Waste Segregation Using CNN SC407 - Group 3

Manan Pareek (202101018), Jayveer Rathwa (202101142),
Keni Patel (202101194), Viraj Rathva (202101196),
Sahilkumar Machhar (202101219), Rohan Mistry (202101231),
Akshat Prasad (202101419), Rohit Thakre (202101441),
Varun Vyas (202101468), Umang Trivedi (202101471),
Nancy Patel (202101491)

***Abstract*—This paper presents a comprehensive approach to waste segregation using deep learning techniques. The proposed methodology involves three phases: Data Collection and Model Training, Model Testing, and API Development with Image Upload. We focus on robust data collection, model training, and testing using convolutional neural network (CNN) architectures like EfficientNet. The trained model is deployed as an API for garbage classification, allowing users to upload images and receive predictions. Experimental results demonstrate the effectiveness of the proposed approach, achieving high accuracy and generalization performance. Future extensions involve real-time garbage classification using camera input for practical implementation.**

***Index Terms*—Waste Segregation, Deep Learning, Convolutional Neural Networks, EfficientNet, Transfer Learning, API Development, Model Testing, Data Collection.**

I. Introduction

OUR ENVIRONMENT, encompassing everything from living things to air and water, provides the foundation for life on Earth. Understanding how these elements interact and influence climate is crucial for maintaining a stable climate. Studying the environment and climate allows us to predict how changes made by human activities will impact others, solve environmental challenges like pollution and waste disposal and make informed decisions about our consumption habits and impact on the planet. Ultimately, this knowledge empowers us to create a healthier and more sustainable future for all.

A. Problem Statement

In modern societies, waste management is a critical aspect of maintaining environmental sustainability and public health. Inefficient waste management, particularly manual sorting of mixed waste in India, poses a severe threat to our environment and health. Landfills overflowing with improperly sorted waste release harmful greenhouse gases (methane: 55%, carbon dioxide: 35%) and contaminate soil and water [1]. India alone generates a staggering 1.28 lakh tons of daily municipal solid waste, with only 70% collected and a mere 12.45% processed [2]. This highlights the urgent need for a more efficient system.

To address this challenge, we propose an automated waste classification system using a Convolutional Neural Network (CNN) integrated with a web API. The system will classify waste materials into twelve distinct categories: batteries, biological waste, brown glass, cardboard, clothing, green glass, metal, paper, plastic, shoes, trash, and white glass. This automated approach will significantly improve waste management efficiency, promote recycling opportunities, and contribute to a more sustainable future.

B. Relevant Works

In recent years, significant progress has been made in the field of garbage classification using machine learning techniques.

The paper [3] discusses garbage classification with CNNs. It emphasizes the significance of accurate garbage classification in waste recycling systems. They evaluate the performance of various CNN models and conclude that even simple CNNs can achieve good performance on this task.

Paper [4] presented a multi-modal garbage classification approach that combined information from different sources such as visual images, depth maps, and infrared sensors. Deep fusion networks were employed to effectively integrate multi-modal data and learn discriminative features for garbage classification. Experimental results demonstrated the superiority of the proposed method over single-modal approaches.

The study of [5] investigates the application of deep learning for automatic waste classification, specifically focusing on categorizing waste items into plastic, paper, cardboard, and metal. The authors propose a CNN -based approach to address the challenges of waste management in metropolitan areas. Their system leverages a labelled dataset of waste images to train the CNN model, achieving a testing accuracy of 76%.

The article [6] is about a deep learning framework for public garbage classification. The proposed algorithm is called PublicGarbageNet. It is a multi-task classification algorithm that can identify four major categories of domestic garbage and ten subclasses of garbage. The authors created a new public garbage dataset to train the model. The model achieved an accuracy of 96.35%.

The paper [7] proposes a mobile application for classifying garbage. The model is based on MobileNet and can sort garbage into six categories. The model was trained on a dataset of over 2,500 images and achieved an accuracy of 87.2%. The authors then optimized the model for mobile devices.

C. Comparison and our Contributions

1) Classification Categories and Database:: Paper [3] , Uses a dataset from Kaggle with 2527 images of single garbage objects on clean backgrounds. Lighting and pose configurations for objects in different images is different. They are divided into six recycling categories: cardboard, glass, metal, paper, plastic, and trash.
Paper [5] mentions using the TrashNet dataset, which consists of 2,527 images categorized into six classes.
Paper [6] provides classification into six broad categories: glass, paper, cardboard, plastic, metal, and other trash which also uses the same dataset TrashNet.
In paper [7] The dataset contains 2527 trash images, with six categories. The object was placed on a white poster board as background. Pictures were taken under sunlight and/or room lighting.
Comparison: The difference in the number and types of categories reflects different priorities or requirements for waste management in each solution. while all these offers a decent level of categorization, they lacks the granularity provided by our solution, which classifies garbage into twelve distinct categories. Our solution aims to classify waste materials into twelve distinct categories, which includes batteries, biological waste, brown glass, cardboard, clothing, green glass, metal, paper, plastic, shoes, trash, and white glass.
Our solution utilizes a dataset from Kaggle with approximately 15,000 images depicting various types of garbage across multiple classes. The dataset is used for training and validating the CNN model. Higher number of training instances provides better generalizability for our model.

2) Model Architecture:: Our study Utilizes the EfficientNetV2B1 CNN architecture for garbage classification, leveraging pre-trained weights from ImageNet for transfer learning.
In Paper [3] Explores Histogram of Oriented Gradients (HOG) features combined with Support Vector Machines (SVM), and ResNet50 for garbage classification.
in paper [7] : Study: Retrains the MobileNet model, originally trained on the ImageNet dataset, using the garbage image dataset.
Comparison: Our paper focuses on a specific CNN architecture (EfficientNetV2B1) optimized for image classification tasks, while the others have explored a variety of techniques including traditional methods like HOG + SVM and deeper architectures like ResNet50.

3) Evaluation and Results:: Paper [3] Divides the dataset into train and test sets, and evaluates model performance using the test set but does not mention visualization of metrics.
Paper [5] presents limited evaluation results, mentioning a maximum accuracy of 76%and providing a training and validation graph for accuracy.
Paper [6] Reports different accuracy rates achieved by various CNN architectures tested on the TrashNet dataset with maximum accuracy of the 98.95%
In paper [7], the study focuses on optimizing the model for inference on mobile devices, which involves techniques like quantization and compression.

Comparison: Our study Evaluates model performance using a separate validation dataset, calculating test loss, accuracy and confusion matrix. Visualizes training and validation metrics to monitor the learning process. Solution achieves a higher accuracy rate of 99.16%. The results are summarized in I

TABLE I
COMPARISON OF ACCURACY FOR DIFFERENT PAPERS

Model	**Accuracy (%)**
Paper [3]	NULL
Paper [5]	76
Paper [6]	98.95
Our Model	99.16

D. Organization

The paper presents a comprehensive approach to waste segregation using deep learning, consisting of three phases: Data Collection and Model Training, Model Testing, and API Development with Image Upload which are descirbed in detail in II. Leveraging EfficientNet architecture, the model achieves high accuracy and generalization performance. Experimental results provided in V demonstrate the effectiveness of the proposed approach in classifying various waste categories. Furthermore, alternative methodologies and architectures were explored and evaluated which are described in IV, with EfficientNet emerging as the most promising. Future work VI involves integrating real-time garbage classification using camera input for practical implementation. The study contributes to addressing the critical challenge of waste management, promoting sustainability and environmental conservation.

II. PROPOSED APPROACH

Our proposal is divided into three phases: **Data Collection and Model Training**, **Model Testing**, and **Api Development with Image Upload**. The Github Link for our Project code is provided here, [8].

A. Data Collection and Model Training

The foundation of our project lies in robust data collection and model training, essential for developing an accurate and efficient waste segregation solution. Leveraging existing datasets is crucial for building a comprehensive understanding of waste categorization. We have obtained our primary dataset from Kaggle, a reputable platform for open datasets. The dataset offers a diverse collection of about 15 thousand images depicting various types of garbage across multiple classes. These images serve as the cornerstone for training our machine learning models.

Technique of resizing is employed to enhance the dataset's robustness and generalizability. This technique involves adjusting the dimensions of images within the dataset to a uniform size. In our case, we have resized the images to 224 pixels by 224 pixels.

Additionally, we perform exploratory data analysis (EDA) to gain insights into the distribution of different waste categories and identify potential biases or anomalies.

For model training, we divided our dataset into training and validation sets to assess the model's performance and prevent

overfitting. We allocated a portion of the dataset (20%) for validation purposes using a validation split parameter. The training process involved feeding batches of images from the training set into the model, adjusting the model's parameters to minimize the loss function, and updating the model's weights iteratively through backpropagation (refer C). During training iterations, the model's performance was monitored using the training data, while the validation set was used periodically to evaluate the model's generalization ability and detect potential overfitting.

We utilized the EfficientNet architecture as the backbone for our convolutional neural network (CNN) model. For more information on EfficientNet, refer to A.

In our implementation, we employed the EfficientNetV2B1 variant, which offers a good balance between model complexity and performance. The model's pretrained weights, obtained from the ImageNet dataset, serve as valuable starting points, enabling efficient transfer learning for our specific classification task. For more information on Transfer Learning, refer to B.

B. Model Testing

In the Model Testing phase, we evaluate the performance of the trained model using a separate validation dataset. To assess the model's effectiveness, we calculate the test loss and accuracy using the 'evaluate' method in Keras library. The test loss indicates how well the model's predictions match the actual labels, while the test accuracy represents the proportion of correctly classified instances.

Additionally, we visualize the training and validation loss, as well as the training and validation accuracy, over the course of training using line plots. These plots provide insights into the model's learning process and help identify potential issues such as overfitting or underfitting.

Furthermore, we constructe a confusion matrix to analyze the model's performance in more detail. The confusion matrix displays the counts of true positive, true negative, false positive, and false negative predictions across different classes. By visually inspecting the confusion matrix, we can identify which classes the model struggles with and assess its overall performance in terms of classification accuracy.

In addition to evaluating our model's performance using the EfficientNet architecture, we sought to enhance our understanding by comparing it with other widely used convolutional neural network (CNN) architectures, namely ResNet50, VGG16, and MobileNet. By experimenting with these alternative architectures, we aimed to gain insights into their respective strengths and weaknesses, as well as their suitability for our specific classification task.

C. API Development Using Image Upload

Utilizing the FastAPI framework, we developed an API for Garbage Classification, enabling users to upload images and receive predictions regarding the type of garbage depicted. Upon uploading an image, the API processes the image data and feeds it into out trained model, which has been trained to classify images into various garbage categories. The model generates predictions, indicating the most probable class of garbage present in the image, along with a confidence score representing the model's certainty in its prediction.

III. ALGORITHMS

In this section, we present the algorithms for data preparation, deep learning model development, and garbage classification API (relavant code file can be found as efficientNet.ipynb and api.py respectively at [8]).

Algorithm 1 Data Preparation and Model training

Result: Preprocessed dataset and trained model

- Load dataset and create dataframe with labels
- Resize all images to 224×224 pixels
- Randomly split dataset into 80% training and 20% validation
- Design CNN architecture using Keras (refer to Algorithm 2 for the Architecture)
- Compile model with Adam optimizer and sparse categorical cross-entropy loss
- Implement early stopping to prevent over-fitting
- Train model for 10 epochs with gradient descent using backporpogation
- Save best performing model and evaluate the model performance

Algorithm 2 Model Architecture

Result: CNN Model Architecture - 7,52,652 trainable parameters and 69,31,124 non-trainable parameters

- Define base model using pre-trained EfficientNet
- Freeze base model's weights
- Create sequential model and add base model
- Flatten base model's output
- Apply dropout regularization with rate 0.5
- Add dense layer with softmax activation for 12 classes

Algorithm 3 Garbage Classification API

Result: Garbage Classification API

- Load pre-trained Keras model
- Define route for file uploads
- Read uploaded image file, resize to 224×224, and convert to array
- Perform predictions using loaded model
- Return predicted class and confidence score

Algorithm 1

- Resizing the images ensures uniformity in input dimensions, which is necessary for feeding images into deep learning models. In this algorithm, images are resized to 224×224 pixels to match the input size expected by the pre-trained EfficientNet base model (referenced in Algorithm 2).
- Dataset is split into two parts for training and validation, which are created randomly and kept disjoint. Doing so

provides us with a subset of dataset to be used for the model evaluation.

- Gradient descent is an optimization algorithm used to minimize the loss function and find the optimal parameters of a model. In each iteration, the algorithm updates the parameters by moving them in the direction opposite to the gradient of the loss function. Mathematically, the parameter update rule in gradient descent can be expressed as:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla J(\theta_t)$$

Where:

  - θ_t represents the parameters of the model at iteration t.
  - α is the learning rate, controlling the size of the update step.
  - $\nabla J(\theta_t)$ denotes the gradient of the loss function J with respect to the parameters θ_t.

This iterative process continues until convergence, where the algorithm reaches a minimum of the loss function, indicating that the model parameters have been optimized.

- The Adam optimizer adaptively adjusts the learning rate for each parameter. This adaptive learning rate helps in efficiently training models by automatically adjusting the learning rates during training based on the magnitudes of the gradients.
- Early stopping aims to prevent overfitting by monitoring the model's performance on a separate validation dataset during training. The training process is halted when the performance on the validation set stops improving or starts deteriorating, indicating that the model is beginning to overfit the training data, thereby saving computational resources and time.
- Find information about back propagation algorithm used for training networks in C

Algorithm 2

The architecture of the convolutional neural network (CNN) model used in the algorithm is depicted in Figure 1. It consists of the layers in the model, their type and the dimensions of the input and output for each of the layer.

- For information on EfficientNet, refer to A.
- Freezing the weights of the base model involves preventing them from being updated during the training process. This is beneficial when the pre-trained model has been trained on a large dataset and has already learned useful features that can be relevant to the new task. For more information about transfer lerning, refer to B.
- Dropout is used to prevent overfitting by randomly dropping out a fraction of the neurons during training. This dropout operation is applied independently to each unit with a certain probability. By randomly removing neurons, dropout introduces noise into the network, forcing it to learn more robust features. Dropout regularization helps to improve the generalization performance of the model by capturing more diverse representations in the data.

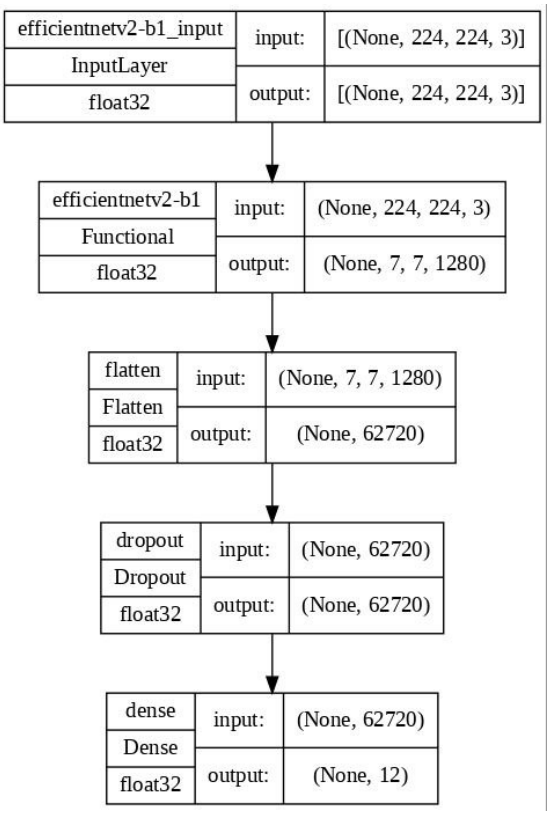

Fig. 1. CNN Model Architecture, consisting of 5 layers, where second layer could itself be containing a number of other layers

- Dense layer is a layer in neural network, where every neuron of the layer is connected to each of the neuron from the previous layer.
- The softmax activation function is commonly used in the output layer of classification neural networks to produce probabilities for each class in a mutually exclusive set. Given a vector of real-valued scores, softmax computes a probability distribution over the classes. Mathematically, the softmax function is defined as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

Where:

  - z_i represents the input logit for class i.
  - N is the total number of classes.

The softmax function ensures that the resulting probabilities sum up to one, making it suitable for multi-class(12 in our case) classification tasks. During inference, the class with the highest probability output by the softmax function is chosen as the predicted class by the model.

IV. Alternate Approaches Experimented

We explored a variety of methodologies using different deep learning models, including the selection of the model described in the preceding section based on comprehensive evaluation. This section outlines some of the alternative models and techniques we investigated.

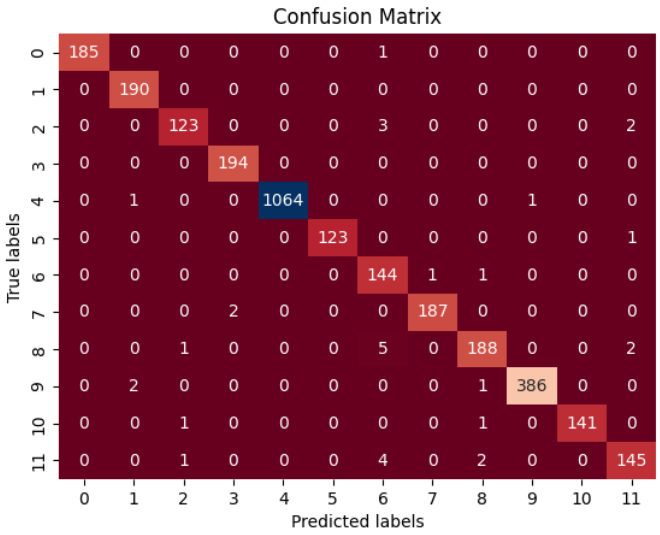

Fig. 2. For our model, we can observe a high rate of true positives and a very low rate of false positives and false negatives on the validation dataset.

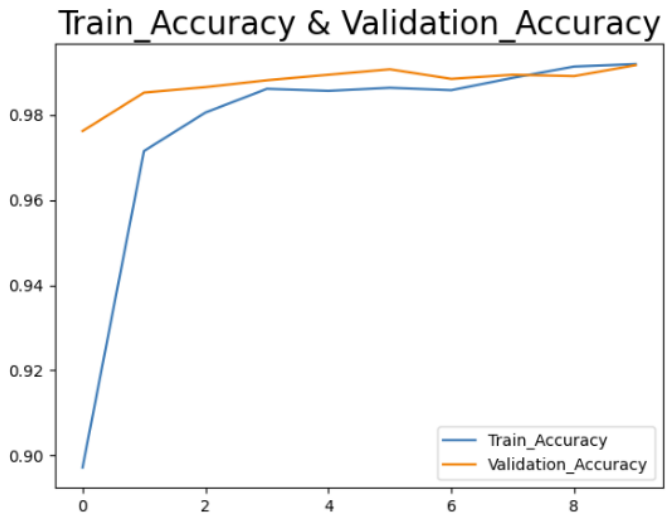

Fig. 3. The training accuracy increases over epochs, indicating that the model is correctly classifying more instances in the training dataset. Similarly, the validation accuracy (val_accuracy) also increases over epochs, showing that the model's performance on unseen data improves over time.

We explored diverse architectural configurations, including variations in the number of layers and neurons, as well as exploration of data augmentation strategies and layer regularization. Additionally, we delved into transfer learning, employing established models such as MobileNet and VGG16. (relavant code file can be found as mobileNet.ipynb and vgg16.ipynb respectively at [8]).

Throughout our exploration, we assessed the performance of each approach. Ultimately, the architecture featuring EfficientNet as the base model emerged as the most promising, demonstrating superior performance compared to the alternative strategies considered. The comparison of their accuracies are summarized in II

TABLE II
COMPARISON OF ACCURACY FOR DIFFERENT MODELS

Model	**Accuracy (%)**
VGG16	87
mobileNet	90.2
efficientNet	99.16

V. NUMERICAL RESULTS

Fig. 2 provides an experimental evaluation of our proposed methodology by use of a Confusion Matrix.

1) **Diagonal Elements (True Positives)**: The diagonal elements represent the counts of instances correctly classified for each class. Higher values along the diagonal indicate strong performance in correctly identifying instances of those classes.
2) **Off-Diagonal Elements (False Positives and False Negatives)**: Off-diagonal elements represent misclassifications. A non-zero value in a row but not in the corresponding column indicates false negatives (instances of that class incorrectly classified as other classes), while a non-zero value in a column but not in the corresponding row indicates false positives (instances of other classes incorrectly classified as that class).

Fig. 3 provides an experimental evaluation of Train and Validation Accuracy.

Fig. 4 provides an experimental evaluation of Train and Validation Loss. The loss was calculated using Sparse categorical cross entropy. Sparse categorical cross entropy is a loss function used in classification tasks where each instance belongs to exactly one out of multiple classes. It calculates the difference between the predicted probability distribution and the true class labels, penalizing deviations from the correct classification. The "sparse" aspect indicates that the true labels are integers rather than one-hot encoded vectors, making it suitable for cases where the number of classes is large. Mathematically, The sparse categorical cross-entropy loss function is defined as:

$$\mathcal{L}(y, \hat{\mathbf{y}}) = -\Sigma_{i=1}^{N} I(y = i) \log(\hat{y}_i)$$

Where:

- N is the number of classes.
- y is the true label.
- $I(y = i)$ is the indicator function that returns 1 if y equals i and 0 otherwise.
- $\hat{y}_i$ is the predicted probability for class i.

The test loss, calculated as 0.1011, represents the discrepancy between the model's predictions and the actual labels in the test dataset.

Additionally, the test accuracy is reported as 99.16%. This metric quantifies the proportion of correctly classified instances out of the total instances in the test dataset.

Fig. 5 provides an experimental result of model prediction on unseen data.

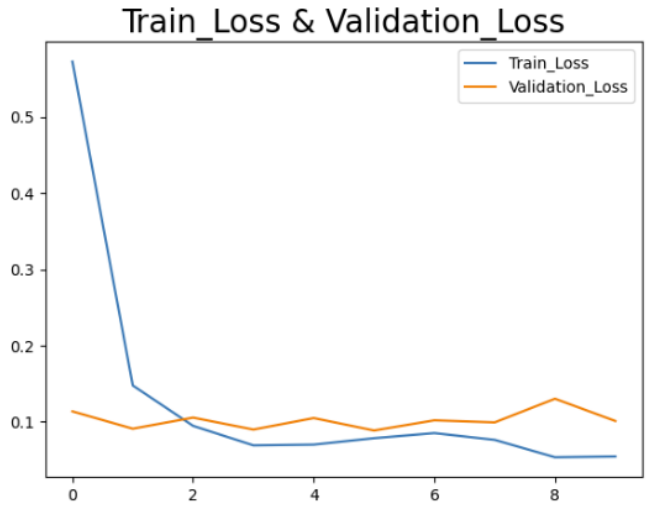

Fig. 4. The training loss (loss) steadily decreases over epochs, indicating that the model is learning and becoming better at minimizing errors on the training data. Similarly, the validation loss (val_loss) also decreases initially, suggesting that the model generalizes well to unseen data.

Fig. 5. Model prediction for some unseen data.

VI. Conclusion

Generalization Performance: The validation metrics (val_loss and val_accuracy) are crucial for assessing the model's generalization performance. A decreasing validation loss and increasing validation accuracy over epochs suggest that the model is generalizing well to unseen data.

A lower test loss indicates better agreement between the predicted and actual values.

A higher test accuracy suggests that the model has successfully learned to generalize patterns from the training data to unseen data, indicating its effectiveness in the classification task.

Future Work: An envisioned idea involves integrating real-time input of garbage images using a camera, capturing images at regular intervals, and employing the trained model to predict the category of garbage in real-time. This proposed approach holds potential for practical implementation at industrial waste segregation sites where a hardware design could be implemented for garbage items to come one after another and then using the camera classifying them into the categories to finally drop into different bins. This could be a proactive solution for automated and efficient waste categorization.

Appendix A
More Information on EfficientNet

EfficientNet is a convolutional neural network (CNN) architecture developed specifically for image classification tasks. What distinguishes EfficientNet from other architectures is its emphasis on achieving high accuracy while maintaining computational efficiency. This efficiency is achieved through a technique known as compound scaling, which systematically balances the depth, width, and resolution of the network.

The core idea behind EfficientNet is to scale up all dimensions of the network architecture in a coordinated manner, ensuring that each dimension contributes optimally to the model's performance. By carefully adjusting the depth (number of layers), width (number of channels), and resolution (input image size), EfficientNet achieves a favorable trade-off between model complexity and computational cost.

Appendix B
More Information on Transfer Learning

Transfer learning is a machine learning approach where a model trained on one task is utilized to improve learning on a related task. Initially, the model is pretrained on a source task using a large dataset. This pretraining phase enables the model to learn generic features and patterns from the data. Then, the pretrained model is fine-tuned on a smaller dataset specific to the target task, adjusting its parameters to better suit the nuances of the target task. By leveraging the knowledge gained during pretraining, transfer learning accelerates learning and enhances performance on the target task, particularly when data or computational resources are limited. This technique has been widely used across various domains, including computer vision, natural language processing, and speech recognition, to achieve state-of-the-art results with less training data and computational effort.

Appendix C
More Information on Backpropagation:

Backpropagation, short for "backward propagation of errors," is a fundamental algorithm used in training artificial neural networks (ANNs), including deep learning models.

1) **Forward Pass:** During the forward pass, the input data is fed through the neural network, layer by layer, to generate predictions. Each neuron in the network performs a weighted sum of its inputs, followed by the application of an activation function to produce an output.

2) **Loss Calculation:** Once the predictions are generated, they are compared to the actual targets using a loss function, which measures the difference between the predicted and actual values. The goal of training is to minimize this loss function.
3) **Backward Pass:** In the backward pass, the gradient of the loss function with respect to each weight in the network is computed using the chain rule of calculus. This gradient indicates how much the loss would change if the value of each weight were adjusted slightly.
4) **Weight Update:** Finally, the weights of the network are updated using an optimization algorithm that takes into account the computed gradients. The weights are adjusted in the opposite direction of the gradient, aiming to minimize the loss function iteratively.

REFERENCES

[1] "1.3. problems caused by mismanagement of waste." https://sisu.ut.ee/waste/book/13-problems-caused-mismanagement-waste, 2016.
[2] R. K. Kodali and V. S. K. Gorantla, "Smart solid waste management," in *2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, pp. 200–204, IEEE, 2017.
[3] Y. Zhang, G. Sun, Z. Liu, and Y. Xu, "Classification of garbage images using convolutional neural networks," *IEEE Access*, vol. 7, pp. 143128–143138, 2019.
[4] J. Tan and A. Liu, "Multi-modal garbage classification using deep fusion networks," *IEEE Transactions on Multimedia*, vol. 25, no. 3, pp. 450–465, 2023.
[5] X. Li, P. He, J. Sun, Y. Peng, and M. Wang, "Deep learning for automatic waste classification," *2019 IEEE International Conference on Environment Design (ICED)*, pp. 1–6, 2019.
[6] L. X. X. W. Z. T. Zeng, M. and Y. Liu, "Publicgarbagenet: A deep learning framework for public garbage classification," *2020 39th Chinese Control Conference (CCC)*, pp. 9189561–9189564, 2020.
[7] S. L. Rabano, M. K. Cabatuan, E. Sybingco, E. P. Dadios, and E. J. Calilung, "Common garbage classification using mobilenet," *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pp. 1–6, 2018.
[8] N. Patel, "EVS-Project." https://github.com/NancyPatel11/EVS-Project, 2024.