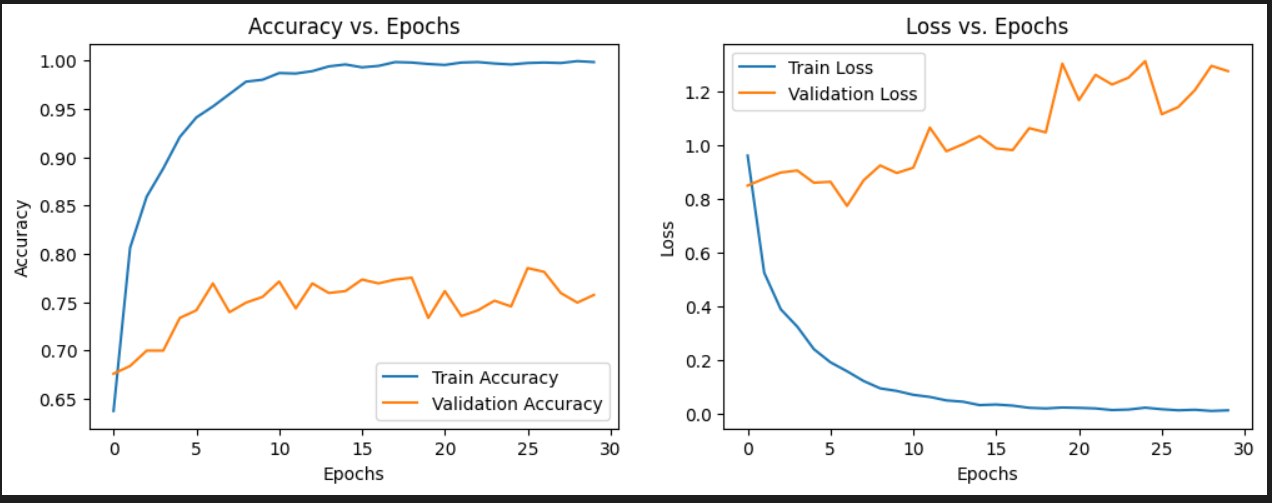
On real world usage, **NasNet** is better than MobileNetV2.

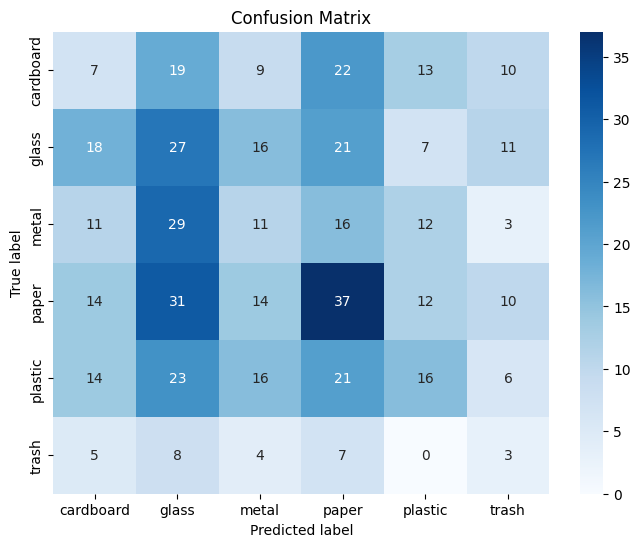
**MobileNetV2:**





**NasNetMobile :**





## Local Setup and Running

1. Install dependencies:

pip install tensorflow opencv-python numpy pillow matplotlib seaborn scikit-learn

1. Download the trained model:

- [MobileNet TFLite.tflite](https://drive.google.com/open?id=1dao8nuLDSRkL4WBj8xVjz0BChBWFbAZg&usp=drive_copy)  
- [trash\_classification\_tf\_mobilenet.h5](https://drive.google.com/open?id=1MJzpYQo7jK-fZjjuM29YF9_WvunILpvE&usp=drive_copy)  
 - [trash\_classification\_tf\_nasnet.h5](https://drive.google.com/open?id=14lXAJL3we5qllEsYdZPFLYmD5QbMcNeN&usp=drive_copy)

1. Run predictions using the provided python file:

python [realtime\_trash\_detection\_TRANSFER\_LEARNING.py](https://drive.google.com/open?id=1vJzM-IbHhNtO9hNlXHfQcClc0ZT2qdrg&usp=drive_copy)

(choose the correct model)