INTELLIGENT WASTE MANAGEMENT

Project report submitted in partial fulfillment for the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering/Information Technology

By

Suveer Sharma (191381)

Under the supervision of

Prof. (Dr.) Vivek Kumar Sehgal

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

I hereby declare that the work presented in this report entitled **"Intelligent Waste Management"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Prof Dr Vivek Kumar Sehgal, Professor and Head, Fellow IEI, SM-IEEE, SM-ACM,** Department of CSE Jaypee University of Information Technology, Waknaghat.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Suveer Sharma, 191381.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Prof (Dr) Vivek Kumar Sehgal

Professor and Head, Fellow IEI, SM-IEEE, SM-ACM

Computer Science and Engineering and Information Technology

Dated: 26 April 2023

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ____________________ **Department:** ________________ **Enrolment No** ________

Contact No. ____________________ **E-mail.** ____________________

Name of the Supervisor: ____________________

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ____________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to Lord Shiva for His divine blessing makes it possible for us to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Prof Dr Vivek Kumar Sehgal, Professor and Head, Fellow IEI, SM-IEEE, SM-ACM,** Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of my supervisor in the field of "**Machine Learning**" to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Prof Dr Vivek Kumar Sehgal, Professor and Head, Fellow IEI, SM-IEEE, SM-ACM,** Department of CSE, for his kind help to finish my project.

I would also generously welcome my friend Piyush Singh and each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

No expression of appreciation is complete without recognition of the prayers, good wishes, advice and moral support of my affectionate parents Mr Neeraj Sharma and Mrs Anjana Sharma, which helped me immensely to achieve my goal.

Name: Suveer Sharma

Enrollment No.: 191381

Group No.: 140

TABLE OF CONTENT

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM
ML	Machine Learning
CNN	Convolutional Neural Network
WHO	World Health Organization
UNESCAP	United Nation Economic and Social Commission for Asia and Pacific
VGG	Visual Geometry Group
ResNet	Residual Network
LBP	Local Binary Pattern
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
AI	Artificial Intelligence
SVM	Support Vector Machine
PET	Polyethylene Terephthalate
SIFT	Scale-Invariant Feature Transform
API	Application Programming Interface
ReLU	Rectified Linear Activation Unit
DNN	Deep Neural Network
NAS	Network Architecture Search
CPU	Central Processing Unit
ANN	Artificial Neural Network
CVPR	Computer Vision and Pattern Recognition
k-NN	K Nearest Neighbors
Xception	Extreme Inception
LFW	Labeled Faces in the Wild
MNIST	Modified National Institute of Standards and Technology
CIFAR	Canadian Institute For Advanced Research

LIST OF FIGURES

LIST OF GRAPHS

LIST OF TABLES

ABSTRACT

With production and utilization of resources and other general stuff, wasteful and other unneeded by products are bound to be generated. In India, especially the problem revolving around proper waste management sticks out like a sore thumb. When moving out of Delhi, the National Capital, the famous landfills, which are essentially massive mounds of waste and garbage cannot be missed. Not only in India, but rather globally landfills, ocean waste and debris, waste accumulation and improper waste disposal and management has become a severe problem.

Although the waste production is a problem that we may not be able to tackle with the current level of technology, we definitely can focus and improve the existing infrastructure around waste disposal and management. The unawareness regarding the same is alarming, even the lack of segregating and separating waste into the defined categories, at personal and household-level is obviously of the major component of this nuisance.

For that purpose, we aim to target the problem at the root and present a working solution towards proper segregation of waste before its disposal and subsequent management. Therefore, we propose a project that aims to assist the users in segregating and separating their waste produce in the most convenient way possible. The project proposes to leverage the techniques of Machine learning and Deep Learning through TensorFlow and Keras Convolutional Neural Networks Architecture to analyze the user input and then segregating the waste produce into its different categories. The software thus developed can be deployed using multiple hardware devices, depending on the ease of use and user experience and ubiquity.

CHAPTER 1: INTRODUCTION

1.1 Introduction

Humans are widely regarded as the most intelligent species on the planet since the early ages when men resided in caves. Since then, humans have been excellent at adapting to their environment and creating new things that will aid our survival, be it the first flintstone spear used for hunting or the life-saving drugs that are being used in medical treatments. However, the process of creation not only gives us our required item but also some unneeded byproducts, that end up being the waste. Not only this but, after utilization most of the products end up becoming waste as well. Our current technology is neither advanced nor refined enough such that the production and consumption process do not result in any waste production, thus there is not much we can do about the waste production with the ever-growing population, especially with global scale urbanization underway.

Since not much can be done with regards to the production, waste management is the most crucial component to solve this problem. In the past years, issues revolving around liquid and gaseous waste did come into to limelight due to increased pollution and harm to ecosystem. Feasible solutions were then created and implemented but due to this there was a severe lack of attention to solid waste management, which has now raised its head and ends up being on top of the list of waste management and disposal nuisances.

According to the World Health Organization, waste management starts from generation to collection, treatment and finally disposal. According to the United Nation Economic and Social Commission for Asia and Pacific, waste disposal guidelines are based on the source and type of material waste is made from. Thus, identifying the components and material of waste becomes the key to for proper waste management and disposal.

The proposed model is meant for industrial implementation so as to assist in segregating and separating the waste produce in the most convenient way possible. The project proposes to leverage the techniques of Machine learning and Deep Learning through TensorFlow and Keras modules in combination with CNN Infrastructure to analyze the input and then segregating the waste produce into its different categories. The project follows an both approaches of a comparative study between the different models which later assisted in the ensemble modelling app-roach used for the final outcome/output.

1.2 Problem Statement

Over the past decade, a lot of focus has been on reducing the pollution, especially air and water pollution. In efforts for reducing these two, many measures were implemented that ended up being advantageous to liquid and gaseous waste management as well. But due this excess of limelight on these two aspects, solid waste management and disposal was severely neglected. This not only led to improper solid waste management but also a lack of awareness regarding the same among the masses. This issue has now become a global nuisance and immediate measures for the same have become a dire need.

Marine litter and ocean debris in the Pacific Ocean, has been coined as the Great Pacific Garbage Patch. It actually is two distinct collections of debris in the North Pacific Ocean. At least about 14 million tons of plastic waste ends up in ocean every year due to improper disposal and ocean dumping.

In India as well the problem revolving around proper waste management sticks out like a sore thumb. When moving out of Delhi, the National Capital, the famous landfills, which are essentially massive mounds of waste and garbage cannot be missed. More than 150 acres of land around the National Capital Region has been dedicated to these landfills and even then, the city can’t seem to be able to deal with its waste management and disposal problems. The

Municipal Corporation estimates these landfills to have about 28 million tons of solid waste piled up over the years.

Improper waste disposal and management, including landfills, ocean waste and debris, and waste accumulation, has become a significant problem not only in India but also globally. The issue of waste mismanagement has resulted in severe environmental challenges, with landfills and oceans being overwhelmed with waste, leading to pollution and ecological damage. This problem requires urgent attention and proper waste management practices to mitigate its detrimental impact on the environment and human health worldwide.

This issue becomes an even bigger problem when the guidelines already in place are not followed as well. The unawareness regarding the same is alarming. People do not know about the categories of solid waste and the instructions to be followed while managing and disposing it. Even the lack of segregating and separating waste into the defined categories, so that the disposal is not further complicated is obviously of the major component of this nuisance.

1.3 <u>Objectives</u>

Awareness regarding solid waste management in India is very less and with the largest population in the world, this will be a major problem in the near future. The primary objective of the proposed project is to provide the user/facility where the project has been deployed with classification of the input data i.e., waste sample, and instructions for its proper disposal and management. As opposed to the currently existing infrastructure that heavily depends on human intervention, our project aims to reduce the manual work that goes into segregation of waste by employing machine learning and deep learning technique through CNN architecture. The initial focus will be on identifying and classifying the waste items into different categories one item at a time and then train and test the model. In order to further develop the project, classification of items into further subcategories and their disposal instructions can be given.

1.4 Methodology

The data set is picked from GitHub repository(WasteImagesDataset) [1] which was gathered by cardstdani for research, non-commercial use only. This is a set of merged datasets that contains 5000+ images of waste images that can be used in classification and segmentation problems. The dataset is divided into 9 classes: ‘Aluminum', 'Carton', 'Glass', 'Organic Waste', 'Other Plastics', 'Paper and Cardboard', 'Plastic', 'Textiles', 'Wood'

Data pre-processing is done as it is the foremost thing to achieve better results. Null values are removed from different columns and replaced with the mean or mode of that column. Some null values are also dropped. Outliers are removed. Some columns are normalised and data binning is done on some columns. A method of organising several more or less continuous values into fewer "bins" is called binning. The correlation between different columns is found and information is gathered on how one column is related to another.

We split our data into two datasets (train and test). Each of them is only used in the corresponding phase of the project (training and evaluation of the model). This operation is critical to make the model capable of generalizing from the provided data to any input that the user will use in production. The usual splitting rates are 70/30, 80/20, and 90/10 for training and testing, respectively.

After this, we implement eight different Deep CNN models are implemented on the dataset, considering different attributes to build a model that helps in accurate classification of the sample input.

The models are compared using different performance parameters like accuracy, precision, recall and f1-score. To enhance the accuracy of the model, stacking ensemble technique is applied. Stacking is an ensemble technique that teaches the model to integrate predictions made by learner models with predictions made by meta-models to create a final model with accurate predictions. The fundamental advantage of stacking ensembles is that they can protect a variety of effective models' abilities to address classification and regression issues.

Multi-Class Stack Ensembling is applied. The top three performing models, ResNet50, EfficientNetSmall and EfficientNetLarge are ensembled by taking the Mode of their output classes. The ensemble model created has an accuracy of 99.34%.

1.5 Organization

There are 5 chapters in this project report. The First Chapter introduces the main issue and describes the problem in its entirety, provides a brief overview of our proposed model and suggest techniques that can be employed for further study and implementation. The literature survey and review that has key for us to determine our approach has been presented in Chapter 2. Chapter 3 consists of the dataset description, data preprocessing and details of the proposed model framework. Software design methodologies, tools and models employed, system and software requirements and schedules have been covered under Chapter 4. The Final Chapter of the report summarizes the project evaluation, advantages and future scope.

CHAPTER 2: LITERATURE SURVEY

Huynh et al. [2] detailed the accuracy and effectiveness of the different CNN models like the VGG, ResNet and EfficientNet to tackle the issue of waste categorization. They followed a comparative approach and later created an ensemble model that achieved the accuracy of 94.11%

In [3], the authors have analyzed and compared the currently existing research studies presented around the world that deal with classification of garbage using image processing. They have followed a comparative approach while listing out all the drawbacks faced by the already existing systems and algorithms.

The authors of [4] have applied CNN to deal with the classification of plastic waste They have proposed a system for classifying the waste into four classes: polyethylene terephthalate, high-density polyethylene, polypropylene and polystyrene. They achieved 91.72% and 96.41% accuracy in 15-layer and 23-layer model used by them.

Adedeji et al. [5] proposed a 50-layer ResNet pre-trained CNN model and SVM to classify the waste based on the materials like glass, metal, paper, plastic, etc. They tested the model the on the trash image dataset which was developed by Gary Thung and Mindy Yang and were able to achieve an accuracy of 87%.

The authors of [6] have used the computer vision approach to classify the garbage images into six classes: glass, paper, metal, plastic, cardboard, and trash. The models used are SVM with SIFT and CNN.

In [7], the authors have proposed a method of combining LBP and CNN. This model gets aims to get rid of the disadvantages of poor stability of CNN greyscale and identify the trained network more effectively, achieving an accuracy of 96.6%.

Krizhevsky et al. [8] trained a large, deep CNN to classify 1.2 million high-resolution images into 1000 different classes at the ImageNet LSVRC 2010 contest. To reduce overfitting in the connected layers, dropout regularization method was used.

In [9], a smartphone application was developed as a trash tracking and reporting tool for citizens to coarsely segment and identify garbage in their neighborhoods using images. The authors of the application used a dataset obtained from Bing Image Search and trained their network using patches extracted from the images. To improve generalization, a pre-trained AlexNet model was utilized, resulting in a mean accuracy of 87.69%. The authors made effective use of the pre-trained model to enhance the performance of their application.

Liu et al. [10] used the Flickr Materials Database for an image-based material classification project. Features including SIFT, color, micro texture, and outline shape were utilized within a Bayesian computational framework to attempt classifying images based on material classes.

A similar project can be seen in [11], where during the 2016 TechCrunch Disrupt Hackathon, a team developed "AutoTrash", a smart trash can that uses a Raspberry Pi powered module and camera to automatically sort between compost and recycling. The project utilized Google's TensorFlow framework and incorporated hardware components. However, its limitation is that AutoTrash focuses solely on classifying items as either compost or recycling.

He et al. [12] introduce Parametric Rectified Linear Units (PReLU) for image classification, improving model fitting with minimal computational cost and overfitting risk. It also proposes a robust initialization method for training extremely deep rectified models from scratch. The approach achieves a top-5 test error of 4.94% on ImageNet 2012, surpassing human-level performance and outperforming GoogLeNet by 26%.

In [13], the Malik et al. present an algorithm for partitioning grayscale images into disjoint regions based on coherent brightness and texture. Contour and texture cues are combined using a gating operator, and a spectral graph theoretic framework is used to find partitions of the image into regions with similar texture and brightness. The approach is evaluated on various images, and experimental results are shown.

In [14], a novel training criterion for deep neural networks is introduced that aims to minimize classification error while maximizing the interval between minimum and maximum errors. The approach combines cross-entropy and M3CE methods after analyzing their effectiveness. The proposed criterion, M3 CE-CEc, is tested on standard databases MNIST and CIFAR-10, demonstrating improved results compared to cross-entropy alone. Experimental findings suggest that M3 CE enhances cross-entropy and serves as an effective supplement to the traditional criterion.

In [15], Sakr et al. focus on automating waste sorting using machine learning techniques, specifically deep learning with CNN and SVM. The SVM achieved higher accuracy (94.8%) compared to CNN (83%), and showed adaptability to different waste types. The SVM model was implemented on a Raspberry Pi 3 and demonstrated quick classification, averaging 0.1 seconds per image.

Intelligent Waste Separator (IWS) is introduced in [16]. It uses multimedia embedded architecture, Image invariant moments for image analysis and object representation, and k-NN as effective lazy-classifiers to deal with the machine learning aspect to automatically sort incoming waste into different containers, aiming to replace traditional waste management methods.

In [17], the impact of convolutional network depth on image recognition accuracy has been studies. Using small 3x3 convolution filters, it was discovered that increasing network depth to 16-19 layers improves performance beyond prior-art configurations. These models also achieved top results in ImageNet Challenge 2014 and generalized well to other datasets.

Aral et al. [18] tested various deep learning models were tested on the Trashnet dataset for waste recycling classification. Densenet121, DenseNet169, InceptionResnetV2, MobileNet, and Xception architectures were used with Adam and Adadelta optimizers. Adam provided better test accuracies, and data augmentation was applied to increase accuracy. The best results were achieved with DenseNet121 and InceptionResNetV2, both achieving test accuracies of 95% and 94% respectively.

The authors of [19] propose a hybrid deep-learning system (MHS) that uses a high-resolution camera and sensors to automatically sort waste in urban public areas. The MHS combines CNN-based image feature extraction with multilayer perceptrons (MLP) for classifying recyclable and non-recyclable waste. The system achieves over 90% accuracy in classifying waste, outperforming a reference CNN-based method that only uses images for classification.

Taigmainet et al. [20] propose a face recognition pipeline that includes 3D face modeling for alignment and a deep neural network with over 120 million parameters trained on a large facial dataset. The method achieves high accuracy of 97.25% on the LFW dataset, outperforming the current state of the art by over 25% and approaching human-level performance in unconstrained environments.

In [21], the dense convolutional network (DenseNet) is introduced as a novel approach that connects each layer to every other layer in a feed-forward manner, unlike traditional convolutional networks. This architecture addresses the vanishing-gradient problem, promotes feature propagation and reuse, and reduces the number of parameters. DenseNets achieve superior performance on challenging object recognition benchmarks (CIFAR-10, CIFAR-100, SVHN, and ImageNet) compared to state-of-the-art methods, while also requiring less memory and computation.

Few of the research papers studied in-depth for the purpose and benefit of this project have been summarized in the table below

Table 1: Literature Review Summarized

S. No.	**Author (s)**	**Approach**	**Datasets**	**Results**
1	Hyunh et al. (2020) [2]	CNN like VGG, ResNet101, EfficientNet-B0 and EfficientNet-B1	Trashnet dataset from Stanford that was developed by Gary Thung and Mindy Yang	94.11% with Ensemble model

2	Flores et al. (2019) [3]	Deep Learning, AI sorting Techniques, SVM and CNN	Dataset with about 500 images of waste and garbage was created.	SVM and CNN have achieved the highest Accuracy of 94%
3	Bobulski et al. (2019) [4]	CNN, AlexNet structure	PET Waste Classification Method and Plastic Waste DataBase - WaDaBa	91.72% and 96.41% accuracy in 15-layer and 23-layer model
4	Adedeji et al. (2019) [5]	50-layer pre-trained ResNet CNN model and SVM	Trashnet dataset from Stanford that was developed by Gary Thung and Mindy Yang	87% Accuracy on the dataset
5	Yang et al. (2016) [7]	SVM with SIFT, CNN	Trashnet dataset from Stanford that	Accuracy of 63% was achieved on the SVM with 70:30 dataset split

			was developed by Gary Thung and Mindy Yang	
6	Wang et al. (2017) [7]	Local Binary Patterns (LBP) with CNN	ORL Database, YALE Database, FERET Database	Accuracy of 96.6% was achieved
7	Krizhevsky et al. (2012) [8]	5-layer CNN with 1000-way SoftMax and dropout regularization	ImageNet LSVRC-2010 contest	Top-5 test error rate of 15.3%
8	Mittal et al. (2016) [9]	Smartphone application was developed based on a pre-trained AlexNet Neural Network	Dataset was obtained from Bing image Search	Mean accuracy of 87.69% was achieved
9	Donovan et al. (2016) [11]	Auto-trash, a smart trash can that incorporated a Raspberry Pi module	Dataset of TechCrunch	Could only classify trash into two categories:

		functioning on Google's TensorFlow framework and a camera	and Disrupt Hackathon	Compost and Recycling
10	Sakr et al. (2016) [15]	SVM and CNN, with implementation of the SVM model on Raspberry Pi3 module.	A dataset of 2000 images, were collected and artificially augmented to 6000 images to avoid overfitting the training set	The SVM model achieved 94.8% accuracy and demonstrated quick classification, averaging at 0.1 second per image.
11	Aral et al. (2018) [18]	DenisenseNet169, InceptionResnetV2, MobileNet, Xception models in combination with Adam and Adadelta optimizers were used.	Trashnet dataset from Stanford that was developed by Gary Thung and Mindy Yang	DenseNet121 gave a test accuracy rate of 95% using fine-tuning. InceptionResNetV2 model using fine-tuning gave test accuracy of 94%.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 Dataset Description

The data set is picked from GitHub repository(WasteImagesDataset) which was gathered by cardstdani for research, non-commercial use only. This is a set of merged datasets that contains 3500+ images of waste images that can be used in classification and segmentation problems. The dataset is divided into 9 classes: ‘Aluminum', 'Carton', 'Glass', 'Organic Waste', 'Other Plastics', 'Paper and Cardboard', 'Plastic', 'Textiles', 'Wood'. The collection includes real-world images that can be used to accurately identify waste belong to which class and is it dry or wet.

Each image in the dataset is a single-object, which is not severely damaged to facilitate the recognition and classification process of the models. In other words, most of the images are intact as recognition of shredded or charred (too damaged) trash is a challenging task for humans let alone machines.

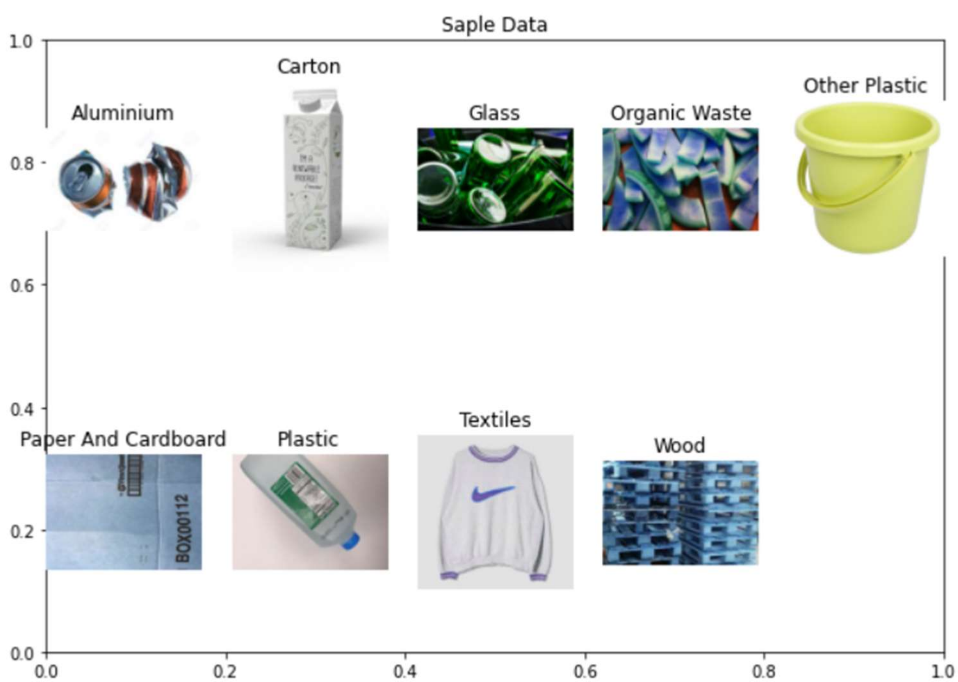


Figure 1: Sample Data

The first class labeled as “Aluminum” contains 550 images of aluminum and metal waste (cans, food packages, utensils, etc.). The second class labeled as “Carton” contains 336 images of carton waste (milk cartons, tetra packs, etc.). The third class labeled as “Glass” contains 550 images of glass waste (glass bottles, jars, water glass, etc.). The fourth class labeled as “Organic Waste” contains 210 images of organic waste.

Table 2: Breakdown of the Dataset

Serial Number	**Class**	**Images Count**	**Images Example**
1	Aluminium	550	cans, food packages, utensils, etc
2	Carton	336	milk cartons, tetra packs, etc
3	Glass	550	glass bottles, jars, water glass, etc

4		Organic Waste	210	Kitchen Waste, Fruits, Vegetables, etc
5		Other plastic	340	Thermosetting plastic, bucket, high quality bottles
6		Paper and Cardboard	550	Newspaper, Scrap Paper, Cardboard
7		Plastic	493	Straws, Thermoplastic, etc
8		Textiles	335	Clothes, Curtains, Bedcovers, Mattresses
9		Wood	347	Chairs, Tables, Broken Furniture

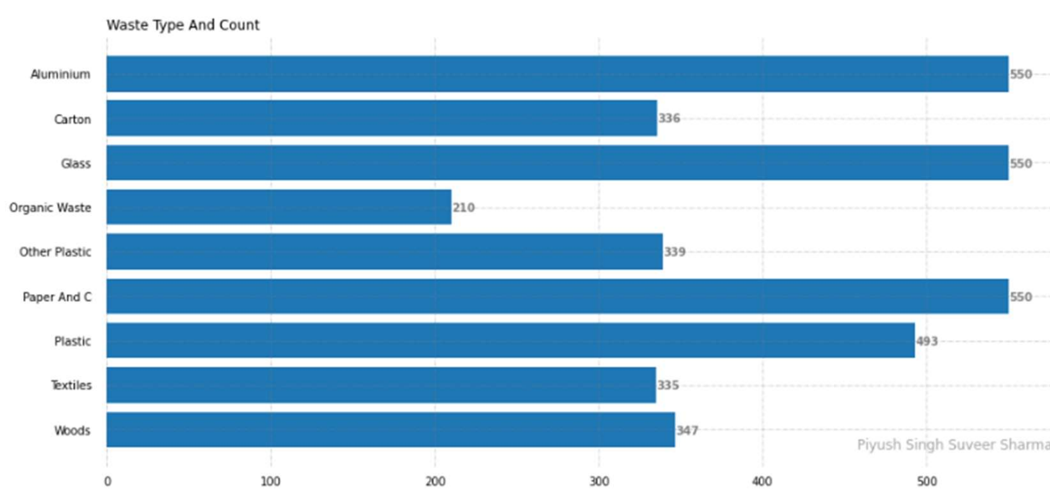
These classes scan be further categorized into two class Dry (Organic Waste) and Wet (All other classes except Organic waste)

3.2 Data Preprocessing and Data Visualization

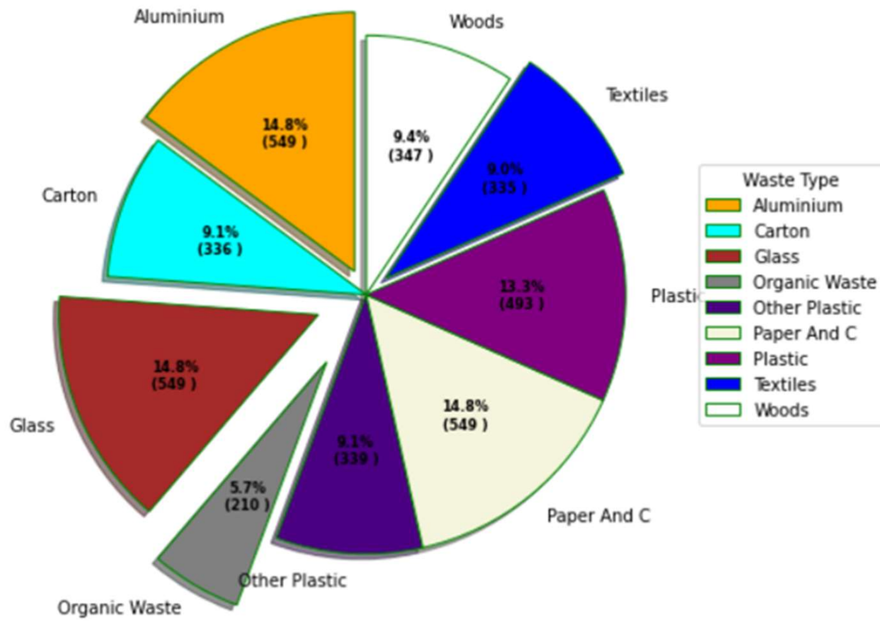
We split the original dataset into two new datasets, one for training the model and other for testing. Each of them will only be used in their corresponding phase of the project training and later evaluation of the model. This operation is critical to make the model capable of generalizing from the provided data to any input that the user will use in production. The usual splitting rates are 70:30, 80:20, and 90:10 for training and testing, respectively.

We have used the Keras API of the TensorFlow library to preprocess the dataset located into a folder by resizing all the images to a standard dimension of 256x256 and setting a batch size of 128, meaning that in the training process, the data will pass through the network in chunks of 128 images.

we store the number of classes in a variable extracting it from the train dataset object (9 classes in this case) and use tf.data.AUTOTUNE to optimize the performance of both training and testing dataset objects.



Graph 1: Waste Type and Count



Graph 2: Pie chart of dataset description

3.3 Brief Description of CNN Techniques

In this section, we have discussed the principle behind CNN and the various classification architectures that are employed for the proposed model. Before the final ensemble of top performing models, other Classifier models were attempted. On the training data set, 8 distinct classifiers were trained. Following the first training, three models were chosen based on their accuracy measure.

A CNN is a type of deep learning neural network that comprises of node layers, categorized as an input layer, one or more hidden layers, and a final output layer. Each of these nodes is connected to one another and have associated weights and threshold values. Once the output exceeds this threshold value, the node is activated, sending the data to the next layer.

The CNN have three main types of layers:

- Convolutional layer

  The convolutional layer is the core of any CNN. Most of the computations occur here and require input data, a filter, and a feature map. The movement and computation of a kernel across the receptive field of the image is known as convolution. Post-convolution, an activation function, usually ReLU, is applied to the feature map so as to introduce nonlinearity into the model and counter the problem of vanishing gradient.

- Pooling Layer

  Pooling layers, also known as downsampling, is responsible for the dimensionality reduction by reducing the number of parameters of the input. Likewise, to the convolution layer, pooling operation also passes a kernel over the entire input. This kernel implements an aggregation function on the values within the receptive fields, populating the output array. There are two main types of pooling- Max pooling (maximum value in the receptive field is sent to the output array) and Average pooling (average value within the receptive field is sent to the output array).

- Fully Connected layer

  In a fully connected layer, each node in the output layer is directly connected to a node in the previous layer. This layer is responsible for classification based on the features extracted from the previous layers and their filters/kernels. This layer usually utilizes softmax as its activation function.

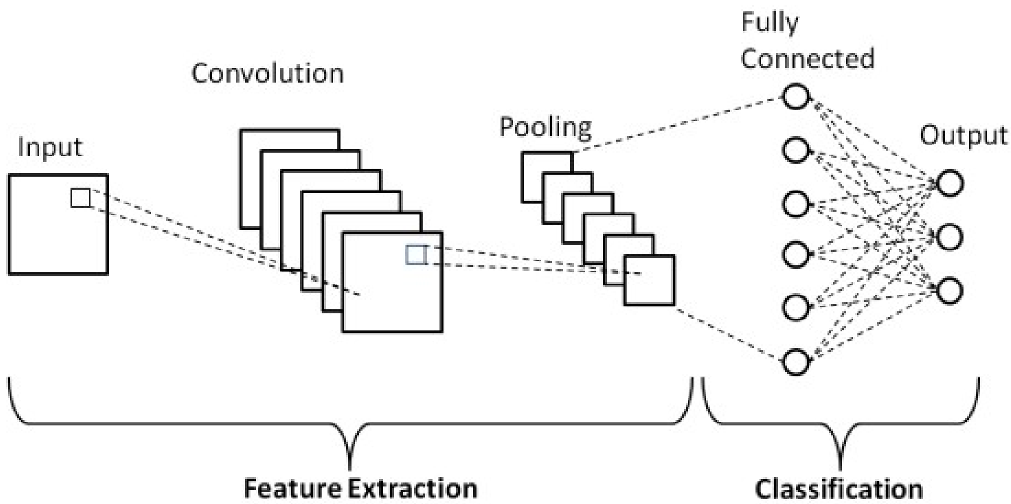


Figure 2: CNN Architecture Overview

3.3.1 Inception [22]

An Inception network is a deep neural network (DNN) employing recurring modules known as inception modules. In this DNN, output of one block is used as the input for the next block. The DNN consists of a 1X1 convolution layer, 3X3 convolution layer, 5X5 convolution layer, a max-pooling layer and finally a concatenation layer that combines the features extracted by the inception blocks.

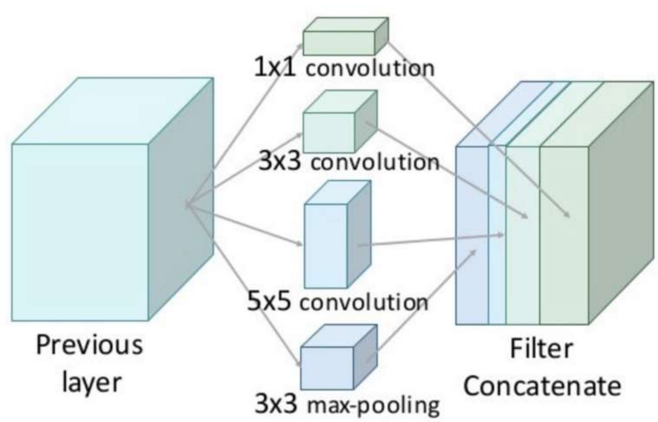


Figure 3: Inception Architecture

3.3.2 Xception [23]

The Xception network is a DNN also known as "extreme inception" that goes beyond the limitations of Inception. The DNN follows an approach that resembles depth-wise separable convolution. Xception first applies the filters to each depth map, and then applies 1X1 convolution over the depth to compress the input space.

Another distinction between Inception and Xception is the existence of non-linearity following the initial procedure. In the Inception model, a ReLU non-linearity follows both processes, whereas Xception does not add any non-linearity.

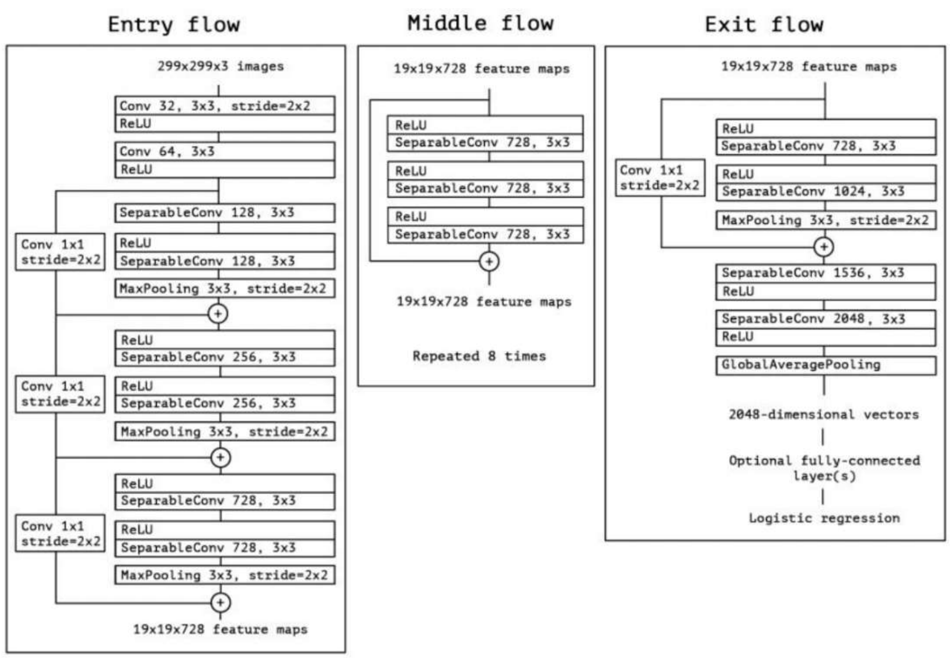


Figure 4: Xception Architecture

3.3.3 VGG16 [24]

The VGGNet is a CNN that supports 16 layers. The convolution layers leverage minimum perceptive fields, followed by linear transformation and ReLU (Rectified Linear Unit) units. Few convolution layers are followed by a pooling layer that reduces the height and width. The hidden layers leverage ReLU and the network only consists of three fully connected layers.

The first and second layers use the most recent feature vector as input and have a channel size of 4096. The third layer creates 1000 channels for 1000 classes. In other words, the third fully connected layer is used to implement the SoftMax function to categorize 1000 classes. Every hidden layer uses ReLU as its activation function. This encourages speedier learning and decreases the possibility of vanishing gradient problems; hence it is more computationally efficient.

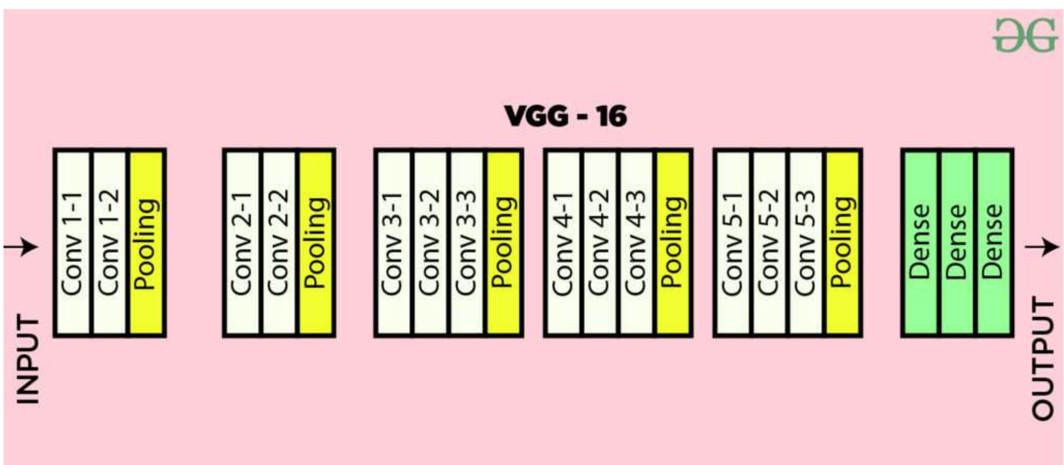


Figure 5: VGG Architecture

3.3.4 MobileNetV3 [25]

MobileNetV3 is a CNN that has been optimized for mobile phone CPUs. This optimization was achieved through a combination of hardware-aware network architecture search (NAS), NetAdapt algorithm for network search, and innovative network improvements. The architecture includes two models, MobileNetV3Small and MobileNetV3Large, which are designed for small and large resource use cases, respectively. These adjustments make MobileNetV3 efficient and effective for running on mobile devices, enabling tasks such as image recognition and object detection with optimized performance.

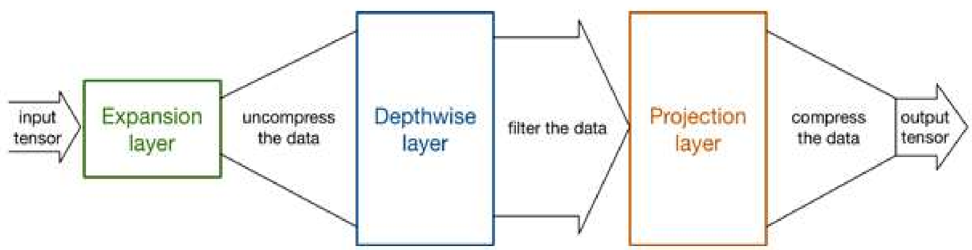


Figure 6: MobileNetV3 Architecture

3.3.5 Resnet [26]

Residual Network (ResNet) is a deep CNN architecture designed to support hundreds and thousands of convolution layers. The ResNet architecture stacks multiple identity mapping, skips those layers, and reuses the activation function of the previous layers. Then, on retraining, the network expands and the remaining parts, known as residual parts, are allowed to explore more of the feature space of the input image. This solves the problem of vanishing gradient that occurs by increasing the number of convolution layers. There are multiple versions of ResNet that use the same basic principle but have various numbers of layers. The model that operates with 50 neural network layers is referred to as Resnet50.

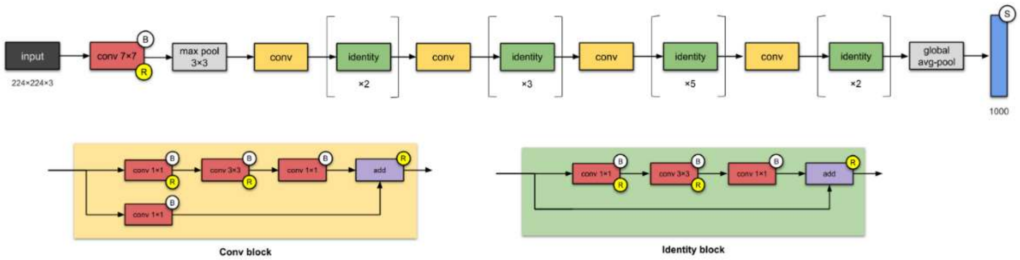


Figure 7: ResNet-50 Architecture

3.3.6 EfficientNet [27] [28]

EfficientNet is a CNN design and scaling technique that uses a compound coefficient to consistently scale all depth, breadth, and resolution dimensions. The EfficientNet scaling method evenly scales network breadth, depth, and resolution using a set of preset scaling coefficients, in contrast to standard practices, which scale these variables arbitrarily. The logic behind the compound scaling method is that larger input images require more layers in order to expand the network's receptive field and more channels in order to capture more fine-grained patterns on the larger picture.

There multiple versions of EfficientNet, EfficientNet-B0 is the baseline model, with its variations being EfficientNet Small and EfficientNet Large implemented in this project.

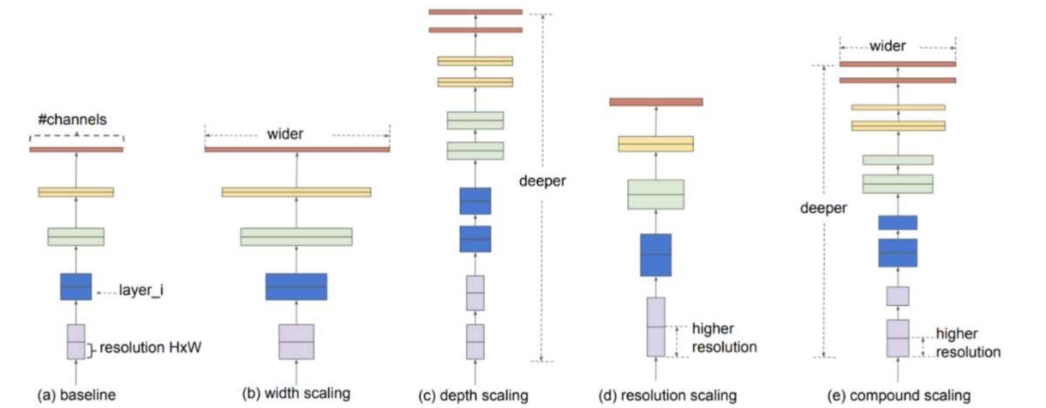


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

Figure 8: EfficientNet Architecture

3.3.7 DenseNet [29]

A DenseNet is a type of CNN that utilizes dense connections between layers through Dense Blocks, where all layers are directly linked with matching feature-map sizes. Each layer in a DenseNet receives additional inputs from all preceding layers and forwards its own feature-maps to all subsequent layers, maintaining the feed-forward nature of the system. This dense connectivity pattern allows for efficient information flow across layers, promoting feature reuse and improving the network's ability to capture complex patterns in the data.

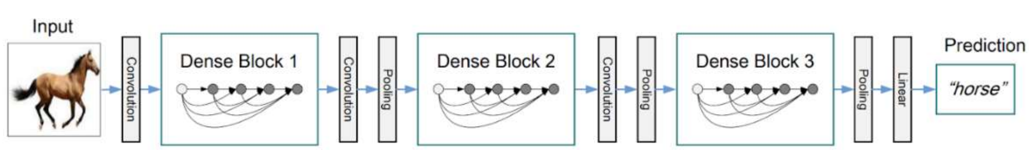


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

Figure 9: DenseNet Architecture

3.3.8 Nasnet [30]

NasNet, short for Neural Architecture Search Network, are models that are automatically generated by training on a specific dataset using neural architecture search techniques. These techniques involve learning the optimal model architecture directly from the dataset of interest, resulting in automatically designed models without human intervention. This approach allows for the creation of customized models tailored to the dataset, enhancing their performance and applicability to specific tasks.

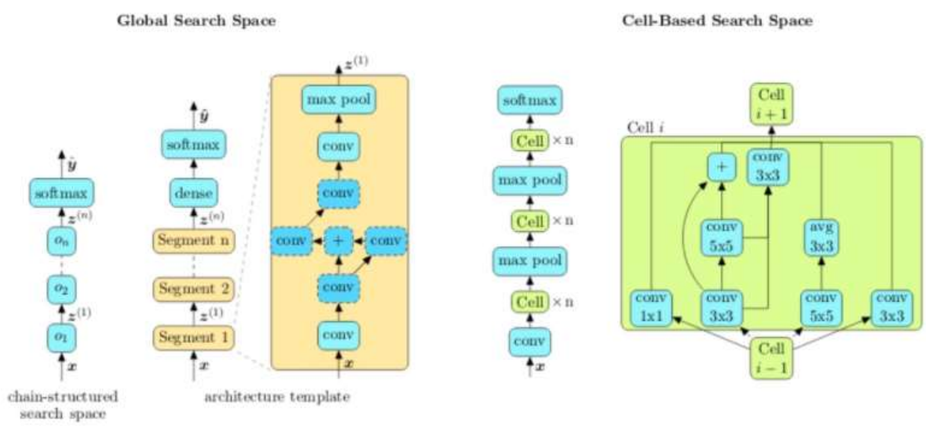


Figure 10: NasNet Architecture

3.4 Transfer Learning

Transfer learning is a powerful machine learning technique that allows a pre-trained model, known as the "source model," to be utilized as a starting point for training a new model on a different but related task or domain. Rather than commencing the training process from scratch, transfer learning leverages the knowledge and learned representations from the source model to enhance the performance of the target model. This approach proves to be particularly beneficial when the target task has limited labeled data or when training time and resources are constrained.

The source model is usually trained on a large dataset and has learned general features or representations from the data. These learned features can capture useful patterns and representations that are transferable to other related tasks or domains. The target model, also referred to as the "transfer model," is then trained using a smaller dataset that is specific to the new task, while leveraging the knowledge from the source model. There are two main ways to implement transfer learning: using the pre-trained model as a feature extractor or fine-tuning its weights.

In the feature extraction approach, the pre-trained model is employed as a fixed feature extractor. The input data is passed through the layers of the pre-trained model to extract relevant features, which are then fed into a new set of layers that are specific to the target task. These new layers are trained using the limited labeled data available for the target task, while the weights of the pre-trained model remain unchanged. This way, the pre-trained model acts as a feature extraction tool that provides useful representations of the input data for the target model to learn from.

In the fine-tuning approach, the pre-trained model's weights are updated during the training process of the target model. The entire model, including the pre-trained layers, is fine-tuned using the labeled data available for the target task. The weights of the pre-trained model are adjusted to adapt to the nuances of the target task, while the new layers specific to the target task are also updated during the training process. This allows the pre-trained model to learn task-specific representations that are optimized for the target task.

Transfer learning offers several advantages over traditional machine learning approaches. One of the main benefits is improved model performance. By leveraging the knowledge learned from a large dataset in the source domain, the target model can benefit from the general features and representations that are transferable to the target task, even with limited labeled data available for training. This can result in faster convergence and better generalization performance of the target model.

Another advantage of transfer learning is reduced training time and resource requirements. Training a deep neural network from scratch on a large dataset can be computationally expensive and time-consuming. By utilizing a pre-trained model as a starting point, transfer learning can significantly reduce the amount of training time and resources needed to train a new model on a related task. This makes transfer learning a practical approach in scenarios where computational resources are limited.

Furthermore, transfer learning allows for leveraging domain knowledge from the source task to the target task. The source model has already learned useful features and representations from the data, which can capture domain-specific patterns and knowledge. This domain knowledge can be transferred to the target task, providing a head start in learning relevant representations and improving the performance of the target model.

Additionally, transfer learning can help address data scarcity challenges. In many real-world scenarios, obtaining a large labeled dataset for a specific task can be challenging or expensive. Transfer learning allows the target model to benefit from the knowledge learned from a larger dataset in the source domain, even when labeled data is limited for the target task. This can be especially valuable in fields such as healthcare, finance, or agriculture, where obtaining labeled data may be difficult.

Despite its benefits, transfer learning requires careful consideration of the similarity between the source and target tasks or domains. The effectiveness of transfer learning depends on the similarity of the source and target tasks or domains. If the tasks are not closely related, the transfer of knowledge may not be effective, and the performance of the target model may not be improved.

Furthermore, the choice of the transfer learning approach, i.e., feature extraction or fine-tuning, also requires careful consideration. Feature extraction is a safer option, as it keeps the pre-trained model's weights fixed and only trains new layers for the target task. This minimizes the risk of negative transfer, as the pre-trained model's representations remain unchanged. On the other hand, fine-tuning allows the pre-trained model's weights to be updated during training, which can be beneficial in certain cases, but also increases the risk of negative transfer if the source and target tasks have significant differences. Therefore, the choice of approach should be based on the specific characteristics of the source and target tasks, and the available data for the target task.

Therefore, it is crucial to carefully evaluate the suitability of the source model for the target task and domain before using transfer learning. Negative transfer, small labeled data size, data quality, and the choice of transfer learning approach are important factors to be considered. Proper evaluation and experimentation are essential to ensure the effectiveness of transfer learning in a specific application.

3.5 Ensembling Approach

Ensemble modeling is a powerful technique in machine learning that combines the predictions of multiple models to improve overall performance and accuracy compared to using a single model. The concept behind ensemble modeling is based on the idea that different models may have complementary strengths and weaknesses, and by combining their predictions, the resulting ensemble can achieve better results.

Ensemble modeling can be applied to a wide range of machine learning tasks, including classification, regression, and anomaly detection. There are several popular methods for creating ensembles, including bagging, boosting, and stacking.

3.5.1 Bagging

Bagging, short for Bootstrap Aggregating, involves training multiple instances of the same base model on different subsets of the training data. These subsets, known as "bags," are created through random sampling with replacement. Each model in the ensemble is trained on a different bag, and the predictions of the individual models are then combined, for example, by taking a majority vote in case of classification or averaging in case of regression. Bagging can help reduce overfitting and improve the overall accuracy and stability of the ensemble.

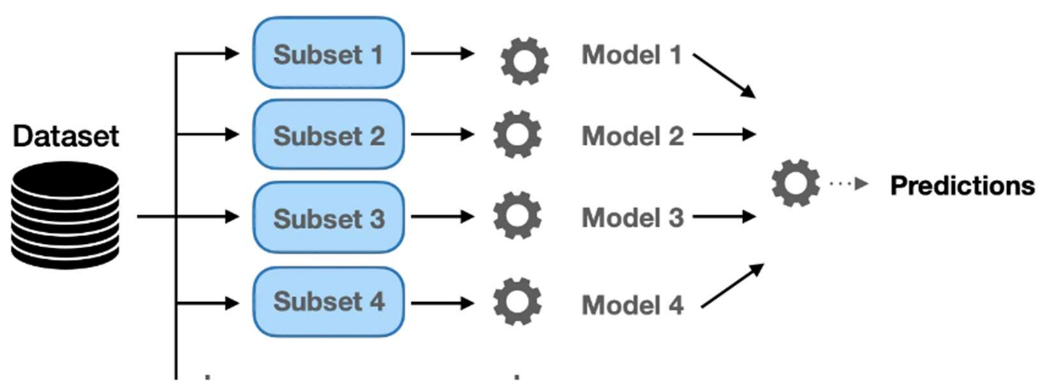


Figure 11: Bagging Technique

3.5.2 Boosting

Boosting is an iterative technique that adjusts the weights of the training samples to give more importance to misclassified samples. The idea is to focus on the samples that are difficult to classify and give them more attention in subsequent iterations. Multiple base models are trained in a sequence, with each subsequent model focusing on the samples that were misclassified by the previous models. The predictions of the individual models are combined using weighted voting, where the weights are determined based on the performance of the individual models. Boosting can often result in highly accurate ensembles, as it focuses on improving the performance on misclassified samples.

Figure 12: Boosting Technique

3.5.3 Stacking

Stacking is also known as meta-modelling. It involves training multiple base models on the same training data and then using their predictions as input to a higher-level model, called the meta-model, which makes the final prediction. The idea is to capture the strengths of different base models and combine them to improve the overall performance of the ensemble. Stacking can be done in multiple layers, with predictions from one layer of base models being used as input to the next layer of base models, and so on, until the final prediction is made by the meta-model.

Figure 13: Stacking Technique

For combining the multiple predictions and outputs that we get from the multiple ensemble models and algorithms, we make use of multiple aggregation methods. The three main techniques used for aggregation predictions are:

- Max-Voting

  Max voting aggregation technique involves combining the predictions from multiple base models by taking a majority vote on the predicted class with the highest number of votes, resulting as the aggregated prediction.

- Averaging

  Averaging is an aggregation technique where the predicted class probabilities or outputs from multiple base models are averaged to obtain a single aggregated prediction. It involves combining the outputs of different models by taking the average, which can help to reduce noise and improve overall prediction accuracy.

- Weighted Averaging

  Weighted averaging aggregation is a technique where the predicted class probabilities or outputs from multiple base models are combined by taking a weighted average, where each model's prediction is multiplied by a weight. The weights are assigned based on the performance or expertise of the individual models, allowing for more influence from better-performing models, and resulting in a single aggregated prediction.

Ensemble modeling has been proven to be highly effective in improving the accuracy, robustness, and generalization of machine learning models. Few reasons that lead to better performance of an ensembled model compared to using a single model are:

- First, ensemble modeling can help mitigate the impact of overfitting, which is a common issue in machine learning where the model performs well on the training data but poorly on new, unseen data. By combining the predictions of multiple models, the ensemble can generalize better to new data and reduce the risk of overfitting.

- Second, ensemble modeling can help capture different sources of information and exploit the strengths of different models. Different models may have different learning biases, and by combining their predictions, the ensemble can achieve a more balanced and accurate prediction.

- Third, ensemble modeling can improve the robustness of the model to noisy or erroneous data. In real-world scenarios, data can be noisy or contain outliers, which can negatively impact the performance of a single model. Ensemble models, by combining the predictions of multiple models, can reduce the impact of noise or outliers and make more robust predictions.

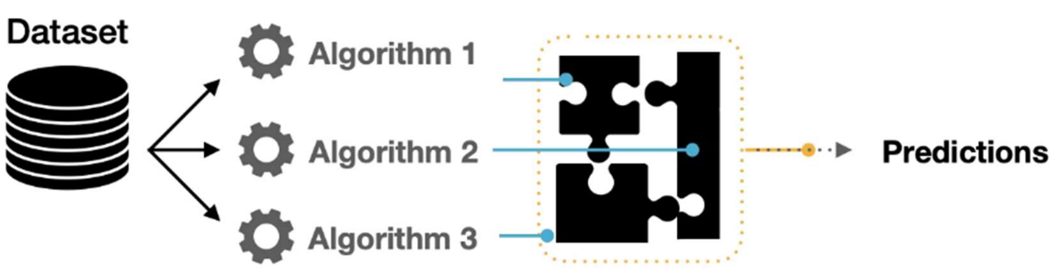


Figure 14: Ensemble Modelling

3.6 Proposed Model

In our model we have ensembled 3 best performing models Resnet50, Efficientnet-small and Efficientnet-large.

The Algorithm of our proposed model is shown:

- Collect the Dataset And divide them into 9 Classes
- Pre-process Dataset by resizing all the images to a standard dimension of 256x256 and setting a batch size of 128, meaning that in the training process, the data will pass through the network in chunks of 128 images.
- Evaluate all the different CNN architectures on this dataset
- Choose three best performing CNN architectures
- Ensemble three chosen architecture
- Evaluate the ensembled model
- Build and test the ensembled model

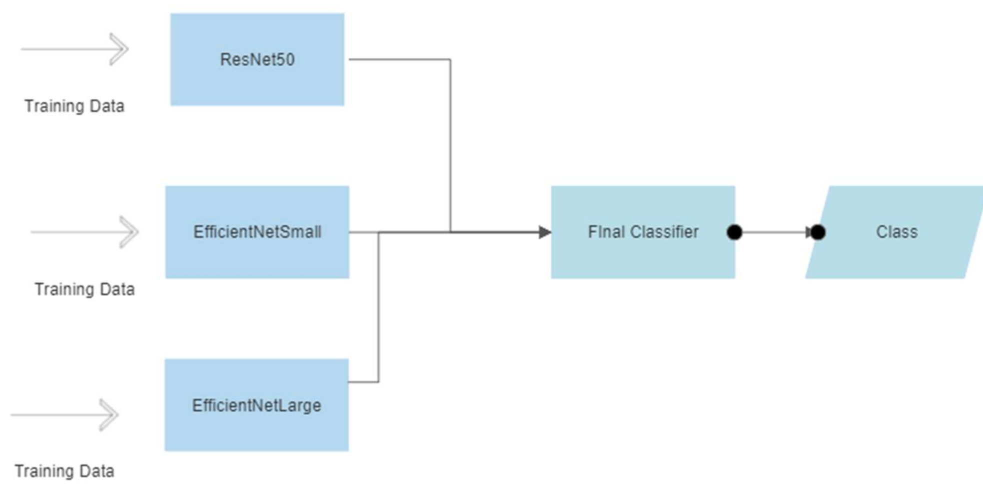


Figure 15: Proposed Ensemble Architecture

CHAPTER-4 EXPERIMENTS AND RESULT ANALYSIS

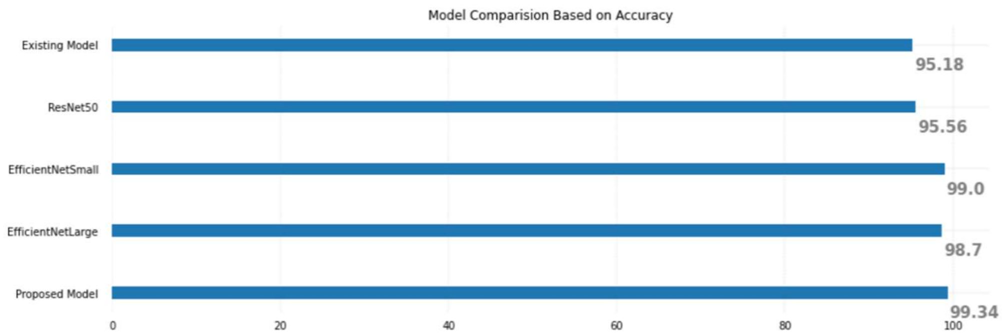
In this part, we examined and analysed the findings of our suggested model. The algorithms were evaluated using a variety of performance indicators. Furthermore, we compared our model to other existing models in terms of accuracy, precision, specificity, sensitivity, precision, F1 Score, AUC-ROC curve. Also analysed the suggested model in relation to other algorithms described in the literature review.

Models which had the highest accuracy were chosen as shown in table.

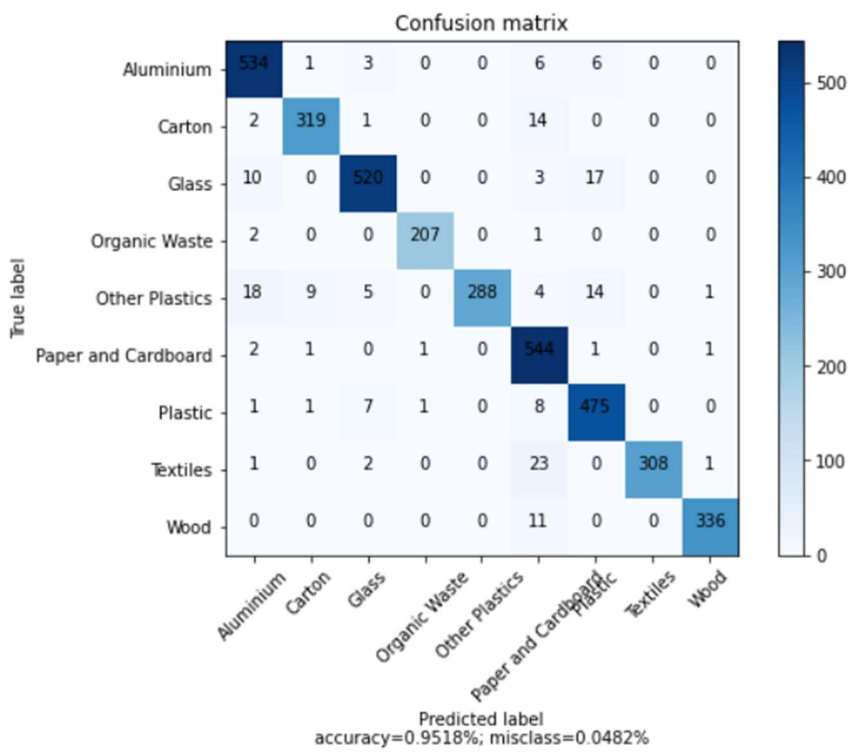
Table 3: Accuracy achievd by all models

S.no	**CNN Architecture**	**Accuracy**
1	EfficientNetSmall	98.3
2	EfficientNetlarge	99.001
3	ResNet50	95.45
4	MobileNetV3	95.50(Extra epochs)
5	NasNet	76
6	Xception	78
7	Inception	73
8	DensNet	82
9	Proposed Model	99.34

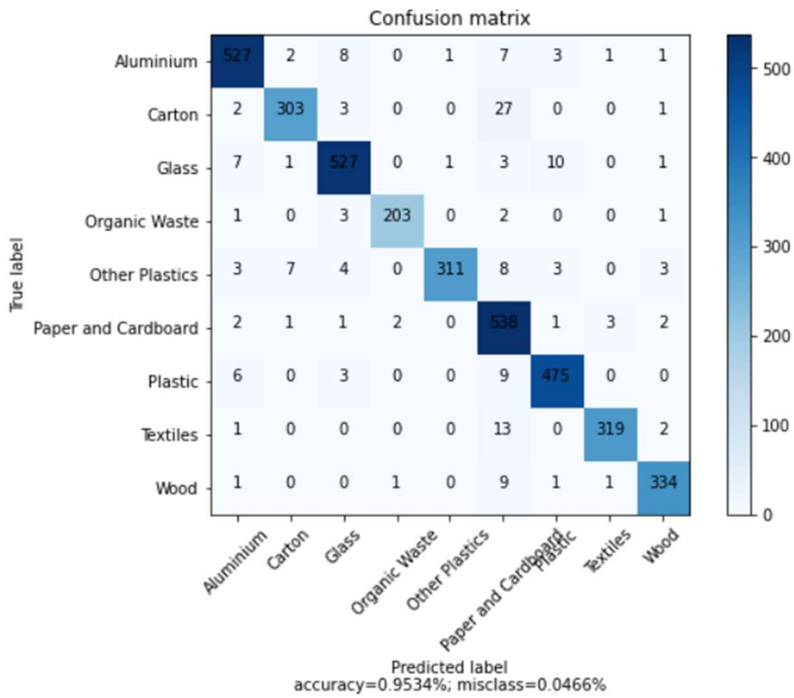
We have chosen ResNet50, EfficientNetlarge and EfficientNetSmall as our base learners and our meta level classifier is extra tree classifier based on these, we got an accuracy of 99.34% and We have compared the values of our model as well as the other models used, shown in table.



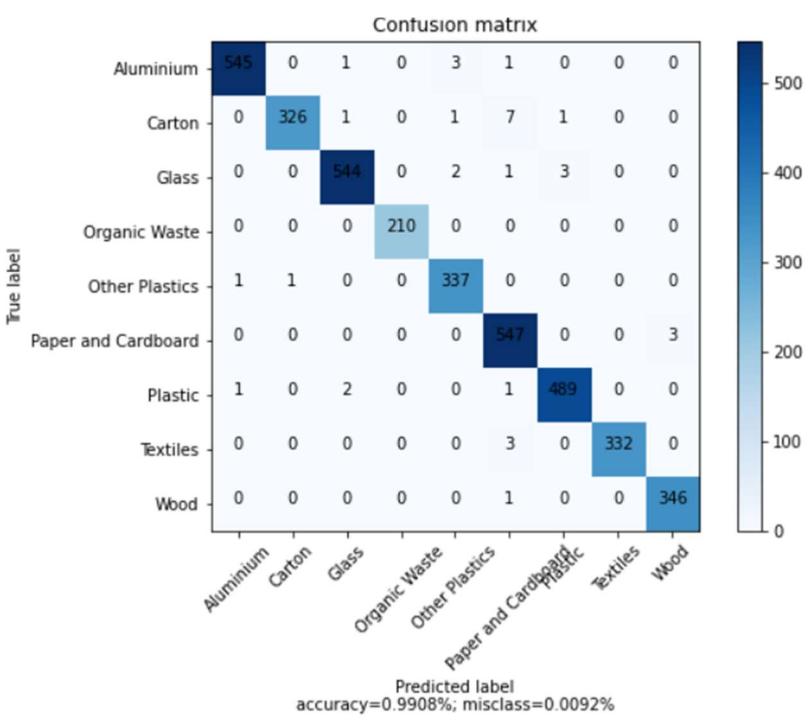
Graph 3: Model Comparison based on Accuracy



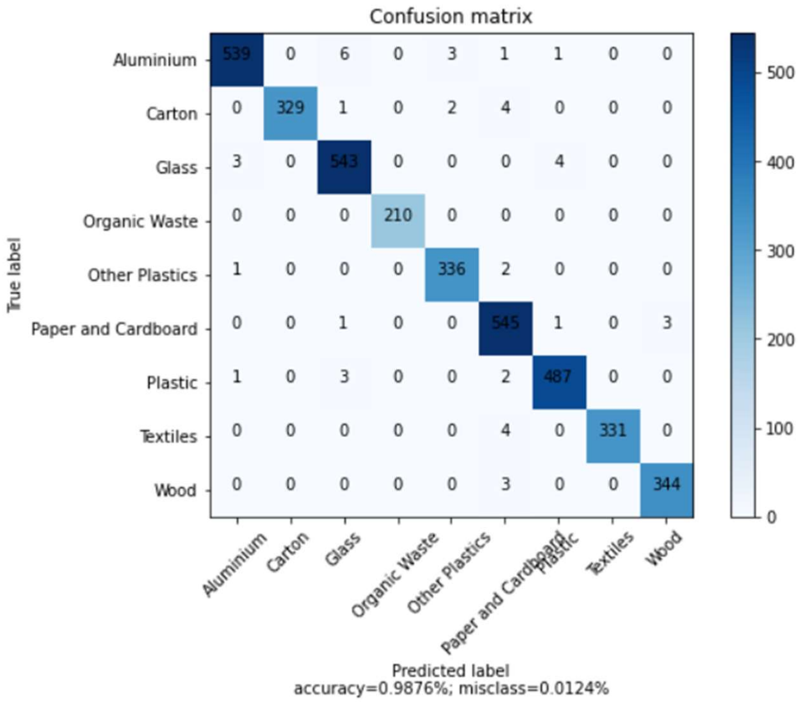
Graph 4: Confusion Matrix for MobileNetV3 model



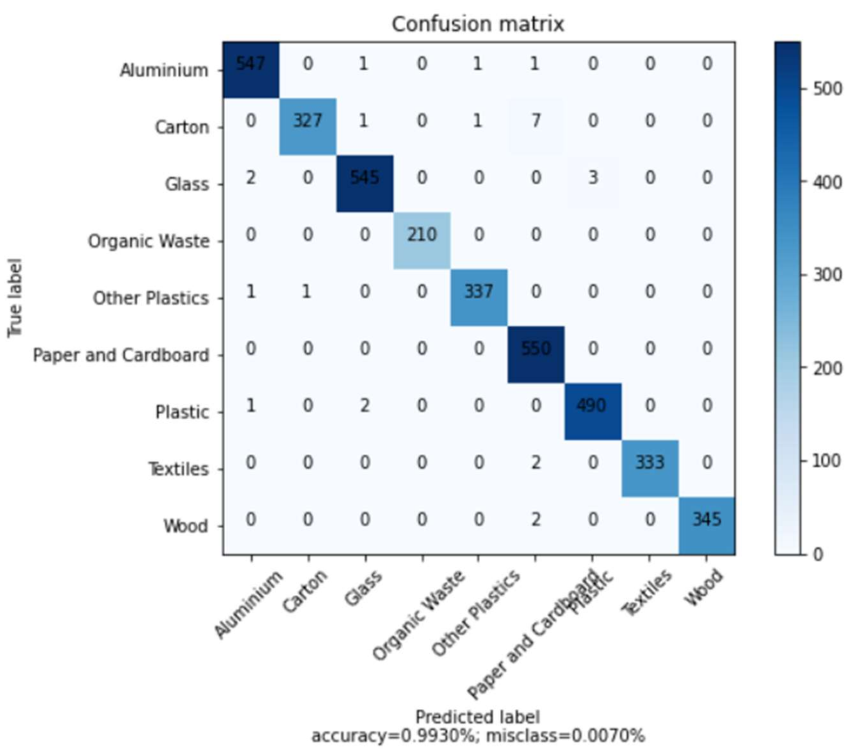
Graph 5:Confusion Matrix for ResNet-50 model



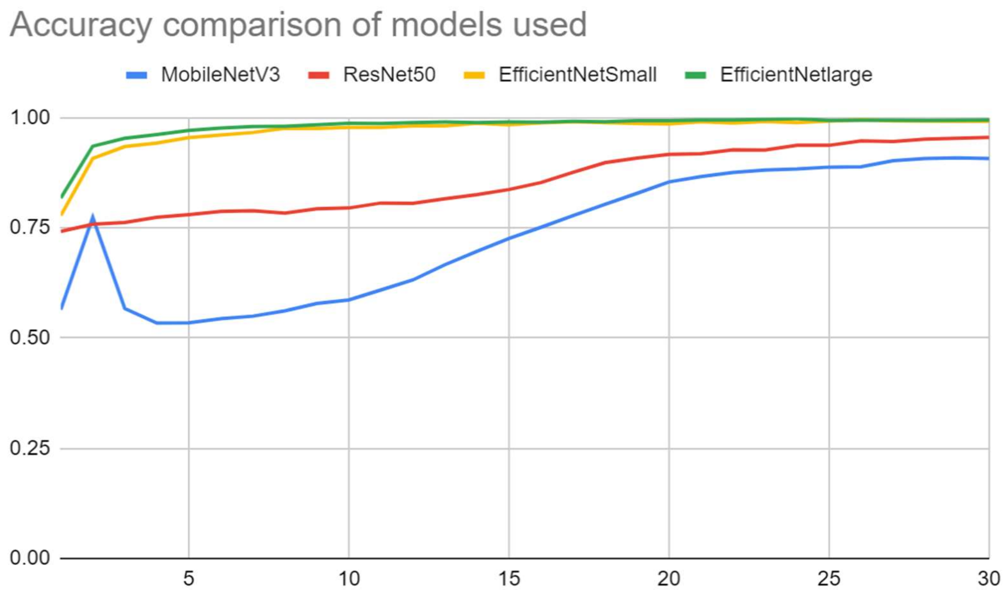
Graph 6:Confusion Matrix for EfficientNetSmall model



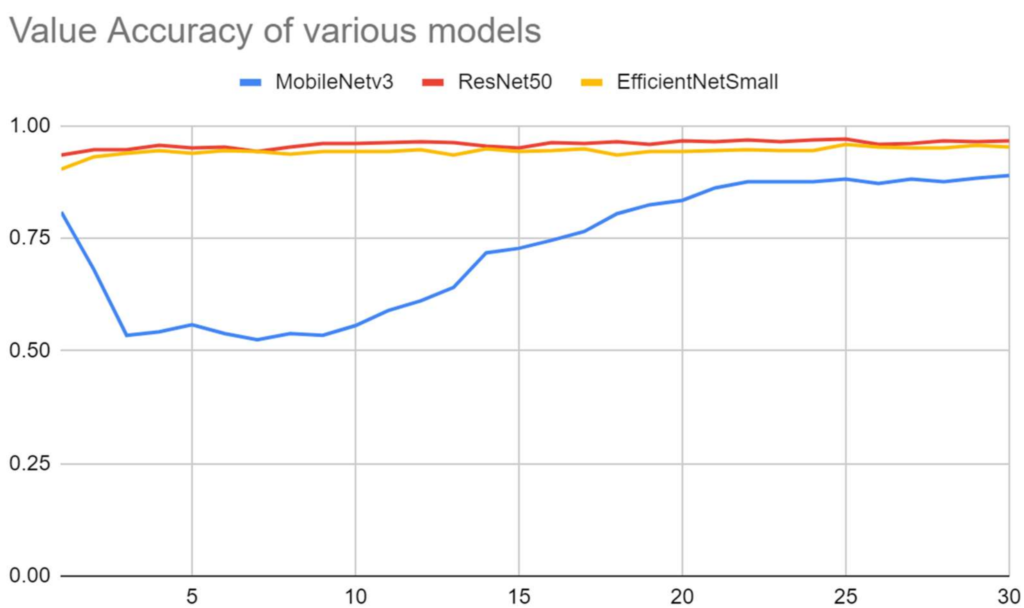
Graph 7:Confusion Matrix for EfficientNetLarge model



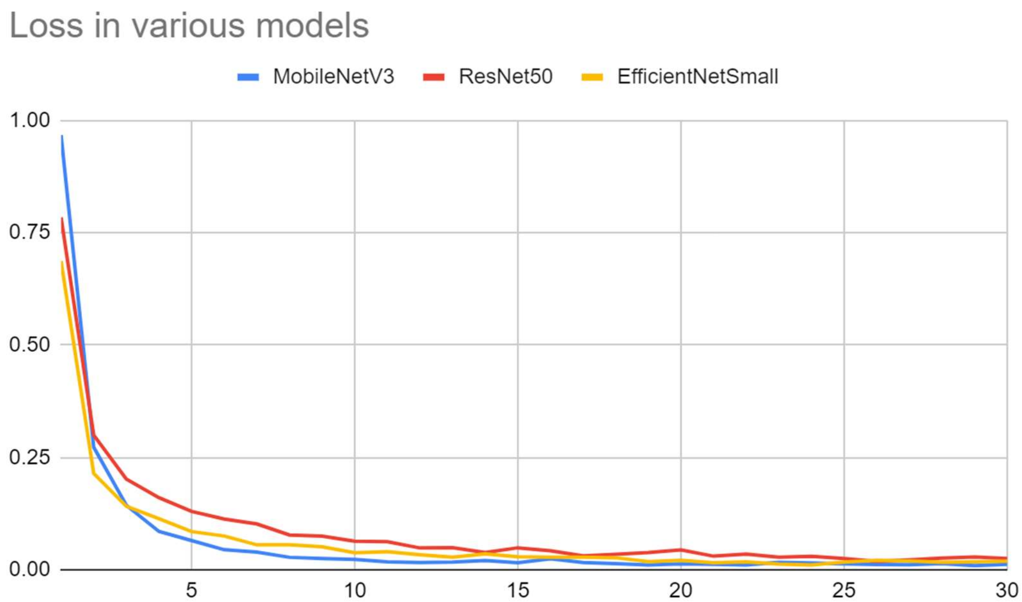
Graph 8: Confusion Matrix for proposed ensemble model



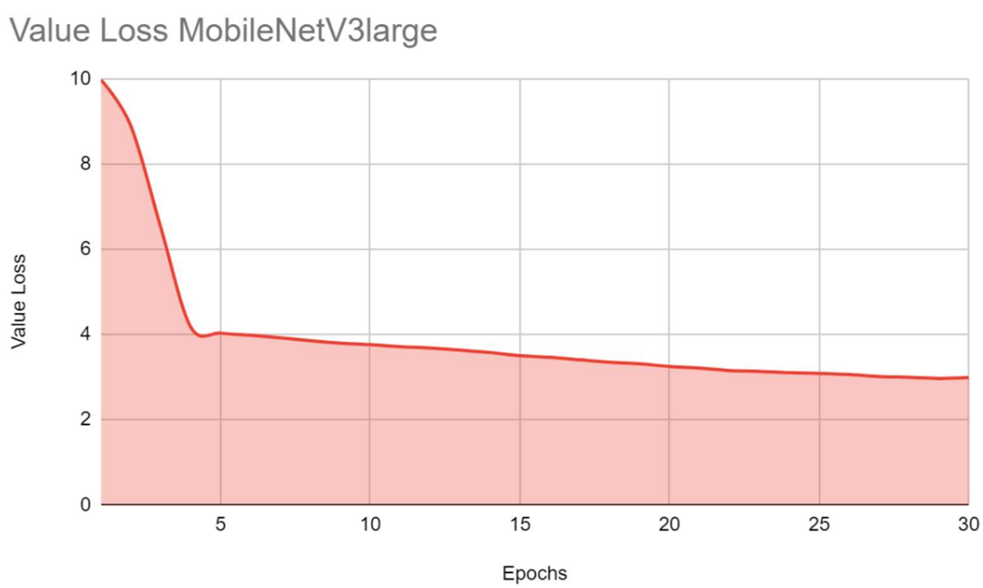
Graph 9: Accuracy comparison of various models



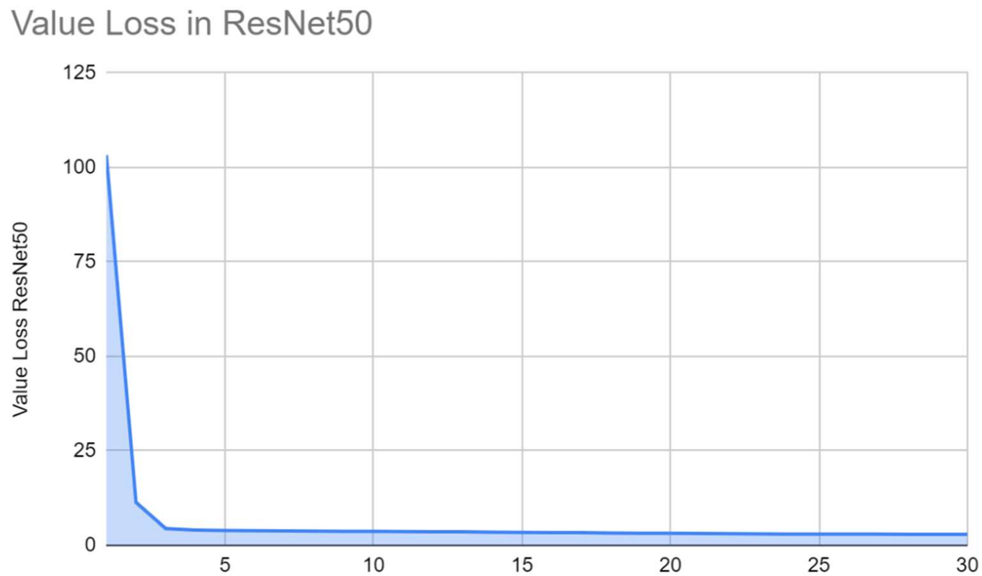
Graph 10: Value Accuracy of various models



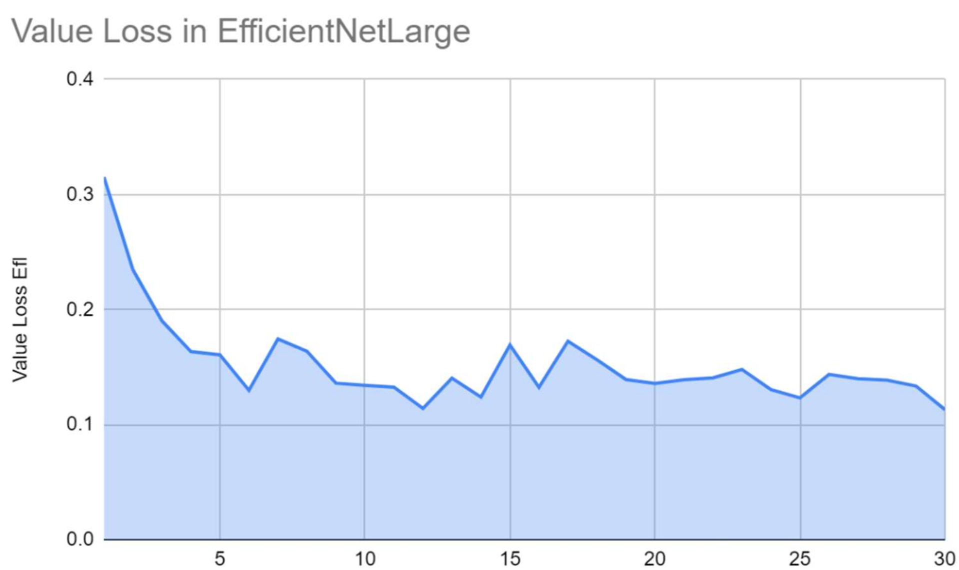
Graph 11: Loss Comparison in various models



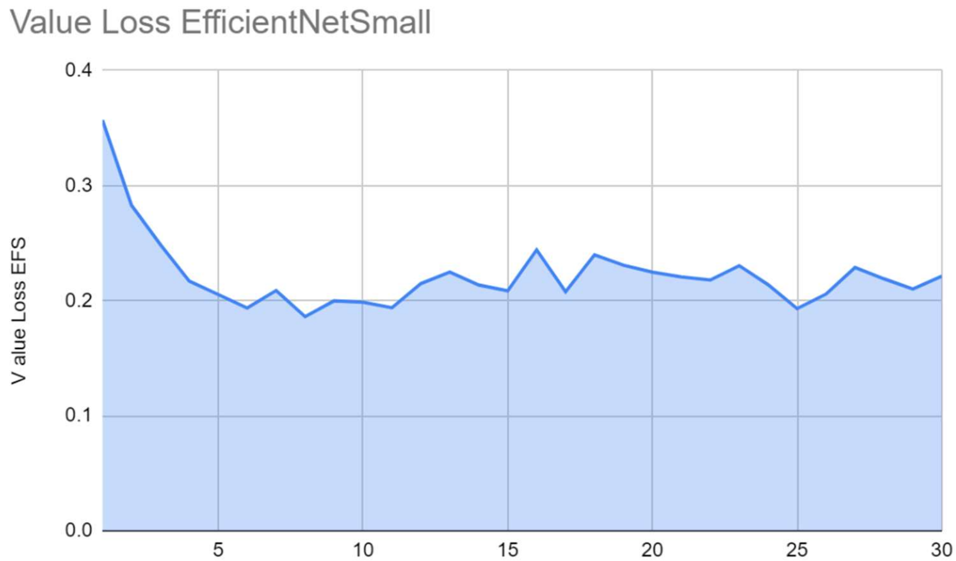
Graph 12: Value loss in MobileNetV3



Graph 13: Value Loss in ResNet



Graph 14: Value loss in EfficientNetlarge



Graph 15: Value loss in EffcientNetSmall

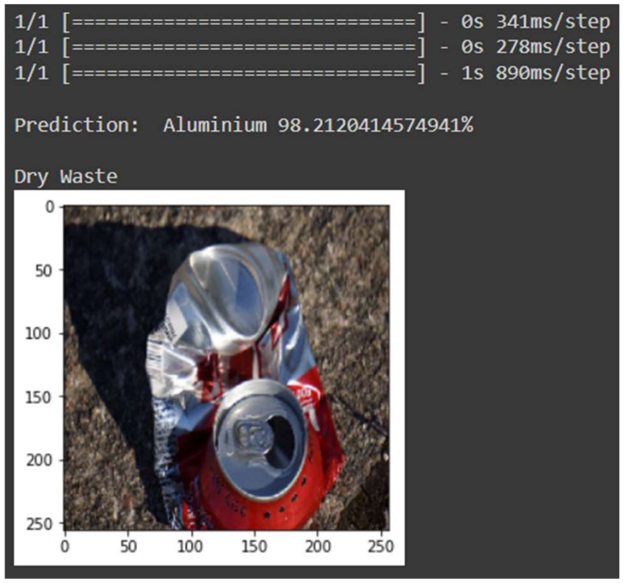


Figure 16: Successful Classification of Aluminum waste by proposed model

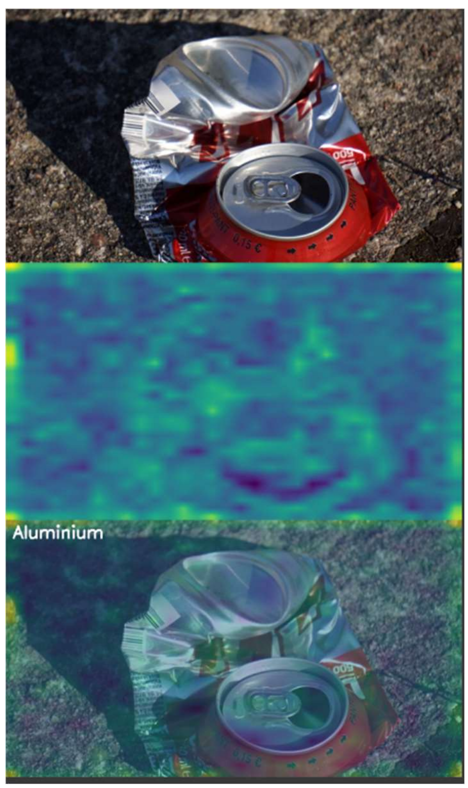


Figure 17:Heatmap of successful Classification of Aluminum waste by proposed model

Figure 18: Successful Classification of Carton waste by proposed model

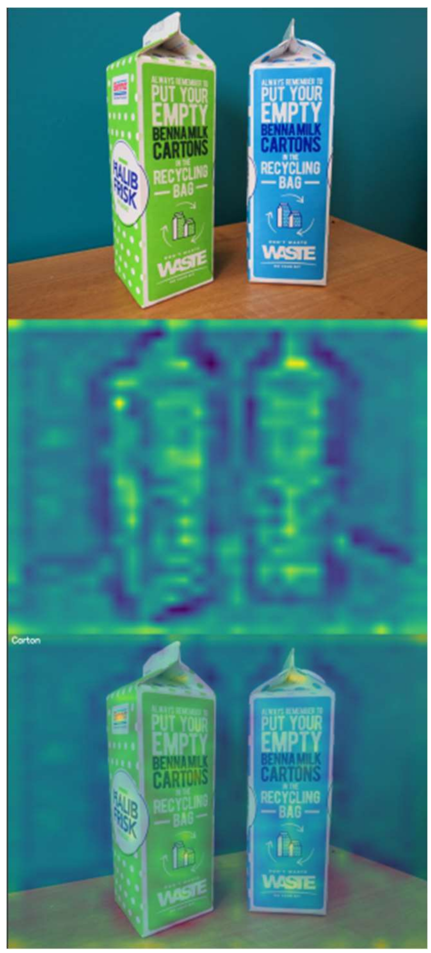


Figure 19:Heatmap of successful Classification of Carton waste by proposed model

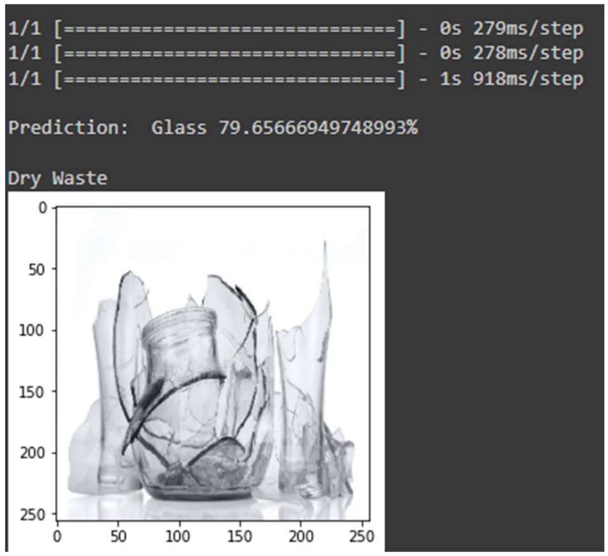


Figure 20: Successful Classification of Glass waste by proposed model

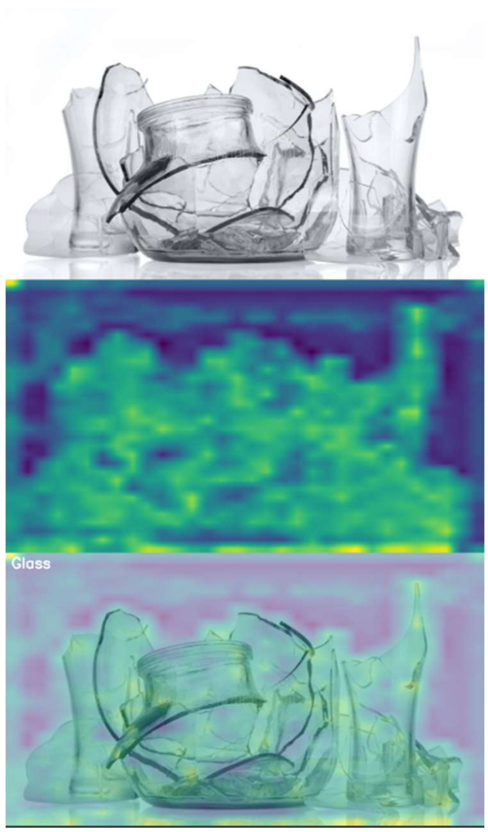


Figure 21:Heatmap of successful Classification of Glass waste by proposed model

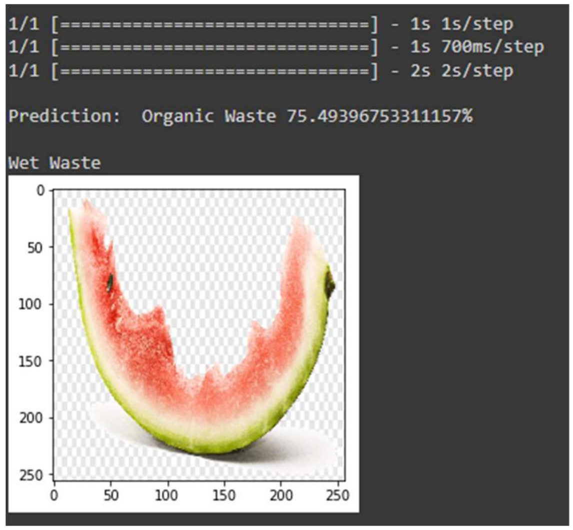


Figure 22: Successful Classification of Organic waste by proposed model

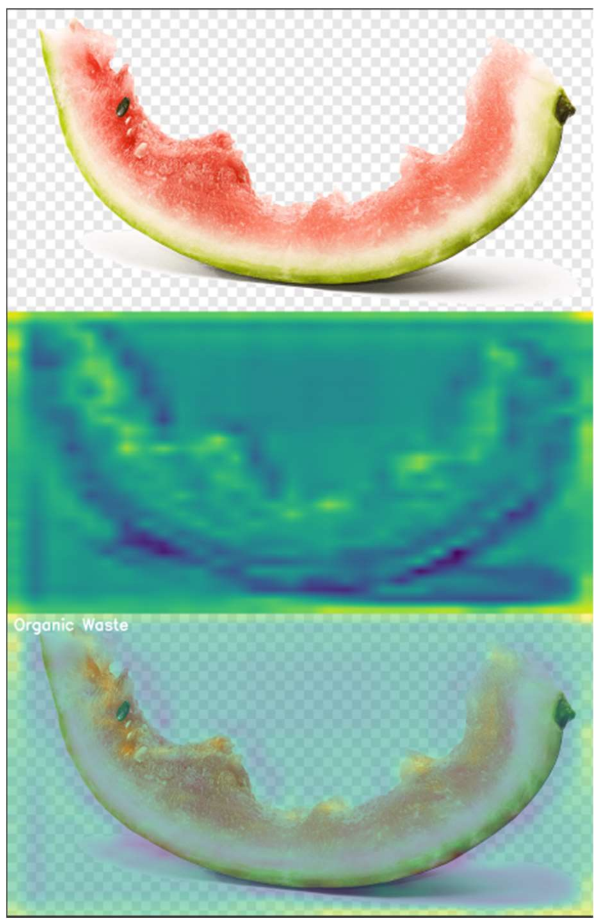


Figure 23:Heatmap of successful Classification of Organic waste by proposed model

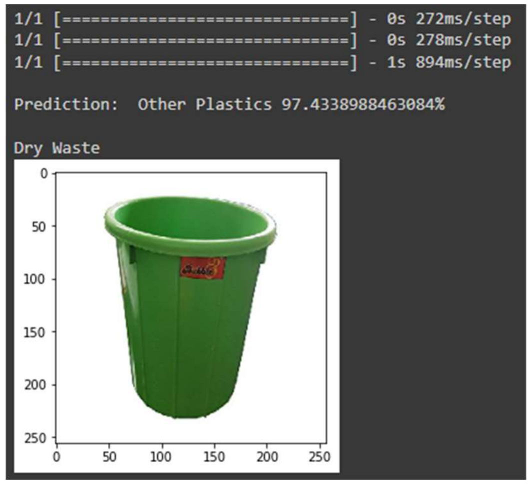


Figure 24: Successful Classification of Other Plastic waste by proposed model

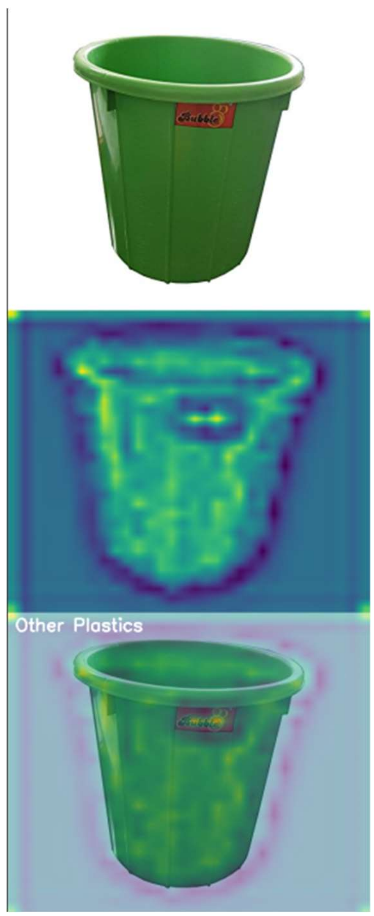


Figure 25:Heatmap of successful Classification of Other Plastic waste by proposed model

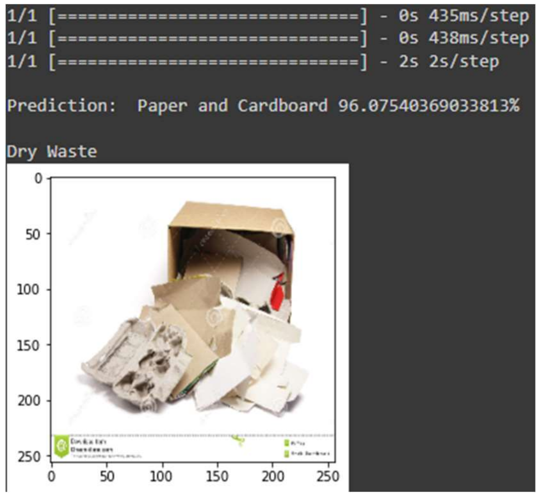


Figure 26: Successful Classification of Paper and Cardboard waste by proposed model

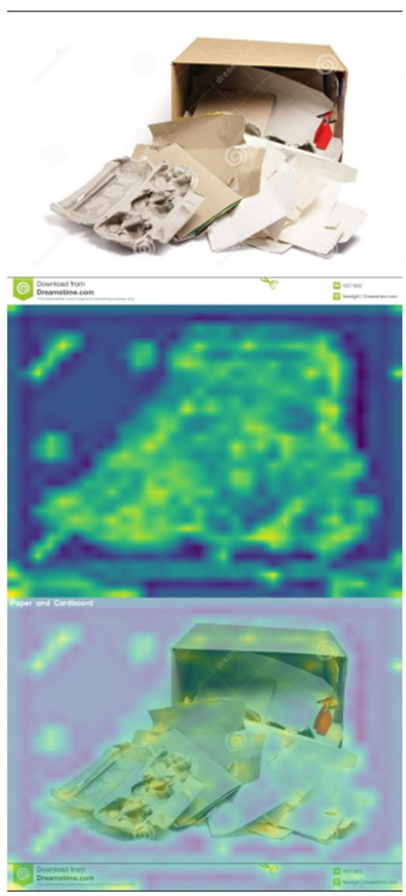


Figure 27:Heatmap of successful Classification of Paper and Cardboard waste by proposed model

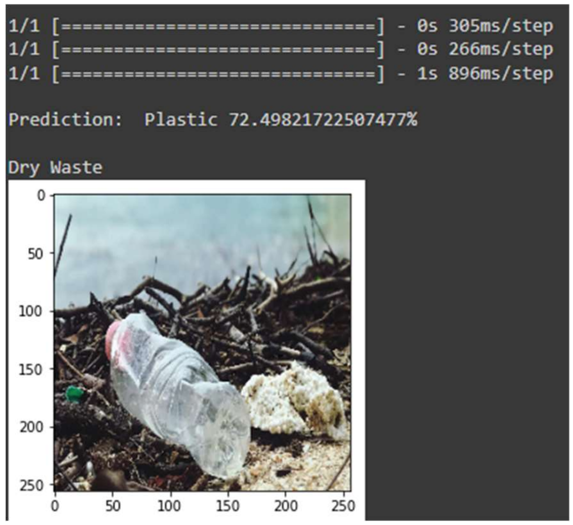


Figure 28: Successful Classification of Plastic waste by proposed model

Figure 29:Heatmap of successful Classification of Plastic waste by proposed model

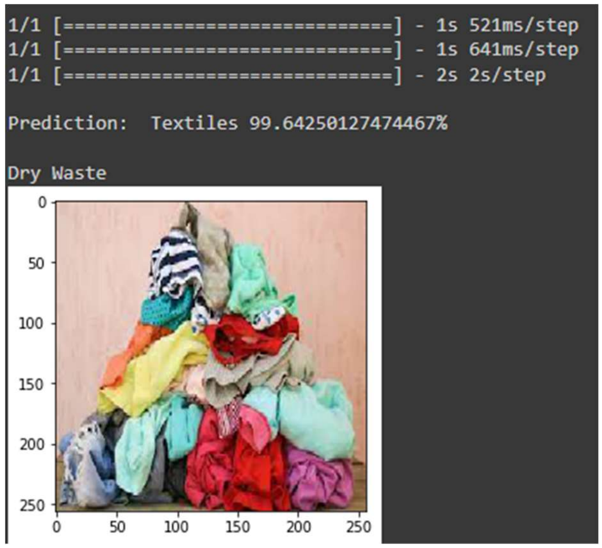


Figure 30: Successful Classification of Textile waste by proposed model

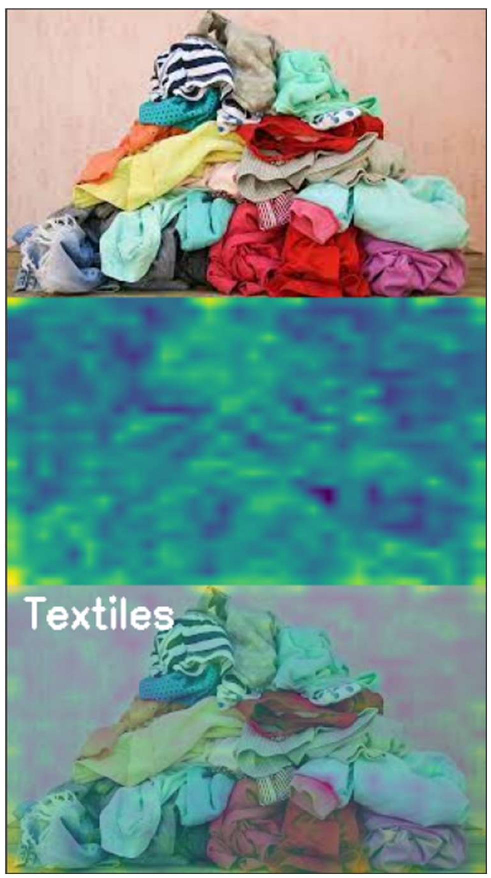


Figure 31:Heatmap of successful Classification of Textile waste by proposed model

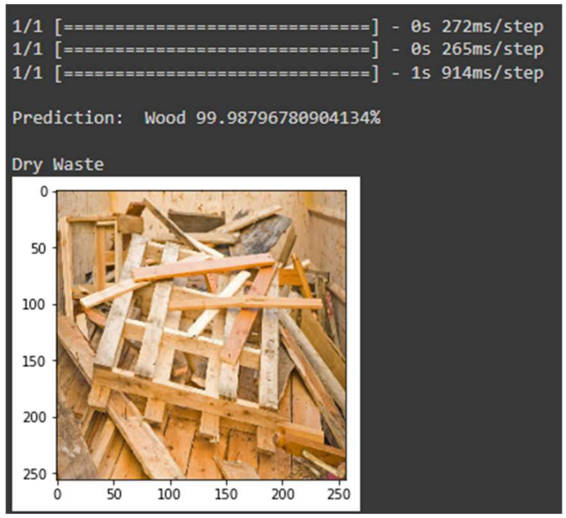


Figure 32: Successful Classification of Wooden waste by proposed model

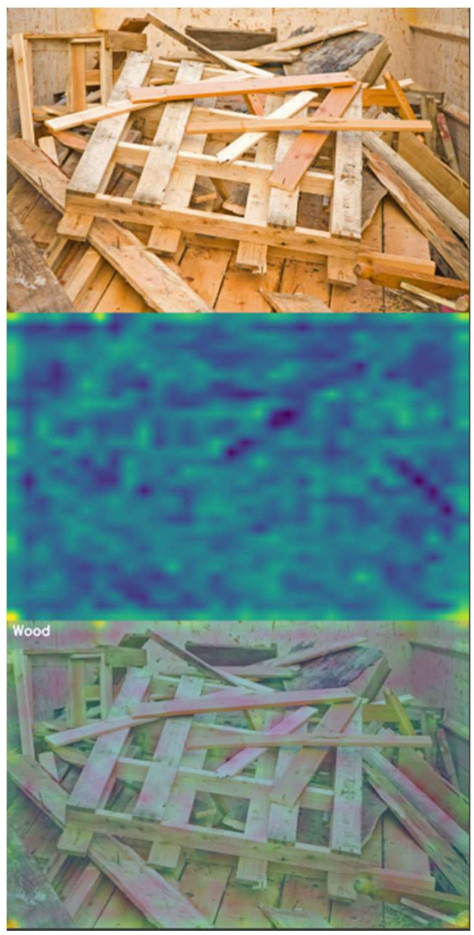


Figure 33:Heatmap of successful Classification of Wooden waste by proposed model

CHAPTER-5 CONCLUSIONS

5.1 Conclusions

Proper Solid Waste Management and Disposal have become very critical as now the world is on the cusp of facing a global crisis. For that classification of the waste into separate categories has become necessary and awareness of how to dispose of these is key. The proposed framework can be used to classify and segregate the waste into different categories based on the material, mainly into 9 classes ‘Aluminum’, ‘Carton’, ‘Glass’, ‘Organic Waste’, ‘Other Plastics’, ‘Paper and Cardboard’, ‘Plasti’, ‘Textiles’, ‘Wood’.

The data set is picked from GitHub repository(WasteImagesDataset) which was gathered by cardstdani and data pre-processing done first. Then various deep CNN models are applied and their accuracy is compared. EfficientNetSmall gives an accuracy of 99.01%, highest achieved among the individual model. The proposed ensemble model is implemented through multiclass stacking technique, by stacking the top three performing models. The models chosen were ResNet-50, EfficientNetSmall and EfficientNetLarge. The proposed model outperformed all the existing models and gave an accuracy of 99.34%. The proposed model also performed better than the existing state-of-art literature. Precision, Sensitivity, Specificity, AUC-Roc curve, Log loss, F1 score are also taken into account to compare the proposed model with already existing models.

5.2 Future Scope

- The model can be improved by collecting information through nearby waste management centres and building our own dataset which would result in a more optimised model.
- Along with k-fold validation technique while implementing ensembling, blending technique can also be applied which might lead to improved results.

- For the purpose of providing all with the means to utilize our findings, a website can be built for solid waste classification. This would involve of input in image format. Then the trained model would therefore classify the waste into the categories.
- To further improve the awareness regarding solid waste management, guidelines and instructions regarding the disposal and management can be provided after successful classification.

5.3 Applications Contributions

We proposed a Stacking based Ensemble Learning model which increases classifier diversity. We proposed a ensembled CNN model which increases classifier diversity and fits better on the given dataset. We take mode of the prediction of 3 best performing CNN architecture. We combined result of ResNet50, EfficientNetSmall and EfficientNetLarge which gave us accuracy of 99.34% compared to 95.45% of the pre-existing model MobileNetV3Large, and outperformed all the existing conventional models as well.

References

[1] https://github.com/cardstdani/WasteClassificationNeuralNetwork/tree/main/WasteImagesDataset

[2] M. -H. Huynh, P. -T. Pham-Hoai, A. -K. Tran and T. -D. Nguyen, "Automated Waste Sorting Using Convolutional Neural Network," 2020 7th NAFOSTED Conference on Information and Computer Science (NICS), 2020, pp. 102-107, doi: 10.1109/NICS51282.2020.9335897.

[3] Flores, Myra G. and Jose Tan Jr. "Literature Review of Automated Waste Segregation System Using Machine Learning: A Comprehensive Analysis." *International journal of simulation: systems, science & technology* (2019): n. pag.

[4] Bobulski, J., Kubanek, M. (2019). Waste Classification System Using Image Processing and Convolutional Neural Networks. In: Rojas, I., Joya, G., Catala, A. (eds) Advances in Computational Intelligence. IWANN 2019. Lecture Notes in Computer Science(), vol 11507. Springer, Cham. https://doi.org/10.1007/978-3-030-20518-8_30

[5] Olugboja Adedeji, Zenghui Wang, Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network, Procedia Manufacturing, Volume 35, 2019, Pages 607-612, ISSN 2351-9789, https://doi.org/10.1016/j.promfg.2019.05.086.

[6] Thung, Gary and Mingxiang Yang. "Classification of Trash for Recyclability Status." (2016).

[7] M. Wang, Z. Wang and J. Li, "Deep convolutional neural network applies to face recognition in small and medium databases," 2017 4th International

Conference on Systems and Informatics (ICSAI), 2017, pp. 1368-1372, doi: 10.1109/ICSAI.2017.8248499.

[8] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.

[9] C. Liu, L. Sharan, E. H. Adelson and R. Rosenholtz, "Exploring features in a Bayesian framework for material recognition," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 2010, pp. 239-246, doi: 10.1109/CVPR.2010.5540207.

[10] Gaurav Mittal, Kaushal B. Yagnik, Mohit Garg, and Narayanan C. Krishnan. 2016. SpotGarbage: smartphone app to detect garbage using deep learning. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16). Association for Computing Machinery, New York, NY, USA, 940–945. https://doi.org/10.1145/2971648.2971731.

[11] J. Donovan, “Auto-trash sorts garbage automatically at the TechCrunch disrupt hackathon.” Online: https://techcrunch.com/2016/09/13/auto-trash-sorts-garbage-automatically-at-the-techcrunch-disrupt-hackathon/

[12] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. 2015 IEEE International Conference on Computer Vision (ICCV). doi:10.1109/iccv.2015.123

[13] Malik, J., Belongie, S., Leung, T. et al. Contour and Texture Analysis for Image Segmentation. International Journal of Computer Vision 43, 7–27 (2001). https://doi.org/10.1023/A:1011174803800

[14] Xin, Mingyuan & Wang, Yong. (2019). Research on image classification model based on deep convolution neural network. EURASIP Journal on Image and Video Processing. 2019. 10.1186/s13640-019-0417-8.

[15] Sakr, G. E., Mokbel, M., Darwich, A., Khneisser, M. N., & Hadi, A. (2016). Comparing deep learning and support vector machines for autonomous waste sorting. 2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET). doi:10.1109/imcet.2016.7777453

[16] Torres Garcia, Andres & Rodea-Aragón, Oscar & Longoria-Gandara, Omar & Sánchez-García, Francisco & González-Jiménez, Luis. (2015). Intelligent Waste Separator. Computacion y Sistemas. 19. 487-500. 10.13053/CyS-19-3-2254.

[17] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[18] R. A. Aral, Ş. R. Keskin, M. Kaya and M. Hacıömeroğlu, "Classification of TrashNet Dataset Based on Deep Learning Models," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 2058-2062, doi: 10.1109/BigData.2018.8622212.

[19] Chu, Yinghao & Huang, Chen & Xie, Xiaodan & Tan, Bohai & Kamal, Shyam & Xiong, Xiaogang. (2018). Multilayer Hybrid Deep-Learning Method for Waste Classification and Recycling. Computational Intelligence and Neuroscience. 2018. 1-9. 10.1155/2018/5060857.

[20] Taigman, Yaniv & Yang, Ming & Ranzato, Marc'Aurelio & Wolf, Lior. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face

Verification. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 10.1109/CVPR.2014.220.

[21] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[22] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818-2826.

[23] Chollet, F. (2016). Xception: Deep Learning with Depthwise Separable Convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1800-1807.

[24] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR, abs/1409.1556*.

[25] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv, abs/1704.04861*.

[26] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.

[27] Tan, M., & Le, Q.V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ArXiv, abs/1905.11946*.

[28] Tan, M., & Le, Q. (2021). EfficientNetV2: Smaller Models and Faster Training. *ArXiv, abs/2104.00298*.

[29] Huang, G., Liu, Z., & Weinberger, K.Q. (2016). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261-2269.

[30] Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. (2017). Learning Transferable Architectures for Scalable Image Recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8697-8710.

Appendices

CODE SNIPPETS OF THE MODEL

```
!git clone https://github.com/cardstdani/WasteClassificationNeuralNetwork.git
```

```
Cloning into 'WasteClassificationNeuralNetwork'...
remote: Enumerating objects: 5077, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 5077 (delta 4), reused 0 (delta 0), pack-reused 5068
Receiving objects: 100% (5077/5077), 196.41 MiB | 29.11 MiB/s, done.
Resolving deltas: 100% (5/5), done.
```

```
#LOAD DATA
DIR = "/content/WasteClassificationNeuralNetwork/WasteImagesDataset"
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(DIR, validation_split=0.1, subset="training", seed=42,
test_dataset = tf.keras.preprocessing.image_dataset_from_directory(DIR, validation_split=0.1, subset="validation", seed=42,

classes = train_dataset.class_names
numClasses = len(train_dataset.class_names)
print(classes)

AUTOTUNE = tf.data.AUTOTUNE

train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

```
baseModel = tf.keras.applications.MobileNetV3Large(input_shape=(256, 256,3), weights='imagenet', include_top=False, classes=numClasses)
for layers in baseModel.layers[:-6]:
  layers.trainable=False

last_output = baseModel.layers[-1].output
x = tf.keras.layers.Dropout(0.45) (last_output)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.BatchNormalization() (x)
x = tf.keras.layers.Dense(256, activation = tf.keras.activations.elu, kernel_regularizer=tf.keras.regularizers.l1(0.045), activity_regularizer
x = tf.keras.layers.Dropout(0.45) (x)
x = tf.keras.layers.Dense(numClasses, activation='softmax')(x)

model = tf.keras.Model(inputs=baseModel.input,outputs=x)
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.00125), loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),

epochs = 50
lrCallback = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-3 * 10 ** (epoch / 30))
stepDecay = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 0.1 * 0.1**math.floor(epoch / 6))
history = model.fit(train_dataset, validation_data=test_dataset, epochs=epochs, callbacks=[])
```

```python
import requests

img_data = requests.get("https://images.unsplash.com/photo-1591872203534-278fc084969e?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxw
with open('img.jpg', 'wb') as handler:
    handler.write(img_data)

path = "/content/img.jpg"

img = tf.keras.preprocessing.image.load_img(path, target_size=(256, 256))
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = model.predict(img_array)

plt.imshow(img)
print(predictions[0]*100, "\n", classes)
print("Prediction: ", classes[np.argmax(predictions)], f"{predictions[0][np.argmax(predictions)]*100}%")
```

```python
img_data = requests.get("https://images.unsplash.com/photo-1591872203534-278fc084969e?ixlib=rb-1.2.1&ixid=MnwxMjA
with open('img.jpg', 'wb') as handler:
    handler.write(img_data)

path = "img.jpg"

img = tf.keras.preprocessing.image.load_img(path, target_size=(256, 256))
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

prediction1 = res.predict(img_array)
prediction2 = effs.predict(img_array)
prediction3 = effl.predict(img_array)
plt.imshow(img)



predclass = [classes[np.argmax(prediction1)],classes[np.argmax(prediction2)],classes[np.argmax(prediction3)]]
print(predclass)

print("\nPrediction: ", mode(predclass), f"{(predictions[0][np.argmax(prediction1)]*100+predictions[0][np.argmax(



if(classes[np.argmax(predictions)]== "Organic Waste"):
  print("\nWet Waste")
else:
  print("\nDry Waste")
```

```
Found 5078 files belonging to 9 classes.
Using 4571 files for training.
Found 5078 files belonging to 9 classes.
Using 507 files for validation.
['Aluminium', 'Carton', 'Glass', 'Organic Waste', 'Other Plastics', 'Paper and Cardboard', 'Plastic', 'Textiles', 'Wood']
```

```python
def plot_confusion_matrix(cm, target_names, cmap=None):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools

    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy

    if cmap is None:
        cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title('Confusion matrix')
    plt.colorbar()

    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="black")

    plt.tight_layout()
```