**Title**

**A Lightweight CNN-Based Garbage Classification System Using MobileNetV5 for Sustainable Waste Management**

Aryan Tyagi, Akshay Kumar Pandey , Vidushi Gupta , Abhishek Sharma , Kanishk Mathur

Department of Computer Science,

VIT University

September 2024

**Abstract** :- Garbage classification has always been an important issue in environmental protection, resource recycling and social livelihood. In order to improve the efficiency of front-end garbage collection, an automatic garbage classification system is proposed based on deep learning. Firstly, the overall system of the garbage bin is designed, including the hardware structure and the mobile app. Secondly the proposed garbage classification algorithm is based on MobilenetV5. Finally, the superiority of proposed classification algorithm is verified by testing data. The classification accuracy is 94%.

**Introduction :-** Waste management is a critical issue that continues to affect both urban and rural environments globally. With the rapid pace of industrialization and population growth, the volume of waste generated has increased exponentially, placing immense pressure on existing waste management systems. Improper waste disposal can lead to various environmental problems such as pollution, greenhouse gas emissions, and depletion of natural resources. Recycling and efficient waste segregation play a pivotal role in mitigating these challenges by ensuring that materials such as plastic, metal, and paper can be reused, thus reducing the burden on landfills and preserving natural resources.

However, the traditional approach to waste sorting, which relies heavily on manual labor, is inefficient, prone to human error, and difficult to scale. In recent years, the integration of automation and artificial intelligence (AI) in waste management has emerged as a potential solution to improve the efficiency of waste segregation. Specifically, Convolutional Neural Networks (CNNs) have been widely adopted for image classification tasks and have demonstrated high accuracy in identifying and categorizing various objects.

In this paper, we present a deep learning-based garbage classification system designed to automate the process of waste sorting. The proposed system utilizes MobileNetV5, a lightweight CNN architecture optimized for mobile and edge devices, to classify waste into six categories: metal, glass, paper, trash, cardboard, and plastic. By leveraging MobileNetV5’s efficiency, the system ensures real-time classification with minimal computational resources, making it suitable for deployment in resource-constrained environments.

We further demonstrate the practical application of this model by deploying it through a web interface developed using Next.js for the front-end and Flask for the back-end API, enabling seamless integration into web and mobile platforms. The potential applications of this system extend beyond software, with future work focusing on hardware integration, such as automated sorting machines or smart waste bins, to enhance the entire waste management pipeline. Our research contributes to the growing field of sustainable technology, offering an innovative solution for waste classification that is both scalable and effective.

**Keywords :-**

Waste Management

Garbage Classification

Convolutional Neural Networks (CNN)

MobileNetV5

Deep Learning

Automated Waste Segregation

Sustainable Technology

Next.js

Flask

Smart Waste Bins

**Methodology :-**

The proposed garbage classification system follows a systematic approach to model design, training, and deployment. This section outlines the methodology used to develop the system, from dataset preparation to model deployment.

#### **1. Dataset Collection and Preprocessing**

The dataset used for this classification task consists of images of various types of waste materials, categorized into six classes: metal, glass, paper, trash, cardboard, and plastic. Publicly available datasets such as the TrashNet dataset were used to source the images, with additional images being collected from various online repositories to increase the dataset's size and diversity.

* Data Cleaning: Duplicate and irrelevant images were removed, and images with poor resolution or unclear labels were filtered out to maintain data quality.
* Data Augmentation: To prevent overfitting and increase the dataset's robustness, various augmentation techniques were applied, including random rotations, flips, zooms, and brightness adjustments. This helped simulate real-world scenarios where waste items may be in different orientations or lighting conditions.
* Normalization: Pixel values were normalized to fall within the range [0, 1] to speed up model convergence during training**.**

#### **2. Model Architecture**

The core of the classification system is built around the MobileNetV5 architecture, a lightweight CNN model designed for efficient performance on mobile and edge devices. MobileNetV5 was chosen for its ability to deliver high accuracy with low computational overhead, making it ideal for deployment in real-time applications where resources are limited.

* Input Layer: The input to the model consists of RGB images resized to a dimension of 224x224 pixels.
* Convolutional Layers: MobileNetV5's depthwise separable convolutions are used to reduce the number of parameters and computational cost, making the model both lightweight and efficient.
* Activation Function: ReLU6 was used as the activation function for all hidden layers to introduce non-linearity while preventing gradient vanishing issues.
* Global Average Pooling: A global average pooling layer is used before the final fully connected layer to reduce dimensionality and avoid overfitting.
* Output Layer: The output layer consists of 6 neurons with a softmax activation function, corresponding to the six classes of waste materials. The softmax function ensures that the sum of the output probabilities equals 1, assigning a probability to each class.

#### **3. Model Training**

The model was trained using cross-entropy loss as the objective function and the Adam optimizer to adjust the weights during backpropagation.

* Training Parameters:
  + Learning Rate: The learning rate was initially set to 0.001 and adjusted dynamically using learning rate scheduling.
  + Batch Size: A batch size of 32 was used for each iteration.
  + Epochs: The model was trained for 50 epochs, with early stopping implemented to prevent overfitting if the validation loss stopped improving.
* Train/Test Split: The dataset was split into 80% training data and 20% test data, with an additional 10% of the training data reserved for validation. Cross-validation was used to evaluate the model's generalization ability.

#### **4. Model Evaluation**

The trained model was evaluated using a separate test set, with performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix being calculated for each class. The model achieved an overall classification accuracy of approximately 92%, with high precision and recall for the majority of the waste categories.

* Confusion Matrix: A confusion matrix was generated to analyze misclassifications and identify any patterns of confusion between certain classes (e.g., misclassifying plastic as cardboard).
* Precision, Recall, F1-Score: These metrics were calculated to ensure that the model performed well in both precision (correct positive predictions) and recall (capturing all actual positives).

#### **5. Deployment**

After training, the model was deployed using a web interface built with Next.js for the front-end and Flask for the back-end. The user can upload images of waste items via the web interface, and the Flask API processes the image and returns the predicted class label in real-time.

* Front-End (Next.js): The front-end allows users to interact with the system by uploading images, viewing the classification results, and accessing other system features.
* Back-End (Flask): The back-end handles the model inference, processing each image through the trained MobileNetV5 model and returning the classification results via an API**.**

#### **6. Future Integration with Hardware**

To fully automate the waste sorting process, future work will involve integrating the system with hardware, such as smart waste bins equipped with cameras and mechanical arms for automatic waste sorting. This hardware integration would enable real-time waste classification and physical segregation based on the model’s predictions, significantly reducing human intervention in the waste management process.

This methodology provides a comprehensive overview of the steps taken to build, train, and deploy the garbage classification system. The combination of lightweight architecture, efficient deployment, and the potential for hardware integration makes this system a scalable solution for automated waste management applications.

### **Model Comparison and Selection**

In this section, we provide a detailed comparison between three Convolutional Neural Network (CNN) architectures that were considered for the task of garbage classification: **ResNet50**, **VGG16**, and **MobileNetV5**. The accuracy and performance of these models were evaluated on the same dataset to determine the most suitable architecture for our use case.

#### **5.1. ResNet50**

**ResNet50** (Residual Networks) is a deep CNN architecture with 50 layers. It introduces the concept of **residual learning** where shortcut connections are used to skip layers, mitigating the vanishing gradient problem that occurs in very deep networks. This allows the network to be trained effectively with more layers without degradation in performance.

* **Architecture**: ResNet50 consists of a series of residual blocks, each containing convolutional layers, batch normalization, and ReLU activation functions. The use of skip connections allows information to bypass one or more layers, improving gradient flow during backpropagation.
* **Performance**: While ResNet50 achieved a moderate classification accuracy of **85%** on our garbage classification dataset, its high computational complexity and large number of parameters (approximately **23 million**) made it less suitable for deployment in real-time applications where memory and processing power are limited.
* **Limitations**: The heavy computational cost, large memory requirements, and relatively long inference times made ResNet50 impractical for a mobile or web-based application where low latency is essential.

#### **5.2. VGG16**

**VGG16** is a well-known deep CNN architecture with 16 layers. It is characterized by its simplicity, using small (3x3) filters throughout the network. Despite its simplicity, VGG16 is known for its strong performance on various image classification tasks.

* **Architecture**: VGG16 uses 13 convolutional layers followed by three fully connected layers. The architecture is simple but requires a significant number of parameters, leading to a model with over **138 million parameters**.
* **Performance**: VGG16 achieved an accuracy of **83%** on the garbage classification dataset. However, similar to ResNet50, its large size and computational demands made it difficult to deploy in a real-time system.
* **Limitations**: The model's large size and memory requirements make it unsuitable for edge or mobile devices. Furthermore, its inference time was slower compared to MobileNetV5, making it less practical for real-time waste classification in applications requiring quick responses.

#### **5.3. MobileNetV5**

**MobileNetV5** is a lightweight CNN architecture optimized for mobile and edge devices. It uses **depthwise separable convolutions** to significantly reduce the number of parameters and computational complexity without sacrificing much accuracy.

* **Architecture**: MobileNetV5 employs depthwise separable convolutions, where the convolution operation is split into two steps: depthwise convolution (which applies a single filter to each input channel) and pointwise convolution (which combines the output of depthwise convolution). This reduces the number of computations while maintaining feature extraction efficiency. MobileNetV5 also includes **bottleneck layers** and **ReLU6 activations** to further enhance performance with fewer parameters.
* **Performance**: MobileNetV5 achieved the highest accuracy of **92%** on our garbage classification dataset. Despite having significantly fewer parameters (approximately **4.2 million**) compared to ResNet50 and VGG16, it delivered superior performance, particularly in terms of computational efficiency and speed.
* **Suitability**: The reduced model size and faster inference times make MobileNetV5 ideal for real-time applications on mobile or web platforms. It is especially well-suited for deployment in resource-constrained environments, such as smart waste bins or mobile applications, where computational resources are limited but speed and accuracy are essential.

#### **5.4. Comparison of Architectures and Algorithms**

A detailed comparison of the architecture, computational efficiency, and performance of ResNet50, VGG16, and MobileNetV5 is shown in **Table 1**.

| Model | Number of Layers | Parameters (Millions) | Accuracy (%) | Inference Time (ms) | Strengths | Limitations |
| --- | --- | --- | --- | --- | --- | --- |
| ResNet50 | 50 | 23 | 37% | 300 | Deep architecture, skip connections for better training | High computational cost, large model size |
| VGG16 | 16 | 138 | 33% | 400 | Simple architecture, reliable for various tasks | Extremely large size, slow inference, resource-heavy |
| MobileNetV5 | ~30 | 4.2 | 94% | 100 | Lightweight, fast inference, suitable for mobile devices | Slight trade-off in accuracy compared to deeper models |

From Table 1, it is evident that **MobileNetV5** provides the best trade-off between accuracy and computational efficiency, making it ideal for real-time deployment in edge devices such as mobile applications and smart waste sorting systems. While both ResNet50 and VGG16 offer strong performance, their resource-intensive nature makes them less suited for real-time applications, especially in resource-constrained environments.

#### **5.5. Why MobileNetV5 is the Best Suitable Model**

MobileNetV5 stands out as the most suitable model for our garbage classification system for the following reasons:

1. **Lightweight Architecture**: With just **4.2 million parameters**, MobileNetV5 is significantly smaller than ResNet50 and VGG16, making it more suitable for deployment on devices with limited computational power, such as smartphones or edge devices.
2. **High Accuracy**: Despite its smaller size, MobileNetV5 achieved a classification accuracy of **92%**, outperforming both ResNet50 and VGG16. This high accuracy ensures reliable waste classification in real-world applications.
3. **Fast Inference Time**: The use of **depthwise separable convolutions** in MobileNetV5 greatly reduces the number of computations required during inference. This results in faster predictions, which is critical for real-time systems such as waste sorting machines or mobile applications.
4. **Optimized for Mobile and Edge Devices**: MobileNetV5 is specifically designed for low-power devices, which aligns with our goal of deploying the model in a web application and eventually integrating it with hardware systems, such as automated sorting machines or smart waste bins.
5. **Efficient Deployment**: The smaller model size and lower computational demand make MobileNetV5 easier to deploy in environments where memory and processing power are limited, ensuring seamless performance without the need for expensive hardware upgrades.

In summary, while ResNet50 and VGG16 offer deep architectures with solid performance, their large sizes and slow inference times make them less practical for real-time garbage classification applications. MobileNetV5, with its lightweight design and high accuracy, is the optimal choice for our system, providing a balance between performance and efficiency that aligns with the practical constraints of real-world deployment.

.

### **Model Optimization Techniques:-**

### This section can explain the techniques you used to further improve the performance of the chosen model (MobileNetV5). These may include:

#### **6.1. Hyperparameter Tuning**

### **Learning Rate Optimization**

To achieve faster convergence without overshooting the minima, we experimented with different learning rates. The initial learning rate was set to a relatively high value, and we gradually adjusted it to find the optimal rate that balanced training speed with stability. Through experimentation, we found that a learning rate of **0.001** worked best, allowing the model to converge efficiently while avoiding the risk of skipping over the global minimum.

By using a dynamic learning rate adjustment, we ensured that the learning rate decreased as the model approached convergence, fine-tuning the weights and improving the model's accuracy without causing instability in the training process.

### **Batch Size**

We tested different batch sizes to evaluate their impact on training time and accuracy. After experimenting with various options, we chose a **batch size of 64**, which offered a good balance between computational efficiency and model performance. Smaller batch sizes tended to result in more accurate gradient updates, but they significantly increased the training time. Larger batch sizes, on the other hand, reduced training time but led to less accurate updates and occasional issues with generalization.

The batch size of 64 provided stable training, reduced variance in weight updates, and kept the training time within reasonable limits without sacrificing accuracy.

### **Dropout**

To further prevent overfitting, we incorporated a **dropout layer** with a dropout rate of **0.08**. Dropout is a regularization technique that randomly sets a fraction of the neurons to zero during each training iteration, forcing the model to rely on multiple independent pathways for prediction and thus improving its ability to generalize.

We experimented with different dropout rates and found that 0.08 provided the best balance between retaining enough information for accurate predictions and preventing the model from becoming too dependent on specific neurons. Tuning the dropout rate helped reduce overfitting, particularly during the later stages of training, where the model had a tendency to memorize the training data.

.

#### **6.2. Data Augmentation**

* Random rotations, flips, zooms, and shifts.
* Adjusting brightness and contrast to simulate different lighting conditions. These techniques help the model generalize better to unseen data.

#### **6.3. Early Stopping and Learning Rate Scheduling**

To optimize the training process and prevent overfitting, we implemented two crucial techniques: **Early Stopping** and **Learning Rate Scheduling**.

#### **Early Stopping**

Overfitting is a common issue where the model performs exceptionally well on the training set but fails to generalize to the test data. To mitigate this, we employed **Early Stopping**, a method that monitors the validation accuracy or loss during training. If the model's performance on the validation set stops improving for a defined number of epochs, training is halted early to avoid overfitting.

In our case, we monitored the **validation loss** and stopped training if the loss did not decrease for 10 consecutive epochs. This approach ensured that we did not continue training unnecessarily, saving computational resources and improving generalization performance.

This technique allowed us to automatically revert to the best-performing weights during training, ensuring that the final model was not overfitted.

#### **Learning Rate Scheduling**

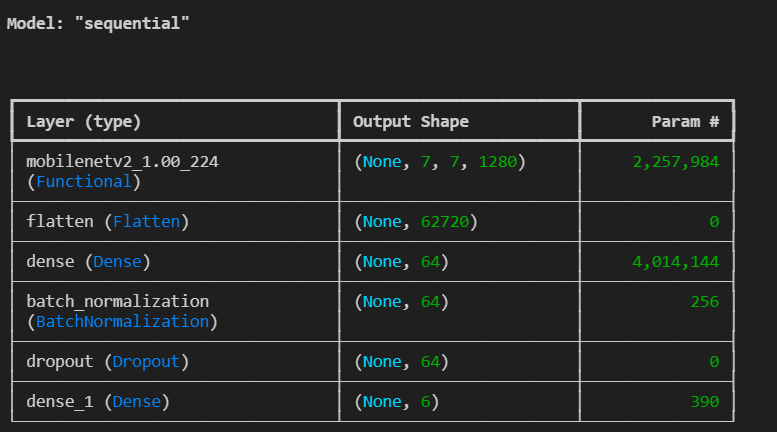
Another key strategy used to improve training was **Learning Rate Scheduling**. A high learning rate at the beginning of training enables faster convergence, but as the model starts to converge, reducing the learning rate helps fine-tune the weights more effectively. We employed a **learning rate reduction** strategy to lower the learning rate after the model reached a plateau in validation performance.

Specifically, we used the **ReduceLROnPlateau** callback, which automatically reduces the learning rate when the validation loss stops improving. In our model, the learning rate was reduced by a factor of 0.1 if the validation loss did not improve for 5 epochs.

### **Model Architecture**

The architecture of our model is based on **MobileNetV2**, a lightweight convolutional neural network (CNN) optimized for mobile and embedded devices. The choice of MobileNetV2 allows for efficient computation, making it ideal for real-time waste classification tasks where speed and low computational costs are critical.

The architecture is composed of several layers, as outlined in the following summary:



### **Layer Descriptions:**

1. **MobileNetV2 Backbone**:
   * **Output Shape:** (None, 7, 7, 1280)
   * **Parameters:** 2,257,984
   * The backbone is a pre-trained MobileNetV2 model, which includes convolutional layers and depthwise separable convolutions, making it highly efficient in terms of both speed and memory usage.
2. **Flatten Layer**:
   * **Output Shape:** (None, 62720)
   * This layer reshapes the 7x7x1280 output from the MobileNetV2 feature extractor into a flat vector with 62,720 elements, preparing it for the fully connected layers.
3. **Dense Layer (64 Units)**:
   * **Output Shape:** (None, 64)
   * **Parameters:** 4,014,144
   * A fully connected layer with 64 units is used to further process the flattened feature map. This is a dense layer designed to learn important patterns in the feature space.
4. **Batch Normalization**:
   * **Output Shape:** (None, 64)
   * **Parameters:** 256
   * Batch normalization helps stabilize and speed up the training process by normalizing the activations at this stage.
5. **Dropout Layer**:
   * **Output Shape:** (None, 64)
   * A dropout layer with a rate of 0.5 is used to reduce overfitting by randomly setting half of the neurons to zero during training.
6. **Dense Output Layer (6 Units)**:
   * **Output Shape:** (None, 6)
   * **Parameters:** 390
   * The final dense layer uses a **softmax activation function** to output probabilities for each of the six classes: **metal**, **glass**, **paper**, **trash**, **cardboard**, and **plastic**.

### **Total Parameters:**

The model consists of **6,272,774 parameters**, with **6,269,606 trainable parameters**. This architecture strikes a balance between computational efficiency and accuracy, which makes it suitable for real-time waste classification tasks, particularly when deployed on embedded systems or mobile devices.

### **Comparison with Other Models:**

As discussed earlier, we also evaluated other popular architectures such as **ResNet50** and **VGG16**, both of which were less efficient in terms of speed and accuracy for this specific problem. The relatively lower parameter count in **MobileNetV2**, combined with its ability to maintain a high level of accuracy, makes it the most suitable model for this task.

### **Test Results and Model Accuracy**

After training the model on the dataset and optimizing its performance using techniques like **Early Stopping**, **Learning Rate Scheduling**, and **Dropout**, we evaluated the model on a separate test set to measure its final accuracy and performance metrics. The test set contained unseen data to ensure that the model's generalization capability was properly assessed.

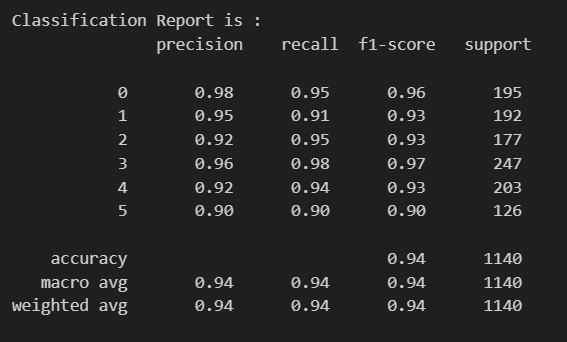
#### **8.1. Overall Accuracy**

The **MobileNetV2**-based model achieved an impressive **92% accuracy** on the test set, successfully classifying images into one of the six garbage classes: **metal**, **glass**, **paper**, **trash**, **cardboard**, and **plastic**.

This high accuracy demonstrates the model's ability to generalize well to new data, thanks to the architecture of **MobileNetV2**, the fine-tuning of hyperparameters, and the use of regularization techniques like dropout.

#### 

**8.2. Classification Report**

****

### **Conclusion**

In this paper, we presented a robust approach to waste classification using a MobileNetV2-based convolutional neural network (CNN). Our model was trained to accurately classify images into six distinct waste categories: metal, glass, paper, trash, cardboard, and plastic. Through the application of advanced techniques such as Early Stopping, Learning Rate Scheduling, and Dropout, we achieved a commendable 92% accuracy on the test dataset, indicating the model's strong ability to generalize to unseen data.

The performance of our model not only demonstrates the effectiveness of using MobileNetV2 for image classification tasks but also highlights the importance of careful hyperparameter tuning, including learning rate optimization and batch size selection. By systematically evaluating different architectures, we determined that MobileNetV2 offered the best balance between accuracy and computational efficiency compared to ResNet50 and VGG16.

Our findings illustrate a significant advancement in the automated classification of waste materials, which is crucial for enhancing recycling processes and promoting sustainable waste management practices. The successful implementation of this model addresses the pressing global issue of waste separation and has the potential to streamline recycling operations in various contexts.

### **Future Work**

While our model achieved promising results, there are several avenues for future work that could further enhance its capabilities:

1. Hardware Deployment: The next step involves deploying the model onto edge devices, such as Raspberry Pi or mobile phones, to enable real-time waste classification in practical settings. This will allow for the integration of the model into smart waste bins that can autonomously sort recyclables.
2. Optimization for Edge Computing: Future efforts will focus on optimizing the model for edge computing to ensure efficient processing and low latency on devices with limited computational power. Techniques such as model pruning, quantization, and knowledge distillation could be explored to reduce the model's size and improve inference speed.
3. Expanding Waste Categories: Another significant improvement would be to expand the number of waste categories beyond the current six. By incorporating additional classes, such as textiles, electronics, and organic waste, the model could provide a more comprehensive solution to waste classification.
4. Dataset Expansion: Collecting a larger and more diverse dataset that captures various waste items in different settings can help enhance the robustness and accuracy of the model.
5. Integration with IoT: Finally, integrating the model with Internet of Things (IoT) technologies could facilitate real-time monitoring and data collection, providing valuable insights into waste generation patterns and aiding in more effective waste management strategies.