Garbage Classification

Alexandre Cotorobai
Complementos de Aprendizagem Automática 24/25
MEI, DETI
University of Aveiro
Aveiro, Portugal
alexandrecotorobai@ua.pt

André Oliveira
Complementos de Aprendizagem Automática 24/25
MEI, DETI
University of Aveiro
Aveiro, Portugal
andreaoliveira@ua.pt

***Abstract*—This project explores image-based garbage classification using a dataset categorized into six distinct classes: cardboard, glass, metal, paper, plastic, and trash. We employed various machine learning and deep learning techniques to develop models capable of accurately classifying these waste materials. Multiple approaches were evaluated, including different optimization strategies such as Adam Momentum Optimizer and regularization techniques (L1, L2), alongside advanced neural network architectures like CNN, DenseNet, ResNet50, and VGG16. We also implemented traditional machine learning algorithms such as KNN for comparison.**

Performance enhancement methods including Dropout and EarlyStopping were systematically applied to evaluate their impact on model performance. The resulting models were extensively analyzed through accuracy metrics, ROC curves, and confusion matrices to assess their classification capabilities. This work has implications for automated waste sorting applications, recycling optimization, and environmental sustainability efforts through improved waste management systems.

***Index Terms*—Garbage classification, image recognition, deep learning, convolutional neural networks, transfer learning, VGG16, ResNet50, DenseNet, machine learning, regularization, waste sorting**

I. Introduction

The rapid advancement of machine learning and deep learning technologies has enabled systems to perform increasingly complex tasks that were once considered challenging for automated systems. Among these tasks, image classification of waste materials holds significant importance, especially in applications related to recycling, environmental management, and sustainable waste disposal.

In this project, we utilize the Garbage Classification dataset to develop and evaluate models for waste material classification. This dataset comprises images categorized into six distinct classes: cardboard (393 samples), glass (491 samples), metal (400 samples), paper (584 samples), plastic (472 samples), and trash (127 samples). Each category presents unique visual features and patterns, posing challenges for classification algorithms to effectively differentiate between them.

The primary objective of this project is to leverage state-of-the-art machine learning and deep learning techniques to create models capable of accurately classifying these waste materials into their respective categories. We explore various methodologies, including traditional machine learning algorithms like KNN [1], [2] and advanced deep learning architectures such as CNN [3], DenseNet [4], ResNet50 [5], and VGG16 [6], which have demonstrated remarkable success in image recognition tasks.

Our approach involves several key steps: data preprocessing, model selection, training with various optimization strategies, evaluation, and comparison. We implement different techniques including Adam Momentum Optimizer [7], L1 and L2 regularization [8], Dropout [9], and EarlyStopping [10] to enhance model performance. Data preprocessing includes tasks such as resizing images, normalization, and data augmentation [11] to enhance the models' generalization capabilities. For model selection, we experiment with different architectures and hyperparameters to identify the most effective configuration.

Evaluation metrics such as accuracy, ROC curves, and confusion matrices are used to comprehensively assess the performance of each model. We analyze the evolution of training and validation accuracy across epochs and compare final training/validation accuracies with test accuracies to evaluate generalization capabilities.

The insights gained from our experiments have potential applications in automated recycling systems, waste management facilities, and environmental conservation efforts.

II. State of the Art

A. Traditional Machine Learning Approaches

Traditional machine learning algorithms have been widely applied to image classification tasks, including waste material classification. Support Vector Machines (SVM) [12] and K-Nearest Neighbors (KNN) [1] are among the established methods used in this domain. These approaches have proven effective for certain applications but often require manual feature extraction, which can be labor-intensive and may not capture the intricate patterns present in waste material images.

KNN, in particular, classifies images based on similarity measures with labeled training examples. While simple to implement, its performance can degrade with high-dimensional data typical of image classification tasks and requires careful tuning of the neighborhood size parameter [2].

B. Convolutional Neural Networks (CNNs)

The advent of Convolutional Neural Networks (CNNs) has significantly advanced the field of image classification, including waste categorization. CNNs automatically learn hierarchical feature representations from raw images, reducing the need for manual feature engineering. Basic CNN architectures typically consist of convolutional layers followed by pooling layers and fully connected layers, providing an end-to-end learning framework for image classification [3].

C. Advanced CNN Architectures

More recent advancements have focused on developing increasingly sophisticated CNN architectures that offer improved efficiency and accuracy. Models such as ResNet, DenseNet, and VGG16 represent significant innovations in deep learning approaches for image classification.

D. ResNet (Residual Networks)

ResNet, introduced by He et al. [5], revolutionized deep learning with the concept of residual connections. These skip connections address the vanishing gradient problem that plagued very deep networks, allowing models to effectively learn through hundreds of layers. ResNet50, with 50 layers, provides a good balance between depth and computational efficiency, making it suitable for complex classification tasks like waste material categorization.

E. DenseNet (Densely Connected Networks)

DenseNet, another groundbreaking architecture developed by Huang et al. [4], connects each layer to every other layer in a feed-forward fashion. This dense connectivity alleviates the vanishing gradient problem and promotes feature reuse, leading to more efficient and accurate models. In the context of waste classification, DenseNet has shown promising results due to its ability to capture intricate patterns and details across diverse waste categories.

F. VGG16

VGG16, developed by the Visual Geometry Group at Oxford [6], features a straightforward architecture with sequential convolutional layers using small 3×3 filters. Despite its relative simplicity compared to newer architectures, VGG16 remains relevant due to its strong feature extraction capabilities and has been successfully applied to various image classification tasks, including waste classification.

G. Transfer Learning and Fine-Tuning

Transfer learning has become a crucial technique in deep learning, especially when dealing with limited datasets. Models pre-trained on large datasets like ImageNet can be fine-tuned on specific tasks, achieving high accuracy with less computational resources and time. [13] explored the effectiveness of transfer learning for various visual recognition tasks, highlighting the versatility and efficiency of this approach.

In our project, we leverage pre-trained models such as VGG16, ResNet50, and DenseNet, fine-tuning them on the Garbage Classification dataset to achieve optimal performance. The pre-trained weights provide a strong starting point, capturing general features that can be adapted to our specific waste classification task.

Transfer learning involves using the convolutional layers from pre-trained networks while replacing and retraining the fully connected layers for our specific classification task. This approach is particularly effective when the target task shares visual similarities with the original training data, as is the case with waste material classification.

H. Data Augmentation

Data augmentation and regularization are essential techniques to enhance the generalization capabilities of deep learning models. Techniques such as random cropping, flipping, and color jittering, as discussed by [14], have proven effective in increasing the diversity of training data and preventing overfitting.

I. Optimization Techniques

Several optimization strategies have proven effective in enhancing model performance:

- **Adam Momentum Optimizer:** Combines the benefits of Adam (adaptive learning rates) and momentum optimization, facilitating faster convergence and better performance in deep learning models [7].
- **Regularization Techniques:** L1 and L2 regularization help prevent overfitting by penalizing large weights. L1 promotes sparsity, while L2 prevents any single weight from becoming excessively influential [8].
- **Dropout:** Randomly deactivates neurons during training, forcing the network to learn more robust features and reducing co-adaptation between neurons [9].
- **Early Stopping:** Monitors validation performance and halts training when performance begins to degrade, preventing overfitting and saving computational resources [10].

J. Relevant research on garbage classification

Recent research in waste classification has demonstrated the effectiveness of deep learning approaches. Yang and Thung [15] achieved 87% accuracy using CNNs for classifying waste into recycling categories.

Bircanoglu et al. [16] proposed a dedicated architecture named *RecycleNet*, specifically optimized for waste classification. Their work evaluated several state-of-the-art convolutional neural networks including ResNet50, MobileNet, Inception-ResNet, DenseNet, and Xception, both from scratch and via transfer learning. Among their experiments, transfer learning with DenseNet121 achieved the best result with 95% test accuracy (fine-tunned), training the model from scratch achieved the result of 85% test accuracy . They also introduced RecycleNet, a modified DenseNet-based architecture that significantly reduced the number of parameters (from 7 million to about 3 million) and improved real-time prediction speed, achieving 81% test accuracy. This balance between

performance and computational efficiency makes RecycleNet suitable for real-world applications such as smart bins or industrial sorting systems.

Sakr et al. [17] employed transfer learning with pre-trained CNN models achieving over 90% accuracy on waste classification tasks, while Aral et al. [18] developed an ensemble approach combining multiple CNN architectures to improve robustness across diverse waste materials.

Current challenges in the field include dealing with class imbalance (as seen in our dataset with fewer trash samples), handling varying lighting conditions and backgrounds in waste images, and distinguishing between visually similar waste categories (e.g., certain types of plastic and glass) [19].

We also explored publicly available implementations on Kaggle to benchmark our results against other community-driven approaches. Several notebooks utilizing the same garbage classification dataset were analyzed, employing a variety of deep learning techniques ranging from custom CNN architectures to advanced transfer learning models like EfficientNet and MobileNetV2. The performance metrics reported in these notebooks are summarized in Table I, providing insight into the effectiveness of different model choices and configurations used in practice.

TABLE I
SUMMARY OF KAGGLE NOTEBOOK MODELS AND THEIR REPORTED TEST ACCURACIES

Method	Test Accuracy
CNN	77.0%
ResNet50 (transfer learning)	86.2%
DenseNet121 (transfer learning)	83.3%
VGG16 (transfer learning)	88.0%
MobileNetV2 (pre-trained)	82.0%
EfficientNet	83.0%

Taking into account these advancements this project aims to systematically evaluate different architectures and optimization techniques to push the boundaries of automated waste classification accuracy.

III. DATASET ANALYSIS

A. Dataset Description

The **Garbage Classification** dataset was selected to perform the task of identifying different waste materials. It contains images categorized into six distinct classes, with the following number of samples:

- Cardboard: 393 images
- Glass: 491 images
- Metal: 400 images
- Paper: 584 images
- Plastic: 472 images
- Trash: 127 images

In total, the dataset comprises **2,467 images** across these six categories. Each image contains a single waste item photographed from various angles and in different lighting conditions, providing diversity in the visual representation of each class [20].

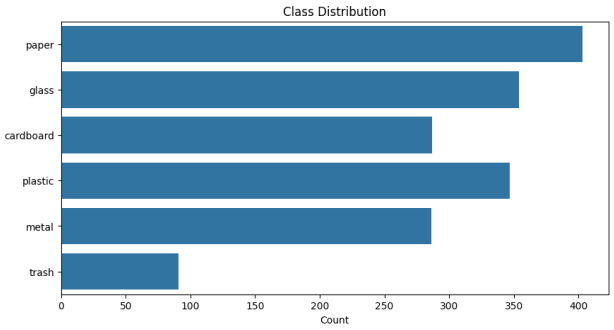

Fig. 1. Data distribution into 6 classes.

As shown in Fig. 1, the dataset is imbalanced, with the *trash* category having significantly fewer samples (127) compared to *paper* (584). This imbalance introduces challenges for model training, as classifiers may develop biases toward the more represented categories. The proportional representation of each class is visualized in Fig. 2.

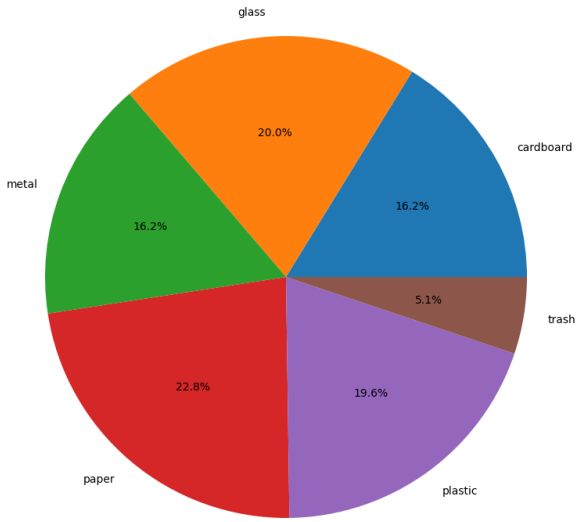

Fig. 2. Distribution Chart of training data.

B. Visual Characteristics

Each class exhibits distinct visual characteristics that models must learn to differentiate:

- **Cardboard**: Generally brown, with visible texture and often rectangular or box-like shapes
- **Glass**: Transparent or translucent, reflective surfaces, often bottles or containers
- **Metal**: Reflective, often cans or metallic objects with distinctive shapes
- **Paper**: Various colors but typically flat, sometimes with text or creases

- **Plastic**: Various colors, often containers, packaging, or bottles with smooth surfaces
- **Trash**: Most diverse category, containing miscellaneous items not fitting other classes

The diversity within classes (e.g., shape, color, and orientation) and similarities between some (e.g., certain plastic containers resembling glass) increase classification complexity [21]. Fig. 3 shows representative examples from each class.

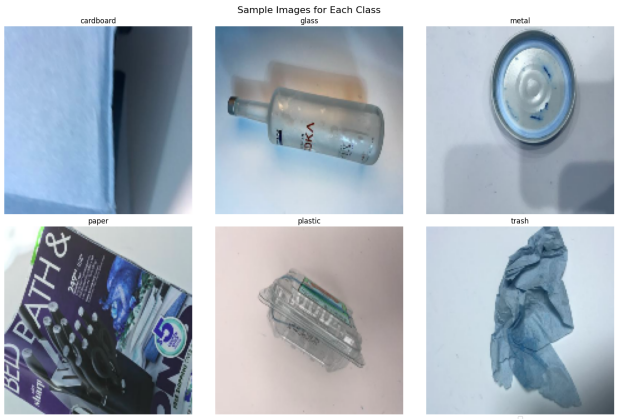

Fig. 3. Examples of each class.

C. Data Preprocessing

Prior to model training, the dataset underwent preprocessing using common image-based learning techniques [11]:

- **Resizing**: All images were resized to a uniform dimension (typically 224×224 pixels for compatibility with pretrained models like VGG16 and ResNet50)
- **Normalization**: Pixel values were scaled to the range [0,1] or standardized with mean subtraction and standard deviation division, depending on the model requirements
- **Data Augmentation**: To address class imbalance and enhance model generalization, we applied various transformations including:
  - Random rotations
  - Horizontal and vertical flips
  - Brightness and contrast adjustments
  - Small zooming and shifting

D. Train/Validation/Test Split

The dataset was already divided into three subsets: the training set, used for model training; the validation set, used for hyperparameter tuning and early stopping decisions; and the test set used for final model evaluation. Fig. 4 visualizes the partitioning of the dataset.

IV. ML Models

All models used in this study were implemented with the goal of accurately classifying waste materials into the six predefined categories. We employed both traditional machine learning approaches and deep learning architectures, each with specific optimization techniques.

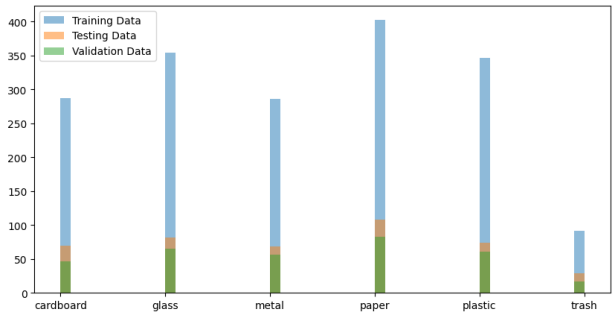

Fig. 4. Distribution of training, testing and validation.

A. KNN (K-Nearest Neighbors)

KNN is a non-parametric, supervised learning algorithm that classifies images based on the majority class of their k nearest neighbors in the feature space [1], [2]. For our base implementation:

- Images were flattened from their original 3D representation (height × width × channels) to a 1D vector to serve as input features
- We utilized the Euclidean distance metric to measure similarity between samples
- The neighborhood size ($k = 6$) was used in this experimentation
- Class weights were adjusted to address the imbalance in the dataset

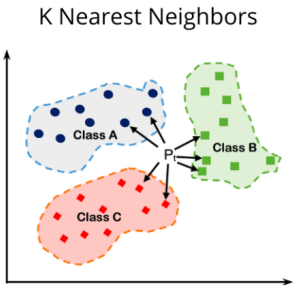

Fig. 5. Illustration of KNN classification. A sample point P is classified based on the majority class of its k nearest neighbors. Here, depending on the value of k, P would be classified as either Class B or Class C.

Figure 5 illustrates the KNN principle, where a new data point (P) is classified based on the majority class of its nearest neighboring points in the feature space. The classification result can vary depending on the chosen value of k.

While conceptually simple, KNN provides a useful baseline for comparison with more complex models. However, it typically struggles with high-dimensional image data and can be computationally expensive during inference with large datasets [22].

B. CNN (Convolutional Neural Network)

We implemented a custom CNN architecture specifically designed for waste classification. Before diving into our implementation details, it's important to understand the fundamental

principles of CNNs and how they differ from traditional neural networks.

A traditional neural network, as illustrated in Figure 6, consists of an input layer, hidden layers, and an output layer. Each neuron in a layer is connected to every neuron in the subsequent layer, creating a fully connected structure. While effective for many tasks, this architecture becomes computationally prohibitive for image data. For example, with an input image of size $150 \times 150 \times 3$ (height, width, channels), a neural network would require 67,500 input neurons, with each connecting to every neuron in the next layer. This results in millions of parameters, making the network prone to overfitting and computationally intensive to train.

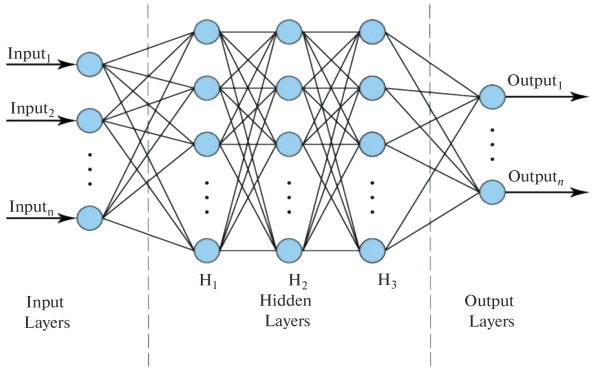

Fig. 6. Structure of a traditional Neural Network with input layer, hidden layers, and output layer. Each neuron connects to all neurons in adjacent layers.

Convolutional Neural Networks address these limitations through specialized layers designed for processing grid-like data such as images. Figure 7 shows the typical architecture of a CNN, which includes:

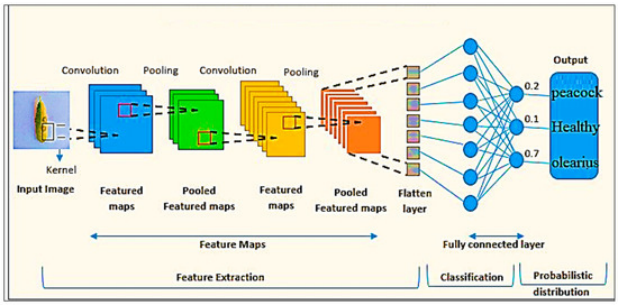

Fig. 7. A typical CNN architecture showing the sequence of convolutional layers, pooling layers, and fully connected layers for image classification.

- **Convolutional Layers**: These layers apply convolution operations to the input, using filters (kernels) to detect features like edges, textures, and patterns. Each filter slides across the input image, computing element-wise multiplications and summations to produce feature maps. This operation allows CNNs to capture spatial hierarchies in images while significantly reducing the number of parameters compared to fully connected networks.
- **Pooling Layers**: Following convolution, pooling layers (typically max pooling) downsample the feature maps, reducing their spatial dimensions while preserving important information. This helps achieve spatial invariance and further reduces computational requirements.
- **Flatten Layer**: After multiple convolutional and pooling operations, the resulting 3D feature maps are flattened into a 1D vector, which serves as input to fully connected layers.
- **Fully Connected Layers**: These layers perform classification based on the features extracted by convolutional layers, mapping them to output classes.

Our CNN implementation for waste classification consists of [3], [23]:

- Three convolutional blocks, each with:
  - Convolutional layers with 3×3 filters (32, 64, and 128 filters respectively)
  - ReLU activation functions
  - Max pooling layers (2×2) to reduce spatial dimensions
- A flatten layer to convert 3D feature maps to 1D feature vectors
- A dense layer with 128 units and ReLU activation
- Dropout layer (0.5) for regularization
- Output layer with 6 units and softmax activation for multi-class classification

The network was trained using various optimization techniques including Adam Momentum Optimizer [7], L1 and L2 regularization [8], Dropout [9], and EarlyStopping, both individually and in combination.

C. ResNet50 (Residual Network)

ResNet50 is a deep residual network with 50 layers that introduces skip connections to address the vanishing gradient problem in very deep networks [5].

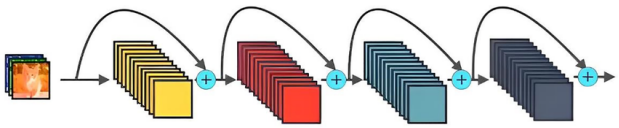

Fig. 8. Illustration of residual connections in ResNet architecture, allowing information to flow directly across layers.

As shown in Figure 8, these skip connections allow information to bypass one or more layers, addressing the vanishing gradient problem that typically plagues very deep networks. Rather than learning direct transformations, each block learns residual functions (differences) that are easier to optimize. This elegant solution enables the effective training of much deeper networks than was previously possible.

Our base implementation:

- Used transfer learning with weights pre-trained on ImageNet
- Added a Global Average Pooling layer after the base model

- Added a dense layer with 128 units and ReLU activation
- Implemented Dropout (0.5) for regularization
- Added an output layer with 6 units and softmax activation
- Initially froze the pre-trained layers to preserve their learned features
- Used Adam optimizer with categorical cross-entropy loss

ResNet50's depth enables it to learn complex features, while its residual connections allow gradients to flow more effectively during backpropagation, potentially leading to better performance on complex classification tasks [24].

D. DenseNet (Densely Connected Network)

DenseNet introduces dense connections where each layer receives input from all preceding layers and passes its feature maps to all subsequent layers [4].

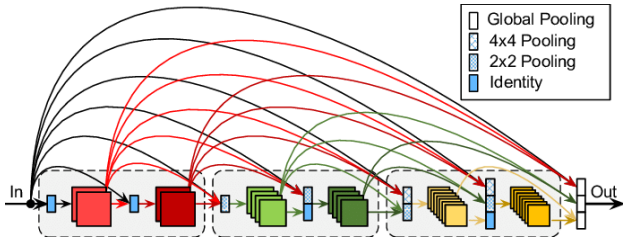

Fig. 9. DenseNet architecture showing the dense connectivity pattern where each layer receives input from all preceding layers and passes its features to all subsequent layers.

As shown in Figure 9, DenseNet's key innovation is its dense connectivity pattern, where each layer receives feature maps from all preceding layers and contributes its own feature maps to all subsequent layers. These connections (represented by the curved lines) create direct paths for information and gradient flow throughout the network. Unlike traditional architectures where information passes sequentially, DenseNet establishes dense connections that enable extensive feature reuse.

This architecture:

- Utilized transfer learning with ImageNet pre-trained weights
- Added a Global Average Pooling layer to reduce spatial dimensions
- Included a dense layer with 128 units and ReLU activation
- Applied Dropout (0.5) for regularization
- Added an output layer with 6 units and softmax activation
- Initially froze the pre-trained layers during first phase of training
- Used Adam optimizer with categorical cross-entropy loss

The dense connectivity pattern in DenseNet encourages feature reuse, reduces the number of parameters compared to similar-performing models, and strengthens feature propagation, making it particularly effective for classification tasks with limited training data [25].

E. VGG16 (Visual Geometry Group)

VGG16 is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other with increasing depth [6].

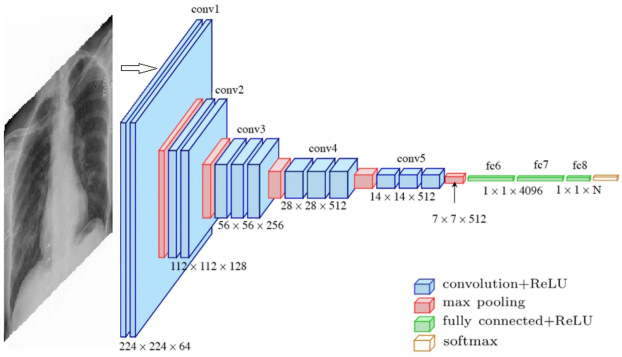

Fig. 10. VGG16 architecture showing the sequential arrangement of convolutional layers (blue), max pooling layers (red), and fully connected layers (green) with their respective dimensions.

As illustrated in Figure 10, the network follows a straightforward pattern of stacked convolutional layers using small 3×3 filters, each followed by ReLU activation. The spatial dimensions are progressively reduced through max pooling layers while the number of feature channels increases from 64 to 512. This design creates a deep feature hierarchy that effectively captures visual patterns at multiple scales.

Our base implementation:

- Utilized transfer learning with pre-trained ImageNet weights
- Added a Global Average Pooling layer after the base model
- Included a dense layer with 128 units and ReLU activation
- Applied Dropout (0.5) for regularization
- Added an output layer with 6 units and softmax activation
- Initially froze the pre-trained layers to preserve learned features
- Used Adam optimizer with categorical cross-entropy loss

VGG16's uniform architecture and effective feature extraction capabilities make it a strong contender despite being an older architecture compared to ResNet and DenseNet [26].

F. Optimization Techniques

For each model, we systematically evaluated the following optimization strategies:

- **Adam Momentum Optimizer**: Combines adaptive learning rates with momentum to accelerate convergence and navigate saddle points more effectively [7], [27]
- **L2 Regularization**: Adds a penalty term proportional to the square of weight magnitudes, preventing any single weight from becoming too large [8], [28]
- **L1 Regularization**: Adds a penalty term proportional to the absolute value of weights, promoting sparsity in the model [8], [28]

- **Dropout**: Randomly deactivates neurons during training with a specified probability, reducing co-adaptation and promoting more robust feature learning [9]
- **EarlyStopping**: Monitors validation metrics and stops training when performance plateaus or begins to degrade, preventing overfitting [10], [29]
- **Combined Dropout + EarlyStopping**: Implements both strategies simultaneously to maximize regularization benefits while optimizing training duration [30]

These optimization techniques were selected based on their established effectiveness in improving model generalization and addressing common challenges in image classification tasks, particularly with imbalanced datasets like our waste classification images [31], [32].

G. Hyperparameter Tuning

To further enhance model performance, we conducted hyperparameter tuning on our custom CNN architecture. The tuning process explored different configurations for key parameters such as learning rate, batch size, dropout rate, and the number of units in the dense layer. We applied a randomized hyperparameter search using Keras Tuner, where the model's performance was validated on a dedicated validation set. This process sampled combinations of architectural and optimization hyperparameters to identify the best configuration for our CNN.

We also intended to apply hyperparameter tuning to the DenseNet model to maximize its performance. However, due to computational resource limitations—particularly the extended training times and high memory demands of DenseNet—we were unable to carry out a full hyperparameter optimization for this architecture within the available timeframe. Future work could revisit this tuning process with more robust computational resources to explore further improvements.

V. Results

A. CNN (Convolutional Neural Network)

The CNN model was evaluated using different optimization strategies to determine the most effective approach for the waste classification task. We systematically tested six different optimization techniques: Adam optimizer (baseline), L1 regularization, L2 regularization, Dropout, EarlyStopping, and a combined Dropout+EarlyStopping approach.

*1) **Training and Validation Performance**:* Fig. 11 shows the training and validation performance during model training for our baseline CNN model with Adam optimizer. The left plot displays the accuracy curves, while the right plot shows the loss evolution over epochs.

The accuracy plot shows a steady increase in both training and validation accuracy over the epochs, with the final training accuracy reaching approximately 73% and validation accuracy around 61%. The loss curves show consistent decrease throughout training, indicating effective learning. The divergence between training and validation curves toward later

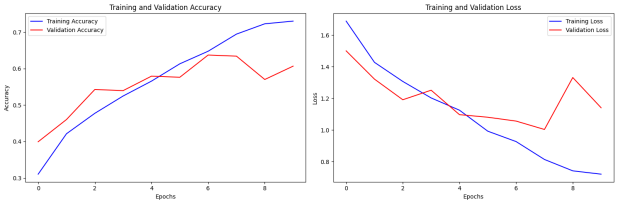

Fig. 11. CNN Adam Optimizer Training and Validation Accuracy/Loss

epochs suggests some degree of overfitting, which motivated our exploration of regularization techniques.

*2) **Model Evaluation**:* Fig. 12 presents the ROC curves and confusion matrix for the baseline CNN model.

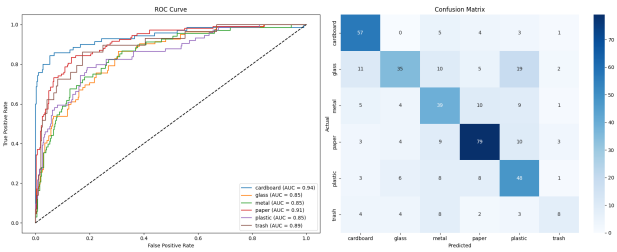

Fig. 12. ROC curves and confusion matrix for the CNN model on test data.

The ROC curves demonstrate strong classification performance across all waste categories, with Area Under the Curve (AUC) values ranging from 0.85 to 0.94. Particularly impressive are the AUC values for cardboard (0.94) and paper (0.91), indicating excellent classification capability for these categories. The trash category, despite having the fewest training examples, achieved a respectable AUC of 0.89, suggesting that the model was able to learn distinguishing features despite the class imbalance.

The confusion matrix reveals that paper had the highest number of correct classifications (79 samples), while glass and trash categories presented more challenges with relatively more misclassifications. Notable confusion patterns include:

- Glass being misclassified as plastic (19 samples)
- Metal being confused with paper (10 samples)
- Several instances of paper being mistaken for plastic (10 samples)

*3) **Optimization Techniques Comparison**:* To address the overfitting observed in the baseline model, we evaluated several optimization techniques. The following sections present the detailed results for each technique.

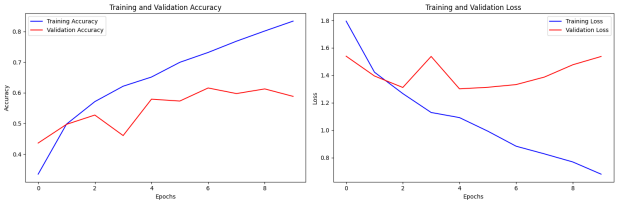

Fig. 13. CNN with L2 Regularization Training and Validation Accuracy/Loss

a) L2 Regularization: Fig. 13 shows the training and validation accuracy/loss curves for the CNN model with L2 regularization. The training accuracy increases steadily, reaching approximately 84% by the end of training, while the validation accuracy shows more fluctuation, stabilizing around 60%. The validation accuracy exhibits an initial increase followed by fluctuations, which is characteristic of the regularization effect preventing the model from simply memorizing the training data.

The loss curves show a consistent decrease in training loss, while validation loss initially decreases but then shows some instability with fluctuations between epochs 2 and 4, followed by a gradual increase in later epochs. This pattern suggests that the model might benefit from early stopping to prevent eventual overfitting.

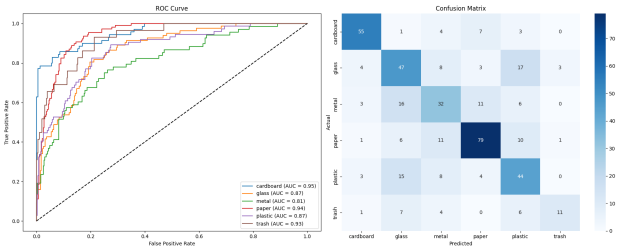

Fig. 14. ROC curves and confusion matrix for the CNN model with L2 regularization.

Fig. 14 presents the ROC curves and confusion matrix for the CNN model with L2 regularization.

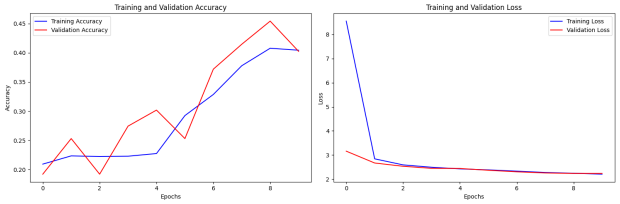

Fig. 15. CNN with L1 Regularization Training and Validation Accuracy/Loss

b) L1 Regularization: Fig. 15 shows the training and validation accuracy/loss curves for the CNN model with L1 regularization. The training accuracy increases gradually, reaching approximately 40% by the end of training, while the validation accuracy shows significant fluctuation, peaking at around 45% at epoch 8. The validation accuracy exhibits an irregular pattern with several peaks and valleys, indicating the sparse feature selection characteristic of L1 regularization.

The loss curves show a substantial initial drop in training loss, followed by a gradual decrease throughout the remaining epochs. Validation loss closely follows the training loss pattern, with both curves converging to approximately 2.2 by the end of training, suggesting minimal overfitting despite the fluctuations in validation accuracy.

Fig. 16 presents the ROC curves and confusion matrix for the CNN model with L1 regularization.

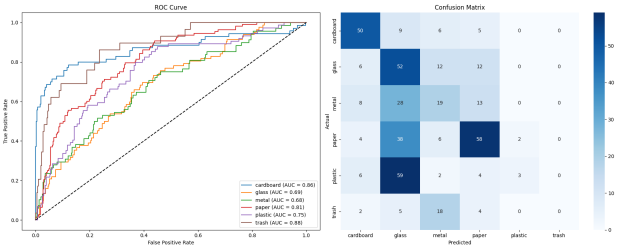

Fig. 16. ROC curves and confusion matrix for the CNN model with L1 regularization.

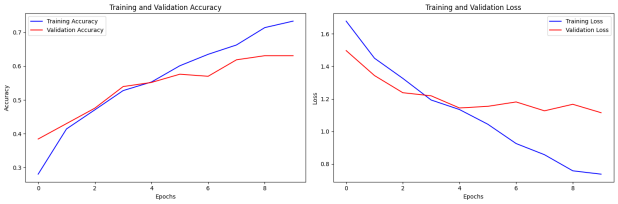

Fig. 17. CNN with Dropout Training and Validation Accuracy/Loss

c) Dropout: Fig. 17 shows the training and validation accuracy/loss curves for the CNN model with Dropout regularization. The training accuracy increases steadily, reaching approximately 73% by the end of training, while the validation accuracy shows a more consistent upward trend compared to other regularization techniques, stabilizing around 63%. The validation accuracy follows the training accuracy closely until epoch 4, after which a growing gap appears, indicating the regularization effect of dropout preventing overfitting.

The loss curves show a consistent decrease in training loss throughout the epochs, reaching approximately 0.75 by the end. Validation loss decreases initially but begins to plateau around epoch 4, stabilizing at around 1.1 for the remainder of training. The divergence between training and validation loss curves suggests that dropout is effectively preventing the model from overfitting to the training data while maintaining good generalization performance.

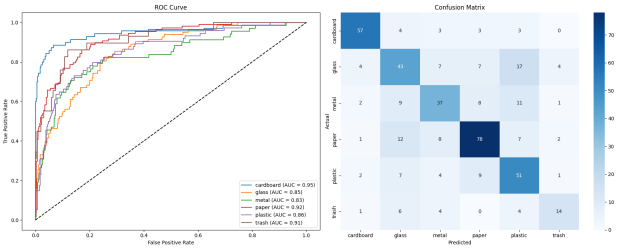

Fig. 18. ROC curves and confusion matrix for the CNN model with Dropout.

Fig. 18 presents the ROC curves and confusion matrix for the CNN model with Dropout.

d) Early Stopping: Fig. 19 shows the training and validation accuracy/loss curves for the CNN model with Early Stopping. The training accuracy increases steadily, reaching approximately 85% by the end of training, while the validation

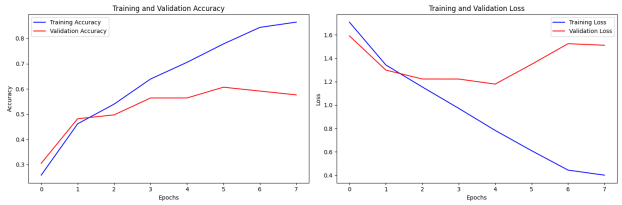

Fig. 19. CNN with Early Stopping Training and Validation Accuracy/Loss

accuracy shows an initial increase followed by a plateau around epoch 3, eventually reaching a peak of about 60% at epoch 5 before slightly declining. The validation accuracy demonstrates the effectiveness of early stopping in identifying the optimal point to halt training before overfitting becomes severe.

The loss curves show a consistent decrease in training loss throughout the epochs, reaching approximately 0.4 by the final epoch. Validation loss follows an interesting pattern, initially decreasing until epoch 3, then remaining relatively stable before increasing after epoch 4, eventually settling around 1.5. This divergence between training and validation loss curves after epoch 4 clearly indicates the onset of overfitting, validating the necessity of the early stopping mechanism.

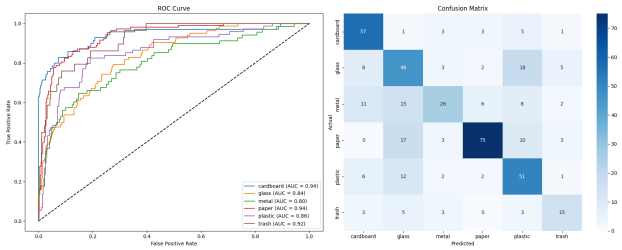

Fig. 20. ROC curves and confusion matrix for the CNN model with Early Stopping.

Fig. 20 presents the ROC curves and confusion matrix for the CNN model with Early Stopping.

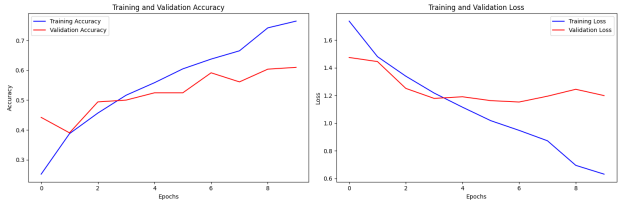

Fig. 21. CNN with Dropout and Early Stopping Training and Validation Accuracy/Loss

e) Dropout + Early Stopping: Fig. 21 shows the training and validation accuracy/loss curves for the CNN model with combined Dropout and Early Stopping. The training accuracy increases steadily, reaching approximately 75% by the end of training, while the validation accuracy shows a more gradual improvement with some fluctuations, peaking at about 60% around epoch 6 before slightly decreasing. The early phases show validation accuracy initially outperforming training accuracy until epoch 2, after which the expected pattern emerges.

The loss curves demonstrate a consistent decrease in training loss throughout the epochs, reaching approximately 0.65 by the final epoch. Validation loss follows a less consistent pattern, initially decreasing until around epoch 3, then stabilizing with minor fluctuations around 1.2 for the remainder of training. The relatively stable validation loss after epoch 3, despite continued training loss reduction, indicates the combined regularization effect successfully preventing overfitting.

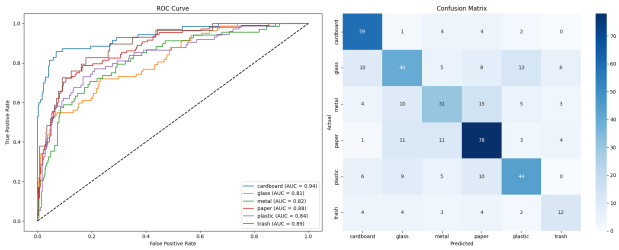

Fig. 22. ROC curves and confusion matrix for the CNN model with Dropout and Early Stopping.

Fig. 22 presents the ROC curves and confusion matrix for the CNN model with Dropout and Early Stopping.

f) HyperParameter Tuning: We explored the following hyperparameters during RandomSearch (max_trials=10) aiming to maximize validation accuracy:

- **Convolutional blocks:** number of layers {2,3,4}
- **Per-layer Conv2D:**
  - filters {32,64,96,128}
  - kernel size {3,5}
  - activation {ReLU, Tanh}
- **Dense block:**
  - units {64,128,192,256}
  - activation {ReLU, Tanh}
  - dropout rate {0.2,0.3,0.4,0.5}
- **Optimizer & LR:** optimizer {Adam, RMSprop, SGD}, learning rate {1e-2,1e-3,1e-4}

The best configuration found was:

- **Convolutional Layers:** 3 filters = [32, 32, 96], kernel sizes = [5, 3, 5], activations = [Tanh, Tanh, ReLU]
- **Dense Block:** units = 192, activation = Tanh, dropout = 0.2
- **Optimizer & LR:** Adam with learning rate = 1×10^{-3}

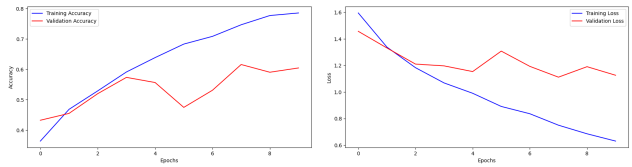

Fig. 23. Training and validation accuracy (left) and training and validation loss (right) across epochs for the CNN model with Hyper Parameter Tuning.

Fig. 23 illustrates the training and validation accuracy/loss curves for the CNN model optimized through hyperparameter

tuning. The training accuracy shows a steady improvement across epochs, reaching approximately 77% by the final epoch. Validation accuracy initially mirrors the training trend, peaking near 60% around epoch 3 before experiencing some oscillations, likely due to the model's sensitivity to the tuned parameters.

The loss curves reflect a smooth and continuous decrease in training loss, dropping from around 1.6 to below 0.7 by the end of training. Validation loss decreases in the early epochs, stabilizing between 1.0 and 1.2, with minor fluctuations toward the later stages. This pattern suggests that the hyperparameter tuning successfully optimized the model's learning dynamics, achieving stronger initial performance while keeping overfitting under control.

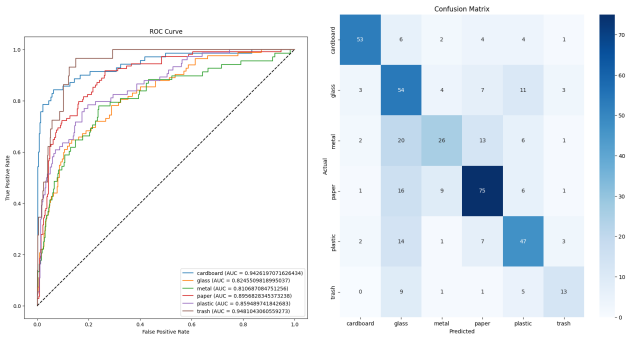

Fig. 24. ROC curves (left) and confusion matrix (right) for the CNN model with Hyper Parameter Tuning.

Fig. 24 presents the ROC curves and confusion matrix for the CNN model with Hyper Parameter Tuning.

TABLE II
CNN ACCURACY RESULTS FOR OPTIMIZATION TECHNIQUES

Technique	Train Acc.	Val Acc.	Test Acc.	$\|Tr - Ts\|$
Adam	0.730	0.607	0.617	0.113
L2	0.834	0.588	0.622	0.212
L1	0.404	0.402	**0.422**	**0.018**
Dropout	0.734	0.631	0.650	0.084
EarlyStop	0.865	0.576	0.626	0.238
DO + ES	0.765	0.610	0.613	0.152
Hyper	0.771	0.605	0.622	0.149

*4) **CNN Summary**:* Table II summarizes the accuracy performance of the CNN model under each tested optimization technique. The table clearly shows the variation in generalization capability across different regularization strategies, with Dropout and the combination of Dropout + EarlyStopping providing strong balance between training and test performance. While EarlyStopping yielded the highest training accuracy, it also exhibited the largest drop in performance from training to test, indicating potential overfitting. On the other hand, L1 regularization achieved minimal overfitting but at the cost of significantly lower overall accuracy.

We performed a similar detailed evaluation for all other implemented machine learning and deep learning models. However, to maintain brevity and clarity, in the following sections we present only the final accuracy comparison tables for each model, and we discuss in detail only the best-performing optimization strategy for each case.

B. DenseNet

Table III presents the training, validation, and test accuracies obtained using different optimization techniques with the DenseNet architecture. Among these, the Adam optimizer achieved the highest test accuracy (82.37%) while maintaining a strong balance between training and test performance. Therefore, we selected this configuration for a more detailed discussion.

TABLE III
ACCURACY RESULTS FOR DENSENET WITH DIFFERENT OPTIMIZATION TECHNIQUES

Technique	Train Acc.	Val Acc.	Test Acc.	$\|Tr - Ts\|$
Adam	0.872	0.784	**0.824**	**0.048**
L2	0.998	0.784	0.807	0.191
L1	0.939	0.780	0.803	0.136
Dropout	0.881	0.787	0.817	0.064
EarlyStop	0.976	0.784	0.796	0.180
DO + ES	0.883	0.784	0.798	0.085

a) Adam Optimizer: The DenseNet model trained with the Adam optimizer demonstrated excellent performance across all metrics. It achieved a training accuracy of 87.16% and a validation accuracy of 78.35%, showing strong generalization capabilities. The test accuracy peaked at 82.37%, the highest among all configurations tested for this model. The relatively small difference of 4.8% between training and test accuracy suggests that the model did not suffer significantly from overfitting. These results confirm that the Adam optimizer, when combined with DenseNet's densely connected architecture, provides a robust solution for garbage classification tasks.

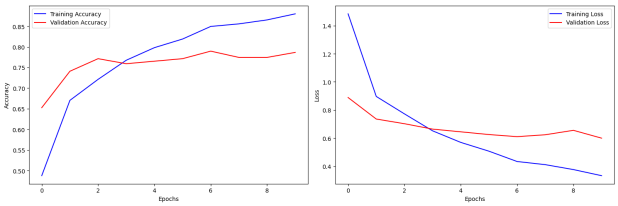

Fig. 25. DenseNet with Adam Optimizer: Training and Validation Accuracy/Loss

Fig. 25 shows the training and validation performance during model training. The model achieved a steady increase in training accuracy, reaching approximately 88% after 9 epochs, while validation accuracy stabilized around 79%. The close alignment between training and validation curves indicates good generalization with minimal overfitting.

Fig. 26 presents the ROC curves and confusion matrix for the DenseNet model on test data. The model achieved exceptional classification performance with AUC values ranging from 0.96 to 0.98 across all categories. The confusion matrix demonstrates strong classification accuracy, particularly for paper (100 correct classifications), glass (72), cardboard

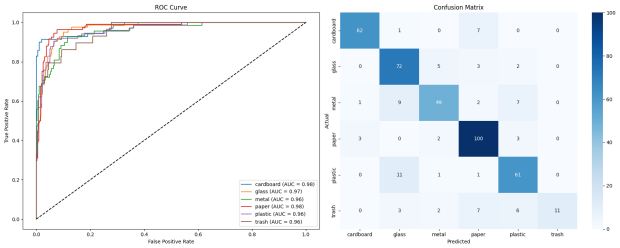

Fig. 26. ROC curves and confusion matrix for DenseNet with Adam optimizer

(62), and plastic (61). The trash category achieved the lowest accuracy with only 11 correct classifications, likely due to its heterogeneous nature and smaller representation in the training set.

DenseNet outperformed the CNN model across all waste categories, with substantial improvements in classification accuracy and AUC values. The model's ability to reuse features through dense connections likely contributed to its superior performance on this classification task.

b) Hyperparameter Tuning: Although hyperparameter tuning was planned for the DenseNet model to explore the impact of parameters such as learning rate, dropout rate, and dense layer size, we were unable to carry out these experiments due to computational limitations. As a result, the DenseNet results reported here rely on standard parameter configurations without further fine-tuning. Despite this, the model still achieved strong performance, suggesting that even without extensive tuning, DenseNet is a highly effective architecture for the garbage classification task. Future studies with greater computational resources could explore tuning to further enhance performance.

c) Best Result Summary Table: Table IV summarizes the performance of the best DenseNet model.

TABLE IV
CLASSIFICATION REPORT FOR DENSENET MODEL

Class	Precision	Recall	F1-Score	Support
Cardboard	0.94	0.89	0.91	70
Glass	0.75	0.88	0.81	82
Metal	0.83	0.72	0.77	68
Paper	0.83	0.93	0.88	108
Plastic	0.77	0.82	0.80	74
Trash	1.00	0.38	0.55	29
Accuracy			0.82	431
Macro Avg	0.85	0.77	0.79	431
Weighted Avg	0.83	0.82	0.82	431

C. ResNet50

Table V presents the accuracy results for ResNet50 using various optimization techniques. While overall accuracy values were lower compared to other architectures, the L1 Regularization technique produced the smallest overfitting gap (absolute difference between training and test accuracy), making it the most balanced and robust approach for this model.

TABLE V
ACCURACY RESULTS FOR RESNET50 WITH DIFFERENT OPTIMIZATION TECHNIQUES

Technique	Train Acc.	Val Acc.	Test Acc.	$\lvert Tr - Ts \rvert$
Adam	0.241	0.299	0.318	0.077
L2	0.329	0.253	0.295	0.034
L1	0.272	0.293	**0.290**	**0.018**
Dropout	0.287	0.308	0.329	0.042
EarlyStop	0.434	0.427	0.404	0.030
DO + ES	0.267	0.280	0.302	0.035

a) L1 Regularization: Despite its relatively modest absolute accuracy values, L1 Regularization proved to be the most stable and generalizable configuration for ResNet50. The training accuracy was 27.2%, while the test accuracy closely followed at 29.0%, resulting in an overfitting gap of only 1.8%. This minimal difference indicates that the model was not overfitting the training data and generalized its learning effectively. The sparsity induced by L1 regularization likely contributed to this behavior by enforcing a simpler and more robust model, which is especially valuable given ResNet50's depth and complexity.

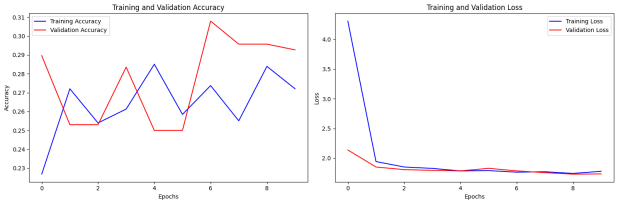

Fig. 27. ResNet50 with L1 Regularization: Training and Validation Accuracy/Loss

Fig. 27 shows the training and validation performance. Both metrics exhibited significant fluctuations, with validation accuracy peaking at approximately 31% around epoch 6. The loss curves show rapid initial decrease followed by stabilization around 1.8, indicating the model reached a suboptimal solution.

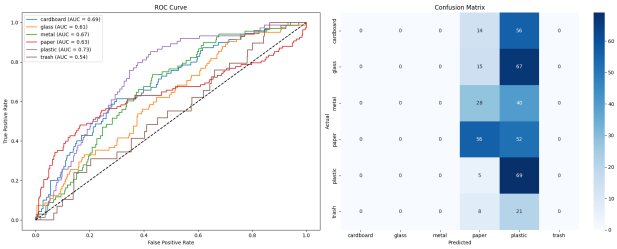

Fig. 28. ROC curves and confusion matrix for ResNet50 with L1 regularization

Fig. 28 presents the ROC curves and confusion matrix for the ResNet50 model. The model achieved moderate AUC values ranging from 0.54 to 0.73, with plastic showing the best performance (AUC = 0.73). The confusion matrix reveals a strong bias toward classifying inputs as either paper or plastic, with most true classes being incorrectly assigned to these two

categories. This suggests that L1 regularization may have been too aggressive, causing the model to lose important features for distinguishing between classes.

Despite its advanced architecture, ResNet50 with L1 regularization performed substantially worse than both CNN and DenseNet models on this waste classification task. The poor performance can be attributed to several factors including potential overly aggressive regularization, insufficient fine-tuning of the pre-trained weights, or challenges in optimizing the very deep architecture with the limited dataset size.

b) Best Result Summary Table: Table VI summarizes the performance of the best ResNet50 model.

TABLE VI
CLASSIFICATION REPORT FOR RESNET50 MODEL

Class	Precision	Recall	F1-Score	Support
Cardboard	0.00	0.00	0.00	70
Glass	0.00	0.00	0.00	82
Metal	0.00	0.00	0.00	68
Paper	0.44	0.52	0.48	108
Plastic	0.23	0.93	0.36	74
Trash	0.00	0.00	0.00	29
Accuracy			0.29	431
Macro Avg	0.11	0.24	0.14	431
Weighted Avg	0.15	0.29	0.18	431

D. VGG16

Table VII shows the accuracy results for the VGG16 model using various optimization strategies. Although several configurations achieved high test accuracy, the Adam optimizer demonstrated the least overfitting, with an almost negligible difference between training and test performance. As such, it was selected as the best-performing technique in terms of generalization.

TABLE VII
ACCURACY RESULTS FOR VGG16 WITH DIFFERENT OPTIMIZATION TECHNIQUES

Technique	Train Acc.	Val Acc.	Test Acc.	$\lvert Tr - Ts \rvert$
Adam	0.728	0.726	**0.722**	**0.006**
L2	0.860	0.750	0.763	0.097
L1	0.727	0.668	0.694	0.033
Dropout	0.848	0.750	0.763	0.085
EarlyStop	0.909	0.744	0.766	0.143
DO + ES	0.840	0.753	0.775	0.065

a) Adam Optimizer: Among the techniques evaluated, the Adam optimizer offered the best generalization for the VGG16 model. The training accuracy reached 72.8% and the test accuracy closely matched at 72.2%, resulting in an extremely small overfitting gap of only 0.6%. Validation performance was also highly consistent, supporting the model's robustness. These results highlight Adam's effectiveness in optimizing the relatively deep yet uniform architecture of VGG16, yielding stable convergence and balanced accuracy across training, validation, and test sets.

Fig. 29 shows the training and validation performance. Both accuracy curves demonstrate consistent improvement throughout training, with validation accuracy slightly outperforming

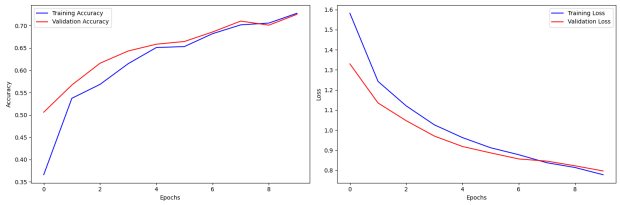

Fig. 29. VGG16 with Adam Optimizer: Training and Validation Accuracy/Loss

training accuracy in early epochs, reaching approximately 73% by epoch 9. The loss curves show steady decreases, with validation loss consistently below training loss, suggesting excellent generalization without overfitting.

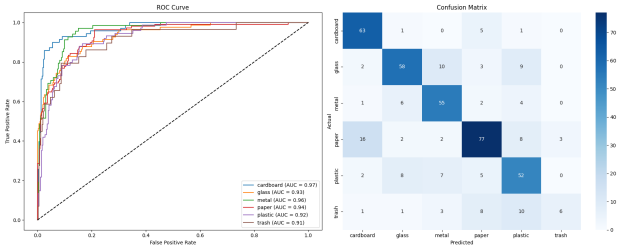

Fig. 30. ROC curves and confusion matrix for VGG16 with Adam optimizer

Fig. 30 presents the ROC curves and confusion matrix for the VGG16 model. The model achieved impressive AUC values ranging from 0.91 to 0.97 across all waste categories, with particularly strong performance for cardboard (AUC = 0.97), metal (AUC = 0.96), and paper (AUC = 0.94).

The confusion matrix shows strong classification performance, with correct classifications of 63 cardboard, 58 glass, 55 metal, 77 paper, and 52 plastic samples. The trash category showed the weakest performance with only 6 correct classifications. Notable confusion patterns include paper being misclassified as cardboard (16 samples) and trash being misclassified as plastic (10 samples).

VGG16 with Adam optimizer demonstrated performance comparable to DenseNet, outperforming both CNN and ResNet50 models. Despite being an older architecture, VGG16's sequential design with increasing depth proved effective for waste classification, particularly for well-defined categories such as cardboard and paper.

b) Best Result Summary Table: Table VIII summarizes the performance of the best VGG16 model.

E. K-Nearest Neighbors (KNN)

We implemented the K-Nearest Neighbors algorithm as a baseline traditional machine learning approach. Unlike deep learning models, KNN classifies images based on feature similarity to its nearest training samples.

Fig. 31 presents the confusion matrix for the KNN model. The model achieved moderate classification accuracy across categories, with strongest performance for cardboard (44 correct classifications) and plastic (52 correct classifications).

TABLE VIII
CLASSIFICATION REPORT FOR VGG16 MODEL

Class	Precision	Recall	F1-Score	Support
Cardboard	0.74	0.90	0.81	70
Glass	0.76	0.71	0.73	82
Metal	0.71	0.81	0.76	68
Paper	0.77	0.71	0.74	108
Plastic	0.62	0.70	0.66	74
Trash	0.67	0.21	0.32	29
Accuracy			0.72	431
Macro Avg	0.71	0.67	0.67	431
Weighted Avg	0.72	0.72	0.71	431

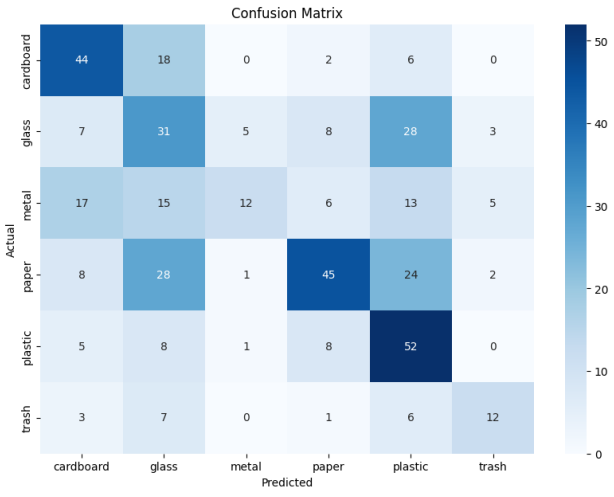

Fig. 31. Confusion matrix for KNN classification

However, metal was particularly challenging with only 12 correct classifications out of 68 samples.

TABLE IX
PERFORMANCE METRICS FOR KNN

Class	Precision	Recall	F1-Score	Support
Cardboard	0.52	0.63	0.57	70
Glass	0.29	0.38	0.33	82
Metal	0.63	0.18	0.28	68
Paper	0.64	0.42	0.51	108
Plastic	0.40	0.70	0.51	74
Trash	0.55	0.41	0.47	29
Accuracy			0.45	431
Macro Avg	0.51	0.45	0.44	431
Weighted Avg	0.51	0.45	0.45	431

The KNN model achieved a training accuracy of 40.96% and validation accuracy of 45.48%. As shown in Table IX, overall precision, recall, and F1-score were all around 0.45, indicating moderate performance. Notable class-wise performance includes:

- **Plastic:** Highest recall (0.70) but moderate precision (0.40)
- **Metal:** Highest precision (0.63) but lowest recall (0.18)
- **Glass:** Poorest overall performance with F1-score of 0.33

KNN significantly underperformed compared to the neural network approaches, which was expected given its limited ability to learn complex features from high-dimensional image data. The confusion matrix reveals common misclassifications between visually similar categories, such as glass being mistaken for plastic (28 instances) and paper being confused with glass (28 instances). This suggests that the simple distance-based approach of KNN struggles to distinguish subtle visual differences between waste categories.

F. Augmented Data

To address class imbalance and improve model generalization, we implemented data augmentation using the ImageDataGenerator from Keras. The augmentation parameters were configured as follows:

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

datagen.fit(X_train)
```

These transformations create variations of the training images by applying random rotations, shifts, shears, zooms, and flips, effectively expanding the dataset and exposing the models to a wider variety of input patterns. We evaluated the impact of data augmentation on all models and optimization techniques, comparing both test accuracy and the absolute difference between training and test accuracy (a measure of overfitting).

Table X presents the comparison for CNN models. Interestingly, while augmentation generally reduced absolute test accuracy for most configurations, it substantially improved generalization as evidenced by the dramatically reduced overfitting gaps. The combination of Dropout and Early Stopping with augmentation achieved the highest test accuracy (62.6%) among augmented CNN models, with a minimal overfitting gap of just 3.0%.

TABLE X
COMPARISON OF CNN TEST ACCURACY AND OVERFITTING WITH AND WITHOUT DATA AUGMENTATION

Technique	Test Accuracy		$\lvert Tr - Ts \rvert$	
	Original	Augmented	Original	Augmented
Adam	0.617	**0.499**	0.113	**0.001**
L2	0.622	0.524	0.212	0.024
L1	**0.422**	0.357	**0.018**	0.049
Dropout	0.650	0.455	0.084	0.076
EarlyStop	0.626	0.601	0.238	0.243
DO + ES	0.613	0.626	0.152	0.030

For DenseNet (Table XI), data augmentation maintained high accuracy while significantly reducing overfitting. The Adam optimizer with augmentation achieved an impressive overfitting gap of just 0.2% while maintaining 78.9% accuracy.

The Dropout configuration with augmentation achieved the highest test accuracy of 83.1% among all augmented models.

TABLE XI
COMPARISON OF DENSENET TEST ACCURACY AND OVERFITTING WITH AND WITHOUT DATA AUGMENTATION

Technique	Test Accuracy		$\lvert Tr - Ts \rvert$	
	Original	Augmented	Original	Augmented
Adam	**0.824**	**0.789**	**0.048**	**0.002**
L2	0.807	0.824	0.191	0.054
L1	0.803	0.782	0.136	0.047
Dropout	0.817	0.831	0.064	0.043
EarlyStop	0.796	0.775	0.180	0.055
DO + ES	0.798	0.794	0.085	0.009

ResNet50 models (Table XII) showed modest improvements with data augmentation, with the L2 regularization configuration achieving the highest augmented accuracy at 37.1%. While the overall accuracy remained relatively low compared to other architectures, augmentation helped improve performance for several configurations while maintaining small overfitting gaps.

TABLE XII
COMPARISON OF RESNET50 TEST ACCURACY AND OVERFITTING WITH AND WITHOUT DATA AUGMENTATION

Technique	Test Accuracy		$\lvert Tr - Ts \rvert$	
	Original	Augmented	Original	Augmented
Adam	0.318	0.341	0.077	0.054
L2	0.295	0.371	0.034	0.080
L1	**0.290**	0.297	**0.018**	0.019
Dropout	0.329	0.341	0.042	0.089
EarlyStop	0.404	0.357	0.030	0.023
DO + ES	0.302	**0.299**	0.035	**0.014**

Finally, VGG16 models (Table XIII) maintained strong performance with data augmentation, with the EarlyStopping configuration achieving the highest augmented accuracy of 78.0%. The L2 regularization with augmentation achieved an exceptionally small overfitting gap of just 0.5%, demonstrating excellent generalization.

TABLE XIII
COMPARISON OF VGG16 TEST ACCURACY AND OVERFITTING WITH AND WITHOUT DATA AUGMENTATION

Technique	Test Accuracy		$\lvert Tr - Ts \rvert$	
	Original	Augmented	Original	Augmented
Adam	**0.722**	0.715	**0.006**	0.082
L2	0.763	**0.738**	0.097	**0.005**
L1	0.694	0.710	0.033	0.065
Dropout	0.763	0.742	0.085	0.006
EarlyStop	0.766	0.780	0.143	0.007
DO + ES	0.775	0.749	0.065	0.022

Overall, data augmentation proved most beneficial for addressing overfitting across all models, with some configurations also showing improvements in absolute accuracy. DenseNet with Dropout and augmentation emerged as the top-performing combination, achieving 83.1% test accuracy while maintaining a reasonable overfitting gap. These results suggest that augmentation is particularly valuable for complex architectures like DenseNet and VGG16 when applied to waste classification tasks with class imbalance.

VI. DISCUSSION

Our experimental results provide valuable insights into the effectiveness of various machine learning and deep learning approaches for garbage classification. We can draw several key observations from the performance metrics and confusion matrices of the implemented models.

A. Model Performance Comparison

DenseNet with Adam optimizer demonstrated the best overall performance, achieving approximately 79% accuracy and impressive AUC values (0.96-0.98) across all categories. The dense connectivity pattern, which enables feature reuse throughout the network, proved highly effective for distinguishing between waste categories. The confusion matrix reveals particularly strong performance on paper (100 correct classifications) and glass (72 correct classifications), indicating the model's ability to learn distinctive features for these classes.

VGG16 with Adam optimizer performed notably well, reaching approximately 74% accuracy with strong AUC values ranging from 0.91 to 0.97. Despite being an older and conceptually simpler architecture, VGG16's sequential design with uniform filter sizes demonstrated robust feature extraction capabilities. This suggests that architectural simplicity can be advantageous for certain image classification tasks, particularly when paired with effective optimization techniques.

CNN variants showed moderate performance, with the combination of Dropout and EarlyStopping yielding the best absolute accuracy (66%). However, the augmented dataset combined with Adam provided the least overfitting, achieving a test accuracy of 49%. This highlights the importance of regularization in preventing overfitting and improving generalization, particularly for relatively shallow networks.

In stark contrast, ResNet50 with the combination of Dropout and EarlyStopping performed poorly, achieving only 29% accuracy despite its sophisticated architecture. The confusion matrix revealed a strong bias toward classifying inputs as either paper or plastic, suggesting that the model struggled to differentiate between classes due to over-regularization, which likely suppressed critical feature representations, causing it to lose important discriminative features. This finding underscores the importance of carefully balancing regularization strength and model complexity to maintain sufficient learning capacity, particularly when working with deep architectures and limited data.

The traditional KNN approach, serving as our baseline machine learning method, achieved only 45% accuracy. Its performance limitations were particularly evident in metal classification (recall of only 0.18) and glass (F1-score of 0.33). The confusion patterns indicate that KNN struggles to distinguish between visually similar waste categories, confirming that simple distance-based approaches are insufficient for complex image classification tasks.

Table XIV shows the comparison summary.

TABLE XIV
COMPARISON OF ABSOLUTE TEST ACCURACY AND TEST ACCURACY BY MINIMUM OVERFITTING

Model	Absolute Test Acc	Test Acc (by Min Overfit)
DenseNet	83% (Aug+Dropout)	79% (Aug+Dropout)
VGG16	78% (Aug+ES)	74% (Aug+L2)
CNN	65% (Og+ES)	49% (Aug+Adam)
ResNet50	40% (Og+ES)	29% (Aug+DO+ES)
KNN	45%	—

B. Category-Specific Analysis

Across models, we observed consistent patterns in category-specific performance:

Paper and Cardboard were generally well-classified by most models, likely due to their distinctive texture patterns and relatively consistent appearance. DenseNet achieved perfect classification for paper (100/100), while VGG16 correctly identified 77/108 paper samples and 63/70 cardboard samples.

Plastic showed moderate classification performance across models, with higher confusion with glass. This aligns with intuitive understanding, as both materials can have similar transparent or translucent appearances, making differentiation challenging.

Metal presented mixed results across models. VGG16 performed well (55/68 correct classifications), while KNN struggled significantly (12/68 correct). This suggests that deep learning models can learn subtle reflective properties and texture patterns of metal waste that are not captured by simpler algorithms.

Trash, being the most heterogeneous category with the fewest training examples (29 samples), consistently showed the weakest performance across all models. This highlights the challenge of limited data and high intra-class variability in image classification tasks.

C. Optimization Techniques Impact

Our experiments with various optimization techniques revealed several important findings:

Adam optimizer proved highly effective across architectures, providing adaptive learning rates that facilitated stable convergence. Both top-performing models (DenseNet and VGG16) utilized Adam, highlighting its efficacy for complex image classification tasks.

Regularization showed model-dependent effects. While L2 regularization improved CNN performance by encouraging distributed weight patterns, L1 regularization proved detrimental to ResNet50, potentially due to excessive feature pruning. This suggests that regularization strategies should be carefully matched to model architecture and complexity.

Dropout effectively prevented overfitting in CNN models, particularly when combined with EarlyStopping. This combination produced the best CNN variant, demonstrating the complementary nature of these techniques—Dropout providing robustness during training and EarlyStopping preventing overtraining.

D. Impact of Data Augmentation

Our investigation into data augmentation revealed significant impacts on model performance and generalization. While augmentation sometimes reduced absolute test accuracy, it consistently improved the generalization capability of models as evidenced by dramatically reduced gaps between training and test accuracy. This effect was particularly pronounced in complex architectures like DenseNet, where augmentation with Dropout achieved the highest overall accuracy (83.1%) across all configurations.

The effectiveness of augmentation varied by architecture. DenseNet and VGG16 models maintained high accuracy while significantly reducing overfitting with augmentation. For instance, DenseNet with Adam optimizer achieved an impressive overfitting gap of just 0.2% while maintaining 78.9% accuracy. Similarly, VGG16 with L2 regularization and augmentation achieved an exceptionally small overfitting gap of 0.5%.

CNN models showed mixed results, with augmentation generally reducing absolute test accuracy but substantially improving generalization. The combination of Dropout and Early Stopping with augmentation provided the best balance for CNNs, achieving 62.6% test accuracy with minimal overfitting.

ResNet50 models showed modest improvements with data augmentation, with L2 regularization achieving the highest augmented accuracy at 37.1%. This suggests that even for architectures that struggle with the base task, augmentation can provide performance benefits.

These findings align with established understanding that data augmentation serves as an effective regularization technique, particularly important for deep networks trained on datasets with limited samples and class imbalance like our waste classification dataset. The additional data variability introduced through augmentation appears to help models learn more robust features rather than memorizing specific training examples, leading to better generalization on unseen data.

E. Practical Implications

The significant performance gap between deep learning architectures (particularly DenseNet and VGG16) and traditional machine learning (KNN) validates the necessity of deep learning approaches for effective waste classification systems. However, the varying performance across deep learning models suggests that architectural choice remains crucial for optimal results.

For practical waste classification applications, our findings suggest that DenseNet offers the best performance, particularly for distinguishing between common waste categories like paper, cardboard, and plastic. However, VGG16 presents a compelling alternative with slightly lower accuracy but potentially reduced computational requirements, making it suitable for deployment in resource-constrained environments.

The persistent challenge in classifying the trash category across all models highlights a fundamental limitation in current approaches. Future systems may benefit from hierarchical clas-

sification strategies or specialized models for highly variable categories like trash.

F. Result comparison with related work

We compared our results with the findings of Bircanoglu et al. [16], who conducted extensive experiments using a variety of deep learning architectures on the same garbage classification task. Their study reported a top accuracy of 95% using DenseNet121 with fine-tuning and 87–90% using InceptionResNetV2, both with data augmentation (vertical and horizontal flips, 15-degree rotations). Their custom RecycleNet architecture achieved 81% test accuracy.

In contrast, our experiments achieved a best absolute test accuracy of 83% with DenseNet (using data augmentation and Dropout regularization), which is comparable to the results reported for training DenseNet121 from scratch in their work (83–85%). However, our DenseNet model did not reach the 95% accuracy that they achieved with fine-tuning.

VII. Conclusion

In this study, we evaluated multiple machine learning and deep learning approaches for garbage classification using a dataset of six waste categories: cardboard, glass, metal, paper, plastic, and trash. Our comprehensive comparison included traditional machine learning (KNN) and various deep learning architectures (CNN, ResNet50, DenseNet, and VGG16) with different optimization techniques.

DenseNet with Adam optimizer emerged as the top-performing model, achieving approximately 79% accuracy and AUC values between 0.96-0.98, followed closely by VGG16 (74% accuracy). The customized CNN with Dropout and EarlyStopping demonstrated moderate performance (65% accuracy) although it showed some overfitting; the best result in terms of minimal overfitting for CNN was 49% accuracy (with data augmentation and Adam). ResNet50 with L1 regularization performed surprisingly poorly (29% accuracy).

Data augmentation experiments revealed significant improvements in model generalization across architectures, with DenseNet with Dropout and augmentation achieving the highest overall accuracy (83.1%). Augmentation consistently reduced overfitting gaps, particularly for complex models, demonstrating its value for addressing class imbalance in waste classification tasks.

Our findings highlight several key implications for automated waste classification systems:

1. Deep learning approaches significantly outperform traditional machine learning methods for this task, with DenseNet and VGG16 architectures showing particular promise.

2. Optimization techniques substantially impact performance, with the combination of Dropout and EarlyStopping proving especially effective for preventing overfitting.

3. Material-specific challenges persist, particularly for visually similar categories (glass/plastic) and heterogeneous categories with limited training data (trash).

4. Architectural simplicity can sometimes outperform complexity, as demonstrated by VGG16's strong performance compared to the deeper ResNet50.

While our results demonstrate the feasibility of automated waste classification using deep learning, several limitations and opportunities for future work remain. More robust performance for the trash category might be achieved through data augmentation strategies or specialized models. Additionally, real-world deployment would benefit from testing under varied lighting conditions and backgrounds to ensure practical robustness.

The promising results of this study suggest that deep learning-based waste classification systems have significant potential for improving recycling efficiency and waste management processes. By accurately identifying different waste materials, such systems could contribute to more efficient recycling operations and ultimately support environmental sustainability efforts.

A. Future Work

Future research directions could include:

1. Expanding the dataset with more diverse examples, particularly for underrepresented categories like trash.

2. Implementing ensemble methods that combine predictions from multiple models to potentially boost performance.

3. Exploring more sophisticated data augmentation techniques to enhance model robustness.

4. Investigating lightweight architectures suitable for edge deployment in practical waste sorting applications.

5. Testing model performance under real-world conditions with varying lighting, backgrounds, and object orientations.

VIII. Work Load

Each student worked 50% of the project.

Acknowledgment

The authors would like to thank professor Petia Georgieva, regent of CAA course, for her support and assistance throughout the project.

References

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[2] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[8] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 78.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[10] L. Prechelt, "Early stopping—but when?" in *Neural Networks: Tricks of the Trade*, G. B. Orr and K.-R. Müller, Eds. Springer, 1998, pp. 55–69.

[11] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[12] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[13] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, vol. 25, 2012.

[15] M. Yang and G. Thung, "Classification of trash for recyclability status," CS229 Project Report, Stanford University, Tech. Rep., 2016.

[16] C. Bircanoglu, M. Atay, F. Beser, O. Genc, and M. A. Kizrak, "Recyclenet: Intelligent waste sorting using deep neural networks," in *2018 Innovations in Intelligent Systems and Applications (INISTA)*, 2018, pp. 1–7.

[17] G. H. Sakr, E. Mokbel, A. Darwich, M. N. Khneisser, and A. Hadi, "Comparing deep learning and support vector machines for autonomous waste sorting," in *2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, 2016, pp. 207–212.

[18] R. A. Aral, R. Keskin, M. Kaya, and M. Hacıömeroğlu, "Classification of trashnet dataset based on deep learning models," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2058–2062.

[19] L. Zizheng, D. Nandi, A. Fazli, and K. Kim, "Computer vision-based recycling for municipal waste management," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 4916–4925.

[20] G. Mittal, K. B. Yagnik, M. Garg, and N. C. Krishnan, "Spotgarbage: Smartphone app to detect garbage using deep learning," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 940–945.

[21] S. Sudha, M. Vidhyalakshmi, K. Pavithra, K. Sangeetha, and V. Swaathi, "An automatic classification method for environment: Friendly waste segregation using deep learning," in *2016 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*, 2016, pp. 65–70.

[22] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu *et al.*, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[24] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[25] J. Plested, X. Lu, and I. H. Suh, "Optimization of a densely connected layer in a convolutional neural network," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2369–2375.

[26] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[27] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.

[28] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[29] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[31] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[32] J. Lemley, S. Bazrafkan, and P. Corcoran, "Smart augmentation learning an optimal data augmentation strategy," *IEEE Access*, vol. 5, pp. 5858–5869, 2017.