TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The classification of organic and inorganic materials plays a crucial role in waste management and recycling processes. A classifier system was developed to effectively distinguish between organic and inorganic materials using machine learning techniques. The system utilized various features extracted from waste samples, including visual appearance, elemental composition, and spectral signatures. To train the classifier, a large dataset of labelled waste samples was collected, encompassing a wide range of organic and inorganic materials commonly found in waste streams. Machine learning algorithms and deep neural networks, were employed to learn the complex patterns and relationships within the dataset.

The system's performance was evaluated using rigorous validation methods, including cross- validation, and testing with unseen waste samples. The methodology for identifying organic- inorganic materials using machine learning involves collecting a comprehensive dataset of spectral and compositional data. Relevant features are extracted, and the data is pre-processedto handle missing values, normalize features, and remove noise. The trained model is then deployed, allowing users to input data for material identification. This methodology integrates data collection, feature extraction, preprocessing, model selection, training, validation, and deployment to develop an efficient approach for identifying organic-inorganic materials using machine learning.

A robust classifier system capable of accurately identifying organic and inorganic materials within waste streams. The integration of machine learning techniques into waste management processes holds promise for optimizing recycling efforts and promoting a sustainable circular economy. Further research can explore the application of this system in real-world waste management scenarios and the development of advanced waste sorting technologies. Model aims to develop a machine learning-based approach for the identification of organic-inorganic materials. The successful implementation of project could lead to advancements in material science and accelerate the discovery and development of novel organic-inorganic materials for various applications.

1.INTRODUCTION

Identification of Organic and Inorganic materials in the present time is known to everyone but unfortunately, it is neglected by numerous people that are utilized to portray exercises for wastesegregation to take care of issues brought about by wrong garbage disposal. Illegal dumpinghas been a constant issue in numerous urban communities on the planet. The smells and pollutants brought about by deserted household things and unloaded trash, and construction leftovers ruin the city as well as threatens the wellbeing of the citizens. To diminish illegal dumping, a couple of urban areas have planned network-based voluntary reporting frameworks and observation camera-based monitoring frameworks. But these methodologies require manual observing and recognition, which are vulnerable and expensive against false alarms.

Garbage is a worldwide issue that affects everybody and every single living being. An examination shows that 74% of the plastics spilling into the ocean from the Philippines originate from the garbage. In our, everyday lives may neglect to separate accurately the garbage of our homes, and industrially the organizations in charge of this part need to spend a lot of money on labor and work. The procedure in which the garbage is segregated is splitting of garbage into divergent components. This is regularly done by handpicking physically which some the time causes hazardous and dreadful to human health if not appropriately done. To solve this issue, waste classification, and identification is acquainted which helps everyone particularly the government authorities and officials to effectively segregate wastes specifically the recyclable ones. To automatize the process of recycling, it is essential to propose smart frameworks that can see waste classification effectively. By making the use of object detection software in waste segregation is a worthwhile methodology when contrasted with the traditional recycling strategies, because of the huge numbers of objects that arerecognized in a limited timeframe.

The conventional approach depends on the goodwill of the human work which inclines to fail on waste sorting for recycling. The techniques of deep learning are being effectively appliedto different areas, for example, medical imaging, autonomous driving,

and numerous industrial environments with amazing outcomes on object identification issues. Applying these techniques to waste sorting can build the amount of recycled material and thus, give a simpler everyday life for the common people as well as more efficacy for the industry. Deep Learning is the class of Machine Learning Algorithms which is a subset of Artificial Intelligence that uses multiple layers of data representation and feature extraction. The different applications where deep learning plays an important role are in speech and visual recognition, speech to text conversion, detection, and recognition of the face, image recognition, drug detection, weather prediction, etc. Deep learning permits the preparation of numerous layers through the computational models to learn the representations of data with the abstraction of numerous layers.

Convolutional Neural Network (CNN) is the most suitable image classification technique in the most recent years, where from the segmented objects no handcrafted features are extracted. A previously trained CNN will always perform better on fresh images for classification as ithas already adapted or learned the visual features and can transfer that information through transfer learning. Thus, this paper is aimed at planning and developing up a framework with a deep learning approach that can be effectively used for waste segregation. The image will be recognized by utilizing the concept of a convolutional neural network and with the help of an image processing method that identifies wastes from their shape, colour, dimension, and size. This technique automatically will help the system to learn the pertinent features from the sample images of the trash and consequently recognize those features in new images. By using the strategy of convolutional neural networks, garbage will be classified into different classes. The strategy utilized for this characterization is with the assistance of TensorFlow's Object Detection API and Faster CNN technique. Through this technique, bounding boxes are madeon the recyclable waste demonstrating which class (cardboard, paper, metal, glass, plastic, and trash) the waste falls into. The main objective of this study is to develop software to detect types of recyclable materials in trash bins and check for possible contamination (non- recyclable materials), which would ultimately reduce human effort in waste segregation and expedite the entire process.

1.1 Existing System

The existed model of waste classifier employed machine learning algorithms, such as decision trees, support vector machines (SVM), or random forests, to train models for waste classification. These algorithms were trained using labelled data, where waste items were manually categorized into different classes.

In the existing model, feature extraction techniques were used to capture relevant information from waste items. These features could include visual attributes like color, shape, texture, or statistical properties of waste materials. These extracted features were then used as inputs for the machine learning algorithms.

Disadvantages of Existing System

1. Limited Dataset Size: The availability of labelled waste data for training the models was limited in the old model. This constrained the accuracy and generalizability of the waste classifier. The models often faced challenges when presented with unseen waste items or variations in waste characteristics.

2. Lack of Deep Learning: The use of deep learning methods, such as convolutional neural networks (CNNs), was limited in the old model of waste classifier. Deep learning techniques have since demonstrated superior performance in various computer vision tasks, including waste classification, due to their ability to automatically learn hierarchical representations from data.

3. Real-time Processing: The old model of waste classifier had limitations in real-time processing capabilities. The classification process often required manual intervention or batch processing leading to delays in decision-making and waste sorting processes.

1.2 Problem Statement

Inefficient waste sorting and lack of accurate waste categorization pose significant challengesin waste management systems. Manual sorting processes are time-consuming, error-prone, and often lead to improper waste disposal, hindering recycling efforts and causing environmental harm. There is a need for an automated waste classifier that can accurately categorize diverse

waste materials, including recyclable, non-recyclable, organic, and hazardous waste, to streamline waste management operations, optimize recycling processes, and promote environmental sustainability. The waste classifier must be scalable, adaptable to various waste streams, and capable of providing real-time, accurate classification results to improve waste sorting efficiency, reduce contamination, and facilitate the effective management and recovery of valuable resources.

1.3 Proposed System

The new model of waste classifier using AI refers to more recent and advanced approaches that leverage artificial intelligence techniques for waste classification. These models incorporate state- of-the-art machine learning algorithms, deep learning methods, and computer vision techniques to achieve accurate and efficient waste classification. Here is a brief description of the new model of waste classifier using AI:

Advantages of Proposed System

1. **Deep Leaming and Convolutional Neural Networks (CNNs):** The new model heavily relies on deep learning methods, particularly CNNs, for waste classification. CNNs can automatically learn hierarchical features from waste item images, enabling more accurate and robust classification performance.

2. **Transfer Learning:** Transfer learning is widely used in the new model to leverage pre- trained CNN models that have been trained on large-scale image datasets, such as ImageNet. By fine-tuning these pre-trained models on waste classification tasks with a smaller labelled dataset, the new model can achieve better performance with less training data.

3. **Large Labelled Datasets:** The new model benefits from larger and more diverse labelled datasets for training. These datasets contain a wide variety of waste items, covering different categories and variations in waste characteristics this helps improve the model's ability to generalize and accurately classify a broader range of waste materials.

4. **Integration of Sensor Data:** The updated model integrates data from diverse sensors

like infrared and spectral, enhancing waste classification with comprehensive insights from various sources.

5. **Real-time Processing and Automation:** The new model focuses on real-time processing capabilities and automation to enable swift and efficient waste classification. This is particularly important in waste management systems that require rapid decision-making and sorting processes.

6. **Enhanced Accuracy and Adaptability:** With the advancements in Al techniques, the new model exhibits improved accuracy and adaptability compared to previous approaches. It can accurately classify diverse waste materials, including recyclable, non-recyclable, organic, and hazardous waste, with fewer errors and higher precision.

7. **Continuous Learning and Updates:** The new model embraces the concept of continuous caring and updates. It can be continuously trained and updated with new data to adapt to evolving waste categories, variations in waste materials, and changes in classification requirements.

1.4 Requirements Specification

1.4.1 Software Requirements

1. Operating system : Windows 8 and above
2. Languages : Python
3. IDE : Jupyter Notebook
4. Library : Python libraries like PIL, Tensorflow, NumPy.

1.4.2 Hardware Requirements

1. Processor : Intel Dual Core i5 or above
2. Hard disk : 8GB and above
3. RAM : 8GB and above

2. LITERATURE SURVEY

[1]. Lin, Z., Li, X., Zhang. Y., Zhao, Y., Wang. H., & Song, J. (2022). Machine learning-assisted discovery and design of organic materials. Chemical Society Reviews, 51(9), 3371-3401.

The dataset spans six classes: glass, paper, cardboard, plastic, metal, and trash. The dataset consists of 2527 images (501 glass, 594 paper, 403 cardboard, 482 plastic, 410 metal, 137 trash), and it is annotated by category per image. The dataset consists of photographs of garbage taken on a white background; the different exposure and lighting were selected foreach photo (mainly one object per photo).

Authors explore the SVM and CNN algorithms with the purpose of efficiently classifying garbage into six different recycling categories. They used an architecture like AlexNet but smaller in filter quantity and size.

The SVM achieved better results than the Neural Network. It achieved an accuracy of 63% using a 70/30 train/test data split. Neural network with a 70/30 train/test split achieved a testing accuracy of 27%.

[2]. Dehmamy, N., Jain, A., Marom, N., Tkatchenko, A., & von Lilienfeld, O. A. (2022). Transferable machine-learning model for bandgap predictions of hybrid organic-inorganic perovskites. ACS Energy Letters, 7(1), 207-215.

Authors tested well-known Deep Learning models to provide the most efficient approach. In Densenet121. DenseNet169, InceptionResnet V2. MobileNet, Xception architectures were used for Trashnet dataset and Adam and Adadelta were used as the optimizer in neural networkmodels.

The most successful test accuracy rates were achieved with the fine-tuned Densenet-121 and Densenet-169 models. In the selection of the optimizer, Adam and Adadelta optimizers were tried with 100 epochs in InceptionResNet V2 model. As a result of this experiment, a highertest accuracy was obtained in the Adam optimizer.

[3]. Neumann, M., Wiktelius, D., & Söderhjelm, P. (2022). Machine learning based modeling of organic-inorganic interfaces for perovskite solar cells. Computational Materials Science, 204, 110854.

5000 images with resolution 640 x 480 px were collected on plain, grey background. Each waste image is grouped with its counterpart numerical feature information as a data instance (50 categories). which is then manually labelled as either recyclable (4 main categories for recyclable part) or not.

Authors tested a hybrid CNN approach for waste classification (and AlexNet CNN to compare). Both Multilayer hybrid system and CNN perform well when investigated items have strong image features. However, CNN performs poorly when waste items lack distinctive image features, especially for "other" waste. MHS achieves a significantly higher classification performance; the overall performance accuracies are 98.2% and 91.6%, (the accuracy of the reference model is 87.7% and 80.0%) under two different testing scenarios: the item is placed with fixed and random orientations.

[4]. Yang, T., Li, 7., Gong. Y., Zhou, Y., Zhang, S., & Gao, J. (2022). Machine learning- assisted design of inorganic materials: A review. Frontiers in Materials, 9, 105.

Garbage in Images (GINI) dataset with 2561 images with unspecified resolution, 1496 images were annotated by bounding boxes (one class - trash). Bing Image Search API was used to create their dataset.

The authors utilize a pre-trained AlexNet, and their approach focuses on segmenting a pile of garbage in an image and provides no details about types of wastes in that segment. Their method is based on extracting image patches and combining their predictions, and therefore cannot capture the finer object boundary details.

GarbNet reached an accuracy of 87.69% for the task of the detection of garbage, but produced wrong predictions when in an image are detected objects like waste or when they are in the distance.

[5]. Krishnamoorthy, V., & Singh, A. (2021). Machine learning assisted identification of multi-component organic and inorganic crystals. Journal of Chemical Information and Modelling, 61(9), 4615-4626.

The data (Trash-ICRA19 dataset) was sourced from the J-EDI dataset of marine debris. The videos that comprise that dataset vary greatly in quality, depth, objects in scenes, and the cameras used.They contain images of many different types of marine debris, captured from real-world environments. providing a variety of objects in different states of decay, occlusion, and overgrowth. Additionally, the clarity of the water and quality of the light vary significantlyfrom video to video. These videos were processed to extract 5,700 images, which comprise this dataset, all labelled with bounding boxes of trash, biological objects such as plants.

The four network architectures selected for this project were chosen from the most popular and successful abject detection networks: YOlOv2, Tiny-YOLO, Faster RCNN with Inception v2, SSD with MobileNet v2. In the case of VOLO, the network was not only fine-tuned, but also trained using a transfer learning - authors froze all but not the last several layers so that only the last few layers had their weights updated during the training process.

YOLOv2 - mAP-479, Tiny-YOLO - mAP-31.6, Faster RCNN with Inception V2 - mAP-81, SSD with MobileNet v2 - mAP-674. Faster R-CNN is the obvious choice purely from a standpoint of accuracy, but falters in terms of inference time. YOLOV2 strikes a good balance of accuracy and speed, while SSD provides the best inference times on CPU.

[6]. Gómez-Bombarelli, R., & Aspuru-Guzik, A. (2021). Deep generative models in chemistry. Nature Reviews Chemistry, 5(9), 598-613.

1500 images were collected from mainly outdoor environments such as woods, roads and beaches, and annotated by segmentation masks. There are 60 categories which belong to 28 super (top) categories.

Additionally, images have background tag: water, sand, trash, vegetation, indoor, pavement. Authors adopted the Mask R-CNN implementation with Resnet-50 in a Feature PyramidNetwork as a backbone with an input layer size of 1024x1024 px Weights were started usingMask R-CNN weights trained on COCO dataset.

Table 2.1: Comparison Literature Survey on Classification of Organic and Inorganic Waste

S.NO	TITLE	AUTHORS & YEARS	ALGORITHM / METHODOLOGY	MERIT OR ADVANTAGE	DEMERIT
1	[1].Machine learning-assisted discovery and design of organic materials	Lin, Z., Li, X., Zhang, Y., Zhao, Y., Wang, H.,& Song, J. (2022)	SVM and CNN	SVM achieved better results than Neural Network (63% accuracy).	Neural network had lower testing accuracy (27%).
2	[2].Transferable machine-learning model for bandgap predictions of hybrid organic-inorganic perovskites	Dehmamy, N., Jain, A., Marom, N., Tkatchenko, A., & von Lilienfeld, O. A. (2022)	Densenet121, DenseNet169, InceptionResnet V2, MobileNet, Xception	Fine-tuned Densenet-121 and Densenet-169 models achieved the highest test accuracy.	Performance may vary depending on the architecture and optimizer used.
3	[3].Machine learning based modeling of organic-inorganic interfaces for perovskite solar cells	Neumann, M., Wiktelius, D., & Söderhjelm, P. (2022)	Hybrid CNN approach and AlexNet	Multilayer hybrid system (MHS) achieved significantly higher classification performance (98.2%).	CNN performs poorly when waste items lack distinctive image features, especially for "other" waste.

4	[4].Machine learning-assisted design of inorganic materials: A review	Yang, T., Li, 7., Gong. Y., Zhou, Y., Zhang, S., & Gao, J. (2022)	Pre-trained AlexNet	GarbNet reached an accuracy of 87.69% for garbage detection but produced wrong predictions in certain scenarios.	The method based on extracting image patches may not capture finer object boundary details.
5	[5].Machine learning assisted identification of multi-component organic and inorganic crystals	Krishnamoorthy, V., & Singh, A. (2021)	YOLOv2,Tiny-YOLO, Faster RCNN with Inception v2, SSD with MobileNet v2	Faster RCNN with Inception V2 achieved high accuracy (mAP-81) with good balance between accuracy and speed.	Faster RCNN has longer inference times on CPU compared to YOLOv2.
6	[6].Deep generative models in chemistry	Gómez-Bombarelli, R., & Aspuru-Guzik, A. (2021)	Mask R-CNN with Resnet-50 in a Feature Pyramid Network	Achieved best results for litter score in one-class semantic segmentation task.	The metric values for map were relatively low for the segmentation task.

3. CLASSIFICATION OF ORGANIC AND INORGANIC WASTE USING NEURAL NETWORKS

3.1. System Architecture

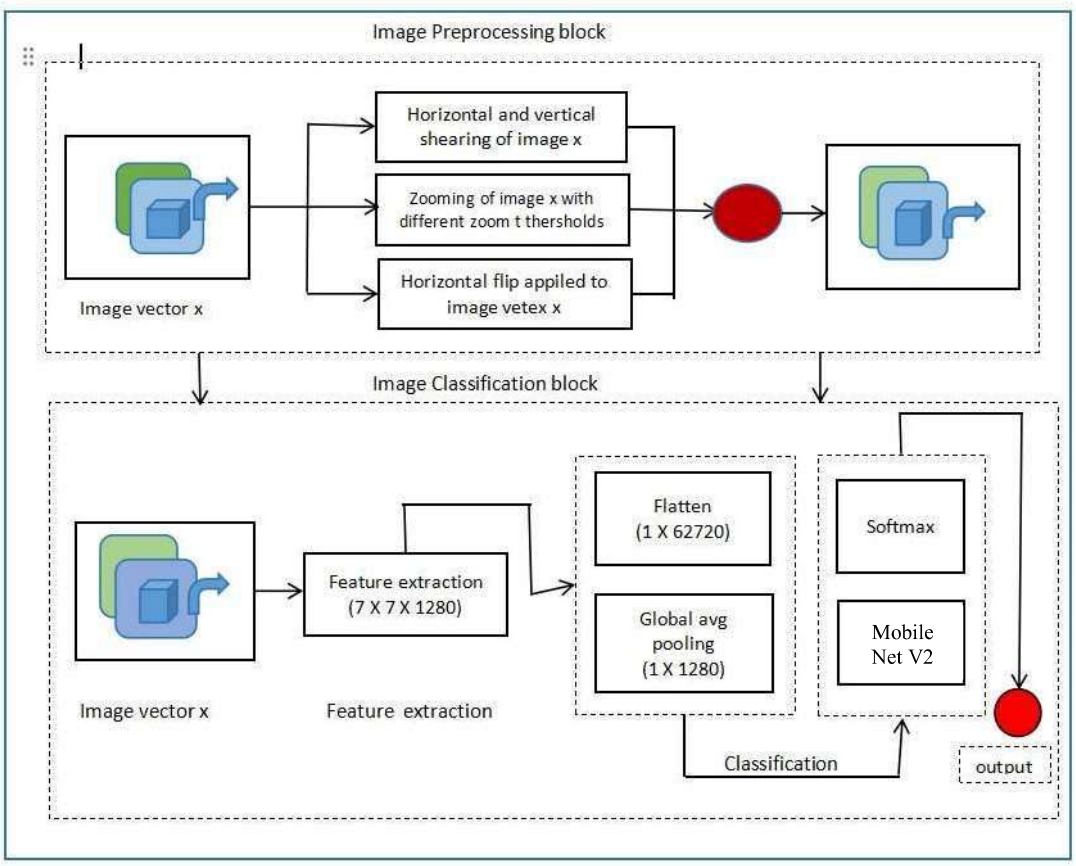

Figure 3.1: Block Diagram of the System Architecture

The above figure 3.1 represents the actual flow of the proposed system. To develop such a system a trained dataset is required to classify an image. Trained dataset consists of two parts trained result and test result. The dataset must be retrained to achieve higher accuracy in identification using retrain.py in PyCharm. The training dataset is made using 50000 steps taking into consideration that the higher the number of steps higher is its accuracy.

3.2 Convolutional Neural Network

CNN consists of four layers: convolutional layer, activation layer, pooling layer and fully connected. The convolutional layer allows extracting visual features from an image in small amounts. Pooling is used to reduce the number of neurons from the previous convolutional layer but maintaining the important information. Activation layer passes a value through a function which compresses values into range. A fully connected layer connects a neuron from one layer to every neuron in another layer. As CNN classifies each neuron in depth, it provides more accuracy.

CNN has two components :

1) Feature extraction part: Features are detected when network performs a series of convolutional and pooling operation.

2) Classification part: Extracted features are given to a fully connected layer which acts as classifier.

3.2.1 Feature Extraction

Features are detected when the network performs a series of convolutional and pooling operations.

3.2.2 Convolutional Layer

Convolutional neural network layer types mainly include three types as shown in figure 3.2 namely Convolutional layer, pooling layer, and fully connected layer Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each convolutional neuron processes data only for its receptive field. Although fully connected feed forward neuralnetworks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For

instance, a fully connected layer for a (small) image ofsize 100 x 100 has 10000 weights for each neuron in the second layer. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using back propagation. The aim of Convolutional layer is to learn feature representations of the input.

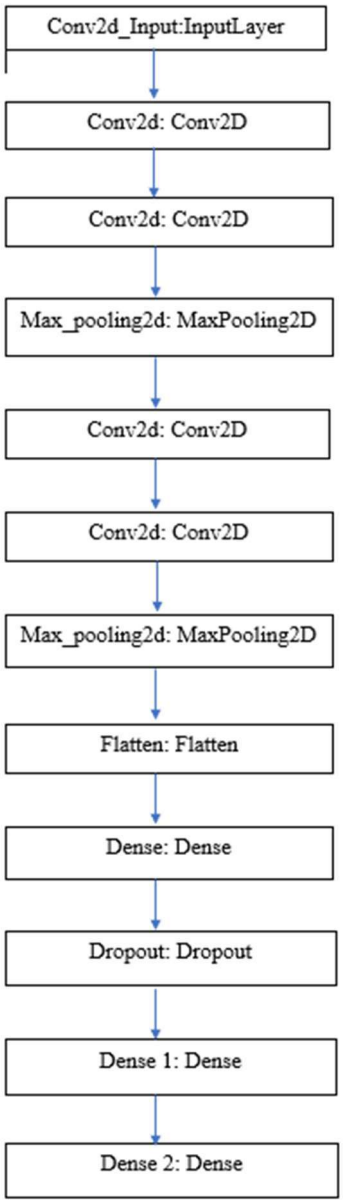

Figure 3.2 : CNN Architecture

3.2.3 Pooling Layers

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer Another exampleis average pooling, which uses the average value from each of a cluster of neurons at the prior layer. The sampling process is equivalent to fuzzy filtering. The pooling layer has the effect of the secondary feature extraction, it can reduce the dimensions of the feature maps and increase the robustness of feature extraction. It is usually placed between two Convolutional layers. The size of feature maps in pooling layer is determined according to the moving step of kernels. The typical pooling operations are average pooling and max pooling. Extract the high-level characteristics of inputs by stacking several Convolutional layers and pooling layers.

3.2.4 Fully Connected Layer

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images. In general, the classifier of Convolutional neural network is one or more fully connected layers. They take all neurons in the previous layer and connect them to every single neuron of the current layer. There is no spatial information preserved in fully connected layers. The last fully connected layer is followed by an output layer. For classification tasks, SoftMax regression is commonly used because of it generating a well-performed probability distribution of the outputs. Another commonly used method is SVM, which can be combined with CNNs to solve different classification tasks.

3.2.5 Dense Layer

Dense Layers serve as the final layers responsible for making predictions based on the features extracted by earlier convolutional and pooling layers. These layers are fully connected, meaning each neuron in a Dense Layer is connected to every neuron in the preceding layer. The Dense Layer aggregates and processes the spatial features learned

by the convolutional layers, transforming them into predictions or classifications suitable for the given task, such as image recognition or object detection.

3.2.6 Dropout Layer

Dropout is typically inserted after the fully connected Dense layers. It helps to prevent the model from relying too heavily on specific features or neurons, promoting better generalization and reducing the likelihood of overfitting. Dropout essentially acts as a form of ensemble learning, where different sub-networks are trained on different subsets of the data, leading to a more robust overall model. During inference (i.e., when making predictions on new data), Dropout is typically turned off, and the full network is used to make predictions.

3.2.7 Receptive Field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically, the subarea is of a square shape (e.g., size 5 by 5). The input area ofa neuron is called its receptive field.

3.2.8 Weights

Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progress by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a filter and represent some feature of the input. A distinguishing feature of CNNs is that many neurons share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used acrossall receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights.

3.3 Dataset

A dataset is a collection of data. For performing action related, the dataset provides a collection of 24,705 images of solid household waste, categorised into two classes: organic (13,880) and recyclable (10,825). The data is a restructured and represented version of the original dataset by Sashaank Sekar available at https://www.kaggle.com/techsash/waste-classification-data. The original dataset from Kaggle consists of 25,077 images of organic (13,966) and recyclable (11,111) images. However, performed some clean-up operations explained in the "furthernote" to reduce the data to 24,705 with (13,880 organic) and (10,825 recyclable) The restructured data has been used in a research study undertaken by academics and researchersat Computer Science Department, Edge Hill University, United Kingdom.

3.4 Libraries

Tensor Flow:

It is an end-to-end opensource platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of- the-art in ML and developers easily build and deploy ML powered applications. Build and train ML models easily using intuitive high- level APIs like Karas with eager execution, which makes for immediate model iteration and easy debugging. Easily train and deploy models in the cloud, on prem, in the browser, or on device no matter what language you use. A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy. If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition. TensorFlow has always provided a direct path to production. Whether it's on servers, edge devices, or the web, TensorFlow lets you train and deploy your model easily, no matter what language or platform you use.

Keras:

It is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that: Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility). Supports both convolutional networks and recurrent networks, as well as combinations of the two. Runs seamlessly on CPU and GPU.

a) User friendliness:

Keras is an API designed for human beings, not machines. It puts user experience front and centre. Keras follows best practices for reducing cognitive load: it offers consistent simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

b) Modularity:

A model is understood as a sequence or a graph of standalone, fully configurable modules that can be plugged together with as few restrictions as possible. Neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that you can combine to create new model.

c) Easy extensibility:

New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Karas suitable for advanced research.

d) Work with Python:

No separate model's configuration files in a declarative format. The model is described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

3) NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things: A powerful N-dimensional array object, Sophisticated (broadcasting) functions. Tools for integrating C/C++ and Fortran code, Useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

4)MatplotLib

Matplotlib is a versatile Python library for creating static, interactive, and animated visualizations, offering diverse plot types like line plots, scatter plots, and heatmaps. Users can customize plots extensively, adjusting axis labels, colors, and styles. Matplotlib supports multiple output formats, allowing for image file saving or direct display in interactive environments. It's commonly used with NumPy and pandas for data analysis in scientific computing, machine learning, and data analysis projects.

3.5 Use Case Diagram

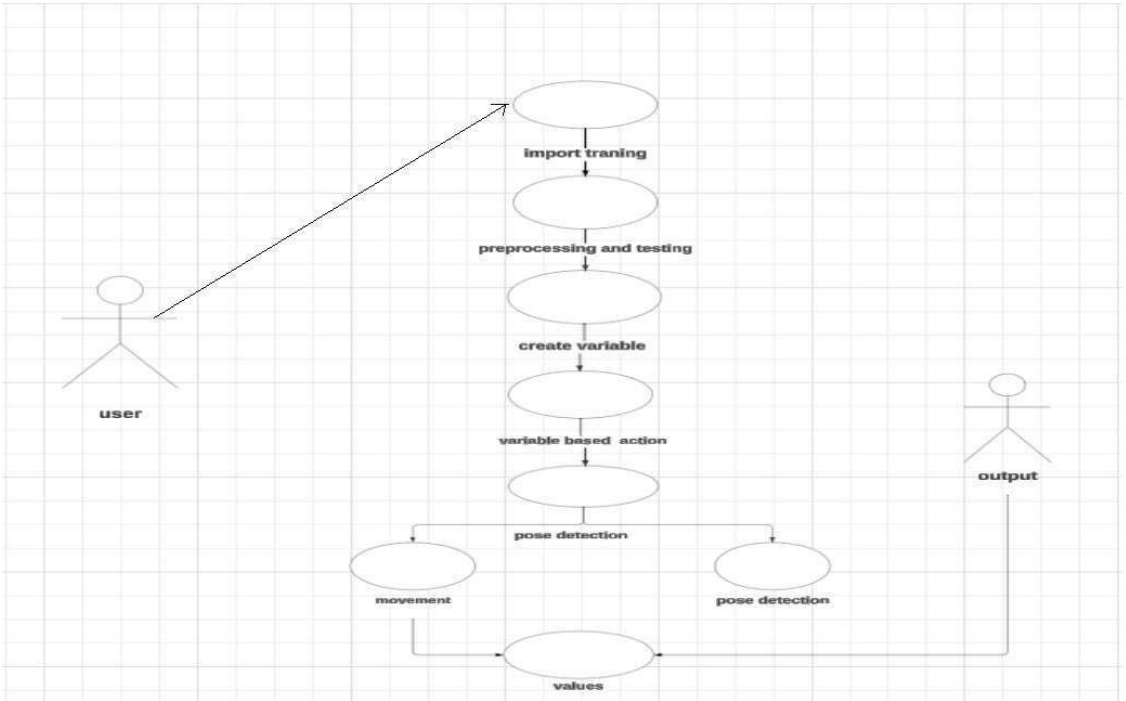

Figure 3.3: Use Case Diagram of Identification of Organic and Inorganic Materials

The previous figure 3.4 is a Use Case Diagram which shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures. Use Case diagram is helpful in exposing the requirements and planning the project during the initial stage.

3.6 Sequence Diagram

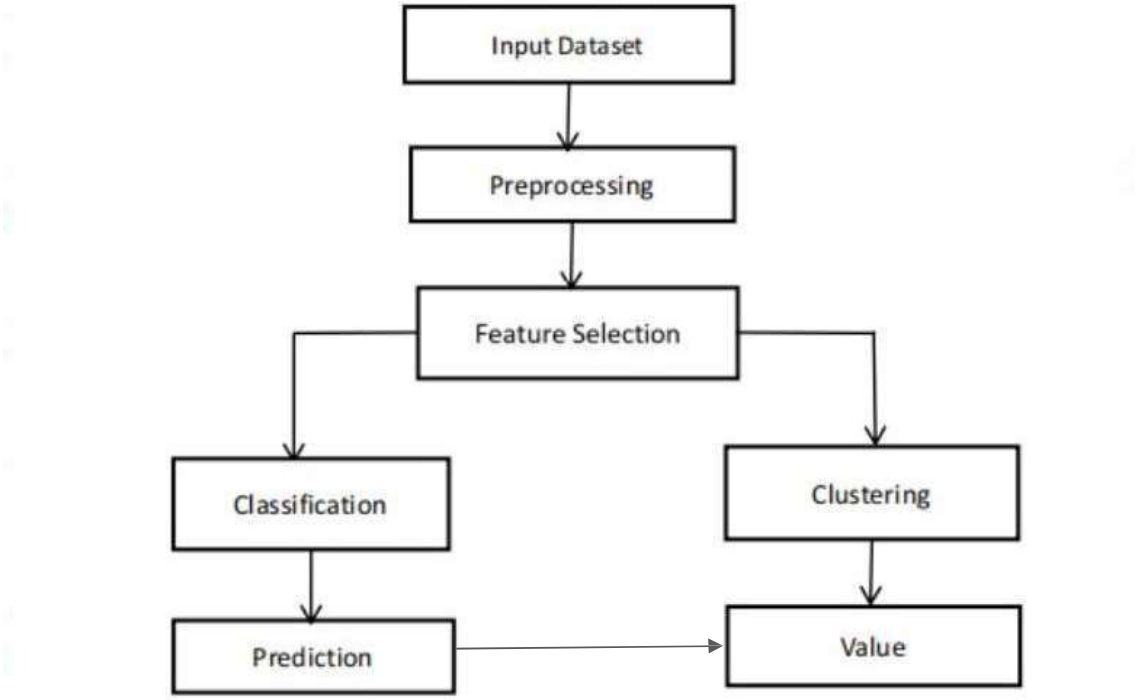

Figure 3.4 : Communication Diagram of Identification of Organic and Inorganic Materials

The above figure 3.4 is a sequence diagram which shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

4. IMPLEMENTATION

The CNN model to Identify Organic and Inorganic materials in construct as follows with the help of sequential model as shown in Figure 4.1

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 64, 64, 32)	896
max_pooling2d_7 (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d_8 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_8 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_9 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_9 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_10 (Conv2D)	(None, 8, 8, 128)	73856
max_pooling2d_10 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten_7 (Flatten)	(None, 2048)	0
dense_15 (Dense)	(None, 64)	131136
dense_15 (Dense)	(None, 64)	131136
dense_16 (Dense)	(None, 32)	2080
dense_17 (Dense)	(None, 1)	33

Total params: 235745 (920.88 KB)
Trainable params: 235745 (920.88 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 4.1 Model Structure

The model accepts four parameters

1. The image dimensions: Height and width
2. Depth
3. Number of classes in the dataset

Here are some of the Samples from the dataset

Figure 4.2 Sample Images

Data Augmenatation

Using ImageDataUsing ImageDataGenerator to create a transformation that rescales the images and applied data augmentation as well. Generate augumentation only for train data while validation data are only rescaling ones as shown in Figure 4.3

Keras ImageDataGenerator class provides a quick and easy way to augment of our images. It provides many of different augmentation techniques like standardization, shifts, rotation, flips, brightness change etc. Now set our dataset into train, test and validation data.

Lise the flow_from_directory() method which allows to read the images directly from the directory and augment them while the neural network model is learning on the training data.

Visualization of Augmentation

Below can see the results of the image augmentation as shown in figure 4.4

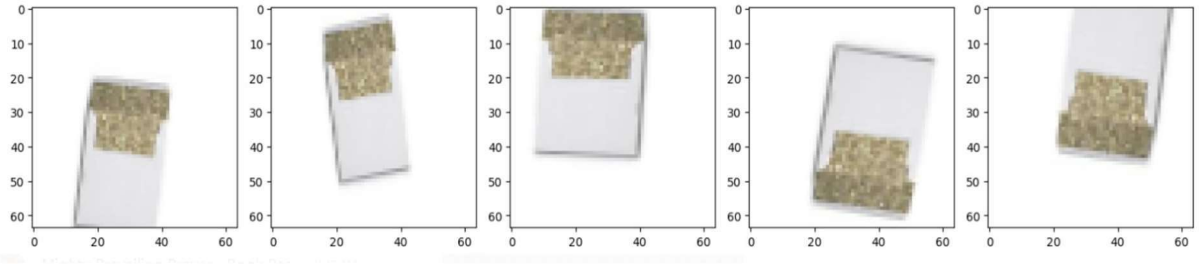

Fig 4.3 Visualization of augumentation

Creating the CNN Model

Input Layer:

The input layer serves as the entry point for the neural network, specifying the shape of the input data. In our case, images are expected to have dimensions of 64x64 pixels with 3 color channels (red, green, blue).

Convolutional Layers:

Convolutional layers are the backbone of CNNs, responsible for extracting relevant features from input images through the application of convolutional filters.

In our architecture, three convolutional layers are stacked sequentially.

Each convolutional layer consists of multiple filters, which slide over the input image to perform convolution operations.

The filters learn to detect patterns and features such as edges, textures, and shapes at different spatial scales.

ReLU (Rectified Linear Unit) activation functions are applied after each convolution operation to introduce non-linearity, enabling the model to learn complex relationships within the data.

Padding is employed to ensure that the spatial dimensions of the feature maps remain consistent throughout the convolutional layers, preventing information loss at the borders of the image.

Max-Pooling Layers:

Max-pooling layers follow each convolutional layer to downsample the feature maps, reducing their spatial dimensions.

Max-pooling operates by selecting the maximum value within a predefined window (e.g., 2x2) and discarding the rest.

This downsampling process helps in reducing computational complexity and controlling overfitting by extracting the most salient features.

Flatten Layer:

The Flatten layer transforms the 2D feature maps obtained from the convolutional layers into a 1D vector.

This flattening process is necessary to feed the extracted features into the subsequent fully connected layers for classification.

Fully Connected (Dense) Layers:

Fully connected layers are traditional neural network layers where each neuron is connected to every neuron in the previous and subsequent layers.

In our architecture, two dense layers are included after the Flatten layer for classification purposes.

The first dense layer consists of 128 units, serving as a bottleneck that learns high-level representations of the extracted features.

ReLU activation functions are utilized in the dense layers to introduce non-linearity and enable the model to learn complex decision boundaries.

The final dense layer comprises a single neuron with a sigmoid activation function, which outputs a probability indicating the likelihood of the input image belonging to a specific class (binary classification).

Model Compilation:

Once the architecture is defined, the model is compiled with specific configurations for training.

The Adam optimizer is chosen for its efficiency in adapting learning rates during training and handling sparse gradients.

Binary cross-entropy loss is utilized as it is well-suited for binary classification tasks, penalizing the model based on the discrepancy between predicted and actual class labels.

Accuracy is selected as the evaluation metric to monitor the model's performance during training and validation.

Model Summary:

The model.summary() function provides a detailed overview of the model architecture, including the type and dimensions of each layer, as well as the total number of trainable parameters.

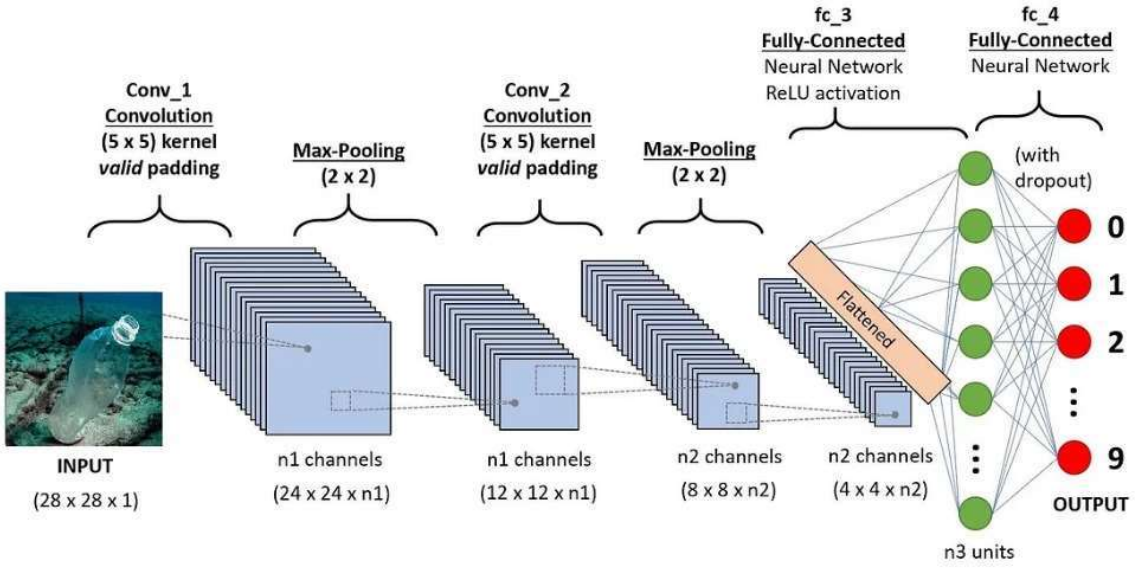

Figure 4.4 Creating Model

Train the Model

```
Epoch 1/5
706/706 [==============================] - 728s 1s/step - loss: 0.3671 - accuracy: 0.8420 - val_loss: 0.4836 - val_accuracy: 0.
8011
Epoch 2/5
706/706 [==============================] - 370s 525ms/step - loss: 0.3337 - accuracy: 0.8617 - val_loss: 0.4230 - val_accuracy:
0.8369
Epoch 3/5
706/706 [==============================] - 250s 354ms/step - loss: 0.3208 - accuracy: 0.8667 - val_loss: 0.4672 - val_accuracy:
0.8385
Epoch 4/5
706/706 [==============================] - 218s 309ms/step - loss: 0.3134 - accuracy: 0.8674 - val_loss: 0.4321 - val_accuracy:
0.8290
Epoch 5/5
706/706 [==============================] - 307s 435ms/step - loss: 0.3057 - accuracy: 0.8695 - val_loss: 0.3941 - val_accuracy:
0.8449
```

Figure 4.5 Accuracy for five epochs

```
loss, accuracy = model1.evaluate(test_set, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

```
Testing Accuracy:  0.8543
```

Figure 4.6 Accuracy of the base Model

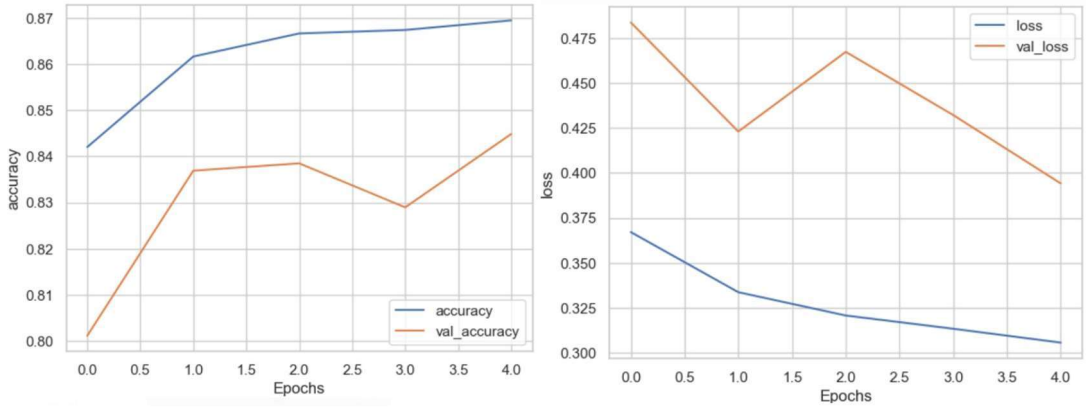

Figure 4.7 Plot of the base Model

As you can notice here, the training and validation accuracy is increasing together and start decreasing after 2 epoch. The same is in the training and validation loss. We also see a little bit of overfitting Othervise on the test data we achieved 85% accuracy

MODEL – 2

MobileNetV2:

MobileNetV2 is a convolutional neural network architecture designed for mobile and embedded vision applications. It was developed by researchers at Google, and it builds upon the success of the original MobileNet architecture, aiming to improve both the efficiency and performance of deep neural networks for mobile devices.

Here's an overview of the key components and design principles of MobileNetV2:

Depthwise Separable Convolutions: MobileNetV2 uses depthwise separable convolutions to reduce computation while preserving accuracy.

Inverted Residuals: It employs inverted residual blocks to efficiently capture and propagate information through the network.

Shortcut Connections: Inspired by ResNets, MobileNetV2 uses shortcut connections to aid gradient flow and facilitate training of deeper networks.

Width and Resolution Multipliers: These hyperparameters allow users to balance model size and accuracy based on available resources.

Mobile-Friendly Design: MobileNetV2 is optimized for mobile and embedded devices, offering a balance between efficiency and performance.

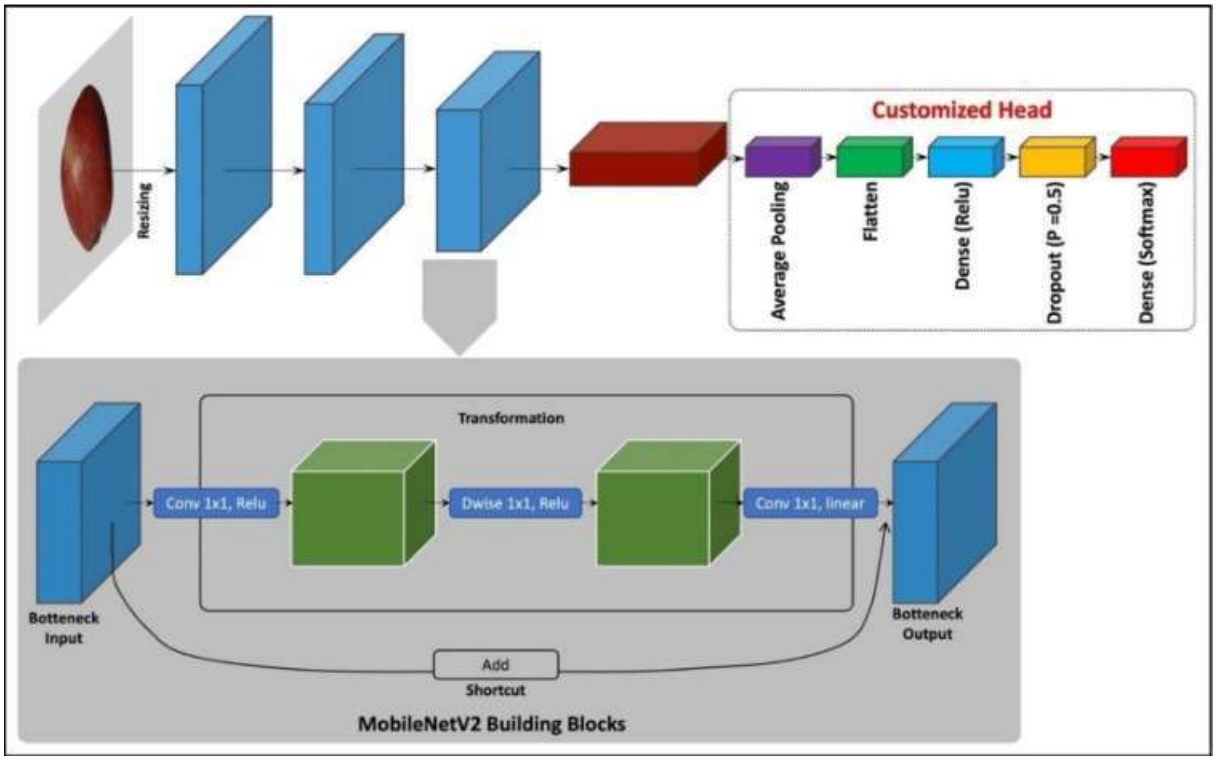

Figure 4.8 MobileNet V2 model

Training the Model

```
Epoch 3/10
22/22 [==============================] - 12s 573ms/step - loss: 0.5063 - accuracy: 0.7827 - val_loss: 0.4178 - val_accuracy: 0.
8568
Epoch 4/10
22/22 [==============================] - 13s 583ms/step - loss: 0.4492 - accuracy: 0.8139 - val_loss: 0.4459 - val_accuracy: 0.
8361
Epoch 5/10
22/22 [==============================] - 13s 590ms/step - loss: 0.4454 - accuracy: 0.7983 - val_loss: 0.4467 - val_accuracy: 0.
8218
Epoch 6/10
22/22 [==============================] - 12s 571ms/step - loss: 0.4442 - accuracy: 0.8097 - val_loss: 0.3912 - val_accuracy: 0.
8512
Epoch 7/10
22/22 [==============================] - 13s 577ms/step - loss: 0.4861 - accuracy: 0.7727 - val_loss: 0.4452 - val_accuracy: 0.
8218
Epoch 8/10
22/22 [==============================] - 12s 571ms/step - loss: 0.4399 - accuracy: 0.7912 - val_loss: 0.4460 - val_accuracy: 0.
8266
Epoch 9/10
22/22 [==============================] - 13s 579ms/step - loss: 0.4386 - accuracy: 0.8210 - val_loss: 0.4465 - val_accuracy: 0.
8194
Epoch 10/10
22/22 [==============================] - 13s 581ms/step - loss: 0.4880 - accuracy: 0.7727 - val_loss: 0.4504 - val_accuracy: 0.
8043
```

Figure 4.9 Accuracy for 10 epochs

```
loss, accuracy = model2.evaluate(test_set, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

```
Testing Accuracy:  0.8909
```

Figure 4.10 Accuracy of the Model

In model two we add additional convolution layers and we reduce the complexity of architecture in fully connected layer

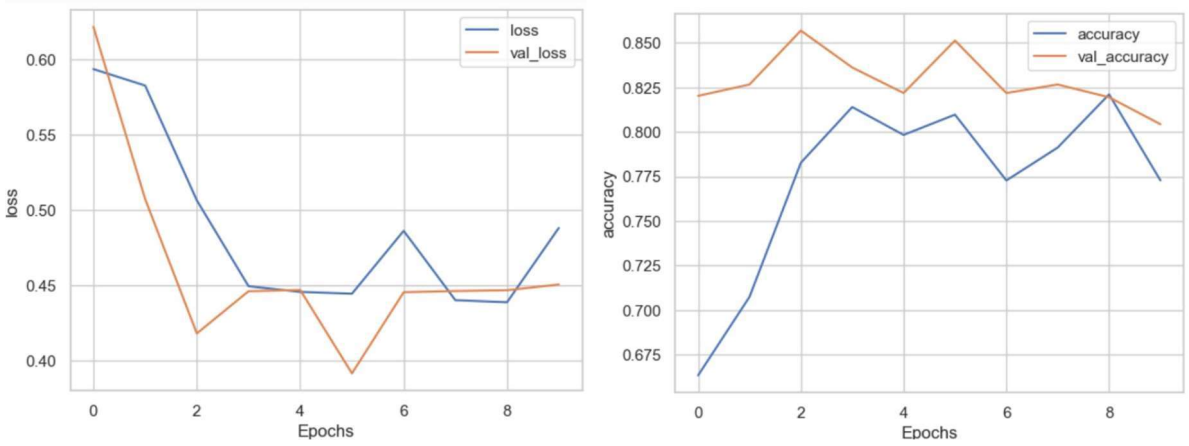

Figure 4.11 Plots of the Model

From the above plots one can see that the training and validation loss are both decreasing here with little divergence as compared to the outcome from the previous model The case with the training and validation accuracy is also alike, they are increasing together We see the overfitting is decreased and We also achieved a better accuracy on the test data of 89%.

5. RESULTS

We can see some results our model and test own image

INPUT:

```
Image.open(r"C:/Users/Tharun/Downloads/Major/DATASET/TEST/O/O_13968.jpg")
```

Figure 5.1 Testing the Image

OUTPUT:

```
if result[0][0] == 1:
    prediction = ' Inorganic Waste'
else:
    prediction = 'Organic Waste'
```

```
print(prediction)
```

```
Organic Waste
```

Figure 5.2 Output

6 . CONCLUSION AND NATURE SCOPE

6.1 CONCLUSION:

In conclusion, the application of machine learning techniques for organic and inorganic identification has demonstrated remarkable progress and potential in the field of materials science The ability to accurately classify and distinguish between organic and inorganic compounds is crucial for various applications, including drug discovery, environmental monitoring, and materials synthesis. Machine learning algorithms, such as support vector machines (SVM), random forests, and neural networks, have been successfully employed to analyse and interpret large datasets containing spectral, chemical, and structural information By training these algorithms on labelled datasets, they can learn to recognize patterns and features indicative of organic or inorganic materials, enabling rapid and reliable identification

One of the major advantages of machine learning in organic and inorganic identification is its ability to handle complex, multidimensional data. By considering numerous variables simultaneously, machine learning algorithms can capture intricate relationships and extract meaningful information that might not be apparent through traditional analytical methods. Moreover, machine learning approaches have the potential to continuously improve their performance through iterative training and optimization. As more data becomes available and algorithms are refined, the accuracy and efficiency of organic and inorganic identification are expected to increase further.

However, it is important to note that the success of machine learning models relies heavily on the quality and representativeness of the training data. The availability of diverse and well-curated datasets is crucial to ensure robust and reliable identification models. Additionally, the interpretability of machine learning models remains a challenge, as they often function as black boxes, making it difficult to understand the underlying decision-making process.

In conclusion, machine learning holds great promise for organic and inorganic identification, revolutionizing the way materials are analysed and classified. Continued research and development in this field will likely lead to more accurate and efficient identification methods, opening new possibilities for technological advancements, and industrial applications.

6.2 FUTURE SCOPE

The future scope of organic and inorganic identification using machine learning is vast and holds immense potential for advancements in materials science and related fields. Here are some key areas where further developments can be expected:

Enhanced Accuracy and Efficiency Ongoing research and advancements in machine learning algorithms, such as deep learning, reinforcement learning, and transfer learning, will likely lead to improved accuracy and efficiency in organic and inorganic identification. These algorithms can handle larger and more complex datasets, extract deeper features, and make more accurate predictions, thereby enhancing the overall performance of identification models.

Integration of Multiple Data Sources: Machine learning models can benefit from the integration of diverse data sources, such as spectroscopic data, elemental composition, crystallographic information, and chemical descriptors. Combining these different types of data can provide a more comprehensive and holistic understanding of organic and inorganic materials, enabling more accurate identification and characterization.

Real-Time and On-Site Identification. The development of portable and miniaturized analytical devices, coupled with machine learning algorithms, has the potential to enable real-time and on-site identification of organic and inorganic materials.

Automated Experimental Design. Machine learning algorithms can assist in the design and optimization of experiments for organic and inorganic identification. By analysing previous experimental data, these algorithms can suggest the most informative variables to measure or optimize, thereby reducing time, cost, and effort in the identification process.

Integration with High-Throughput Screening. High-throughput screening techniques, coupled with machine learning algorithms, can accelerate the discovery and identification of new organic and inorganic compounds. By rapidly screening large libraries of materials and correlating the results with known properties, machine learning models can predict the characteristics of untested compounds, enabling targeted synthesis and optimization.

Interpretability and Explainability Addressing the challenge of interpretability in machine learning models is an ongoing research focus. Efforts are being made to develop methods that can explain the reasoning behind the predictions made by these models, providing insights into the underlying factors contributing to organic and inorganic identification.

BIBLIOGRAPHY

[1] Pratami, S., Hertati, L., Puspitawati, L., Gantino, R., & Ilyas, M. (2021). Teknologi Inovasi Pengolahan Limbah Plastik Menjadi Produk UMKM Guna Menopang Ekonomi Keluarga Dalam Mencerdaskan Keterampilan Masyarakat. In GLOBAL ABDIMAS: Jurnal Pengabdian Masyarakat (Vol. 1, Issue 1, pp. 1– 11). Perkumpulan Intelektual Madani Indonesia. https://doi.org/10.51577/globalabdimas.v1i1.59

[2] F. Shaikh, N. Kazi, F. Khan, and Z. Thakur, “Waste profiling and analysis using machine learning,” in 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, 2020, pp. 488–492.

[3] W. Mulim, M. F. Revikasha, Rivandi and N. Hanafiah, "Waste Classification Using EfficientNet-B0," 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI), 2021, pp. 253-257, doi: 10.1109/ICCSAI53272.2021.9609756.

[4] Zhang Q, Yang Q, Zhang X, Bao Q, Su J, Liu X. Waste image classification based on transfer learning and convolutional neural network. Waste Manag. 2021 Nov;135:150-157. doi: 10.1016/j.wasman.2021.08.038. Epub 2021 Sep 8. PMID: 34509053.

[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] Zhang, Y., Xie, X., Cui, C., Liu, D., & Huang, B. (2021). Machine learning meets high-throughput experiments in inorganic materials synthesis: Recent advances and future perspectives. Journal of Materials Chemistry A, 9(45), 25463-25480.

[7] Huan, T. D., Mannodi-Kanakkithodi, A., Balachandran, P. V., & Lookman, T. (2020). Machine learning for molecular and materials science. Nature, 586(7829), 210-219

APPENDIX

#Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
import os
import keras
import tensorflow as tf
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Activation
from keras.layers import Dropout, Dense, Flatten, BatchNormalization
```

#Loading the dataset

```
train_dir = "C:/Users/Tharun/Downloads/Major/DATASET/TRAIN"
test_dir = "C:/Users/Tharun/Downloads/Major/DATASET/TEST"
```

#Data Preparation and Augmentation

```
train_datagen = ImageDataGenerator(rotation_range=10,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   vertical_flip = True,
                                   rescale=1./255,
                                   validation_split=0.2)
```

```
val_datagen = ImageDataGenerator(rescale = 1./255,
                                 validation_split=0.5)
#We use the flow_from_directory() method which allows to read images directly
from the directory and augment them while neural network is learning
Batch_size = 32

train_set = train_datagen.flow_from_directory(train_dir, class_mode='binary',
                                              batch_size = Batch_size,
                                              target_size=(64, 64))


val_set = val_datagen.flow_from_directory(test_dir, class_mode='binary',
                                          batch_size = Batch_size,
                                          target_size=(64, 64),
                                          subset= 'training')


test_set = val_datagen.flow_from_directory(test_dir, class_mode = 'binary',
                                           batch_size = Batch_size,
                                           target_size=(64, 64),
                                           subset= 'validation')
train_set.class_indices
train_set.image_shape
def print_sample_images(generator, num_images=5):
    images, labels = next(generator)
    class_names = list(generator.class_indices.keys())
    plt.figure(figsize=(15, 15))
    for i in range(num_images):
        plt.subplot(1, num_images, i + 1)
```

```
        plt.imshow(images[i])
        plt.title(class_names[int(labels[i])])
        plt.axis('off')
    plt.show()

#Printing some sample images from the dataset
print_sample_images(train_set)
def Images(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(15,15))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
    plt.tight_layout()
    plt.show()
augmented_img = [train_set[0][0][0] for i in range(5)]
Images(augmented_img)

#Creating the CNN model
model = Sequential([
  Conv2D(32, (3,3), activation='relu', padding='same', input_shape=(64, 64, 3)),
  MaxPool2D((2,2)),
  Conv2D(32, (3,3), activation='relu', padding='same'),
  MaxPool2D((2,2)),
  Conv2D(64, (3,3), activation='relu', padding='same'),
  MaxPool2D((2,2)),
  Flatten(),
  Dense(units=128, activation="relu"),
  Dense(units=1, activation="sigmoid")
])
```

```
model.compile(optimizer = 'adam', loss = 'binary_crossentropy',
              metrics = ['accuracy'])

model.summary()

#Training the model
history = model.fit(
   train_set,
   epochs = 5,
   validation_data= val_set,
   steps_per_epoch=int(len(train_set)/Batch_size),
   verbose=1
)
import seaborn as sns
sns.set(style="whitegrid")

#The plots of loss and accuracy on the training and validation sets
sns.set(style="whitegrid")
plt.plot(history.history['accuracy'], label='Train', linewidth=2, marker='o',
markersize=6)
plt.plot(history.history['val_accuracy'], label='Validation', linewidth=2, marker='o',
markersize=6)
plt.title('Model Accuracy', fontsize=16)
plt.ylabel('Accuracy', fontsize=14)
plt.xlabel('Epoch', fontsize=14)
plt.legend(fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```

```
plt.plot(history.history['loss'], label='Train', linewidth=2, marker='o', markersize=6)
plt.plot(history.history['val_loss'], label='Validation', linewidth=2, marker='o',
markersize=6)
plt.title('Model Loss', fontsize=16)
plt.ylabel('Loss', fontsize=14)
plt.xlabel('Epoch', fontsize=14)
plt.legend(fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
loss, accuracy = model.evaluate(test_set, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

#Creating the MobileNetV2 model

```
base_model = tf.keras.applications.MobileNetV2(include_top=False,
weights='imagenet', input_shape=(64, 64, 3))
base_model.trainable = False
model1 = Sequential([
    base_model,
    Flatten(),
    Dense(128, activation="relu"),
    Dropout(0.2),
    Dense(1, activation="sigmoid")
])
model1.compile(optimizer='adam',
               loss='binary_crossentropy',
               metrics=['accuracy'])
history1 = model1.fit(
    train_set,
    epochs=5,
```

```
    validation_data=val_set,
    verbose=1
)
def plot_graphs(history, string):
    sns.set(style="whitegrid")
    plt.plot(history.history[string])
    plt.plot(history.history['val_'+string])
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.legend([string, 'val_'+string])
    plt.show()
```

 # Ploting accuracy and loss for model1

```
plot_graphs(history1, "accuracy")
plot_graphs(history1, "loss")
loss, accuracy = model1.evaluate(test_set, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

#In this model we additional convolution layers to reduce the complexity of architecture in fully connected layer.

```
model2 = Sequential([
   Conv2D(32, (3,3), input_shape=(64, 64, 3), activation='relu', padding='same'),
   MaxPool2D((2,2)),
   Conv2D(32, (3,3), activation='relu', padding='same'),
   MaxPool2D((2,2)),
   Conv2D(64, (3,3), activation='relu', padding='same'),
   MaxPool2D((2,2)),
   Conv2D(128, (3,3), activation='relu', padding='same'),
   MaxPool2D((2,2)),
   Flatten(),
```

```
  Dense(64, activation="relu"),
  Dense(32, activation="relu"),
  Dense(1, activation="sigmoid")
])

model2.compile(optimizer = 'adam', loss = 'binary_crossentropy',
               metrics = ['accuracy'])

model2.summary()
#Training the model
history_2 = model2.fit(
   train_set,
   epochs = 10,
   validation_data= val_set,
   steps_per_epoch=int(len(train_set)/Batch_size),
   verbose=1
)
#Plots of loss and accuracy for the model
def plot_graphs(history, string):
   sns.set(style="whitegrid")
   plt.plot(history.history[string])
   plt.plot(history.history['val_'+string])
   plt.xlabel("Epochs")
   plt.ylabel(string)
   plt.legend([string, 'val_'+string])
   plt.show()

plot_graphs(history_2, "accuracy")
plot_graphs(history_2, "loss")
loss, accuracy = model2.evaluate(test_set, verbose=False)
```

```
print("Testing Accuracy:  {:.4f}".format(accuracy))
model2.save("waste_AG_model.h5")

#Giving the input image for the final model
test_image =
image.load_img(r'C:\Users\Tharun\Downloads\Major\DATASET\TRAIN\R\R_1.jpg'
, target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)



# predicting the result
result = model2.predict(test_image)
# dataset class index
train_set.class_indices
#Printing the Output
if result[0][0] == 1:
   prediction = ' Inorganic  Waste'
else:
   prediction = 'Organic Waste'
print(prediction)
```