A Major Project On

Waste Material Classification System using CNN

Submitted in partial fulfillment of the requirements for the award of the

Bachelor of Technology

In

Department of Computer Science and Engineering

By

R. Sai Ramana	**19241A05G2**
R. Revanth	**19241A05G3**
P. Pranay Prasad	**19241A05F6**
B. Harinath	**19241A05C5**

Under the Esteemed guidance of

G. Neelima

Assistant Professor

Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

GOKARAJU RANGARAJU

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

CERTIFICATE

This is to certify that the major project entitled "**Waste Material Classification System using CNN**" is submitted by **R. Sai Ramana (19241A05G2), R. Revanth(19241A05G3), P.Pranay Prasad (19241A05F6), B. Harinath(19241A05C5)** in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during academic year 2022-2023.

INTERNAL GUIDE	HEAD OF THE DEPARTMENT
G. Neelima	**Dr. K. MADHAVI**
Assistant Professor	**Professor**

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

There are many people who helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we would like to express our deep gratitude towards our internal guide **G. Neelima, Assistant Professor.** Department of CSE for his support in the completion of our dissertation.

We wish to express our sincere thanks to **Dr. K. Madhavi, HOD, Department of CSE** and to our principal **Dr. J. Praveen** for providing the facilities to complete the dissertation. We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

R. Sai Ramana (19241A05G2)

R. Revanth (19241A05G3)

P. Pranay Prasad (19241A05F6)

B. Harinath(19241A05C5)

DECLARATION

We hereby declare that the industrial major project entitled **“Crime rate analysis and Prediction”** is the work done during the period from **16th Dec 2022 to 3rd June 2023** and is submitted in the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad).The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

R. Sai Ramana (19241A05G2)

R. Revanth (19241A05G3)

P. Pranay Prasad (19241A05F6)

B. Harinath(19241A05C5)

ABSTRACT

The classification of waste material is a societal problem that affects sustainable development and people's livelihoods. The availability of models or tactics that assist individuals in sorting trash has grown to be crucial for the proper disposal of such garbage. Around the world, it is believed that waste management and the methodical sorting of waste helps in the development of the ecology. Our project's goal is to use a deep learning algorithm to classify waste materials like paper, cardboard, trash, glass, metal, and plastic so that they may be appropriately disposed of in the non- recyclable and recyclable bins. The proposed application can classify garbage efficiently to automate the recycling process. This approach can reduce the physical effort involved in the segregation of trash. The proposed application is solely based on CNN. The approach is to use a deep learning algorithm to classify waste materials like paper, cardboard, trash, glass, plastic, and metal so that they may be appropriately disposed of in the non-recyclable and recyclable bins. Convolutional Neural Network (CNN) is an effective method for classifying images. The idea of a convolutional neural network will be used to identify the image, together with an image processing technique that recognizes wastes based on their colour, shape, size, and dimension. In this approach, CNN's ResNet model is employed. ResNet is the abbreviation for Residual Network. We have successfully shown how trash may be precisely classified.

1. INTRODUCTION

The growth of solid waste in urban areas is a major problem, if not effectively handled, could lead to environmental destruction and be dangerous to human health. Intelligent waste management system is crucial to get rid of this problem. If we as a society don't do something to stop the buildup of non- recyclable waste on landfills all over the world, it could have a substantial impact on our way of life in the near future. Changes take a long time to occur and little effort is made to address the issue. As we all know, 17.7% of the world's population resides in India. A smart garbage management system is crucial considering the expansion of smart cities across India.

The purpose of our project is to build a real time application to classify or segregate the waste into different categories. By implementing our project "Waste Material Classification System using CNN" can reduce the physical efforts which included in the segregation process. Our project helps us to segregate the trash into six categories and it will show the details of that particular material. This will encourage individuals to recycle and reduce their waste.

Convolution Neural Network (CNN), a machine learning algorithm, was utilized as the model in our project and was applied to a dataset of photographs of trash. This technology guarantees the best waste management system and will accelerate the process with more accuracy. We use ResNet algorithm in our project. Our goal is to achieve best efficiency of garbage processing solution.

1.1 Existing System

For the classification of photographs, a variety of methods have been developed, including RNNs, SVMs, ANNs, and others.The system has been tested on various datasets of garbage images, and it has been able to categorize the junk images. The approach does, however, have certain drawbacks,including the need for a substantial amount of training data and the high computing cost of both training and testing. Additionally, the system has certain drawbacks, including the need for a substantial amount of training data and the high computational cost of building and evaluating AlexNet.

1.1.1. Limitations in Existing System

Compared to deep learning models like ResNet, SIFT and SVM are relatively straightforward algorithms, however they might not scale well to larger datasets. The computational cost and time needed to train and evaluate the models can become prohibitive as the number of images and classes rises. Garbage photos can vary widely in terms of colour, shape, texture, and lighting, making it difficult to extract features that can faithfully represent every image. The classification models may become less accurate as a result.

1.2 Proposed System

A new idea uses deep learning algorithms to separate the garbage at the beginning level, strengthening waste management. The devised method more accurately divides the garbage into various categories. This study examines the most efficient method for classifying different forms of garbage.
As opposed to simple neural networks, where we just keep stacking layers, training error does not grow as the network's depth increases. There are no new parameters introduced by identity mapping. Therefore, there is no increase in computing complexity. ResNet considerably improved the performance of neural networks with more layers when compared to neural networks with plain layers.
CNN has been granted an additional preferred position. CNN's engineering makes it extremely effective at photo recognition. Features are extracted from input photos. With the aid of the first few layers, the basic features are extracted.

1.2.1 Advantages over Existing System

ResNet provides a number of advantages over AlexNet and other older CNN architectures for classifying garbage images. ResNet can have up to hundreds of layers, which is far deeper than older CNN architectures without experiencing vanishing gradient problems. This enables it to pick up on more intricate traits, potentially enhancing categorization accuracy.
As a result, the gradient can move across the network more quickly during training because to ResNet's usage of residual connections, which skip over some network layers. As a result, training times may be sped up and deeper networks may be trained more easily.

2. LITERATURE STUDY

S.No.	Title of the Paper	Authors	Details
1	Classification of Trash for Recyclability Status	Mindy Yang, Gary Thung	Stanford University
2	Design of a Convolutional Neural Network Based Smart Waste Disposal System	Md. Samiul Haque sunny, Debopriya Roy Dipta, Shifat Hossian	IEEE
3	Auto-trash sorts garbage automatically at the TechCrunch disrupt hackathon	Jay Donovan	SMPM 2019

Table 1: Literature study

2.1. Classification of Trash for Recyclability Status

AUTHORS: Mindy Yang, Gary Thung

ABSTRACT: This classification challenge involves consumer taking the picture of material to recognize it and classify it into recycling material at recycling plant. Images with a single object on a pure white background are the input to the pipeline. The image is divided into six garbage class groups using an SVM and CNN. The category of garbage to which an object belongs can be predicted using computer vision from merely a picture.
For the initial round of waste sorting into recycling categories, an SVM was utilized. Because it is thought to be among the finest initial classification algorithms, the SVM was chosen. For multidimensional data, the SVM categorizes objects by defining a separating hyperplane. Largest minimum distance of training samples is given by the hyperplane. SIFT characteristics were employed for the SVM. In general, blob like characteristics are identified by the SIFT algorithm and 128 numbers are assigned to every image to describe it. The dataset uses SIFT features, which are potent values depending on the initial clustering. Since an image is converted to a histogram, the amount of time needed for SVM training is significantly decreased.

2.2. Design of a Convolutional Neural Network Based Smart Waste Disposal System

AUTHORS: Md. Samiul Haque Sunny, Shifat Hossian, Debopriya Roy Dipta

ABSTRACT: Recently, Bangladesh has faced a significant difficulty with regard to trash management, which is negatively affecting the environment. The idea of creating an intelligent embedded system and a smart garbage bin that resembles an automated teller machine is presented in this paper and is referred to as an Automated Teller Dustbin (ATD). An effective image classifier built on a convolutional neural network (CNN) is created, and it can analyze training features to detect and identify any object that is considered to be garbage. Additionally, it has the ability to count the quantity of identified items and give each one a price. As a result, it is feasible to exchange waste directly for its comparable price, which encourages people to make use the smart dustbin we suggest. By recycling the trash in it, you can offset the cost of installation and reduce the cost of operation and maintenance. A dataset of twenty photos out of which the ten classed objects was obtained from several trash disposal shops in Bangladesh, and used to train and evaluate an AlexNet-based CNN-based model.

2.3. Auto-trash sorts garbage automatically at the Tech Crunch disrupt hackathon

AUTHOR: J. Donovan

ABSTRACT: A Raspberry Pi module having camera is used to observe each item placed on the spinning can's top. The intelligent garbage can detect objects and spin top in accordance with the rotation of the item to drop it into the appropriate parts of a partitioned can using image identification, which is powered by their own unique software system developed on top of TensorFlow, Google's AI engine. The TensorFlow neural network would also be connected to numerous cans, and each can would be learning from the contents of the others as a result. The can only classify between degradable items and recyclable items. The network of cans would develop greater intelligence over time and improve its ability to sort.

2. SOFTWARE REQUIREMENT SPECIFICATIONS

3.1 IEEE Standard

1. Introduction

A crucial task for trash management and environmental sustainability is garbage classification. The traditional approach of manually sorting trash is labor-intensive, ineffective, and can be hazardous to the health of the workers involved. The creation of automated systems that can divide rubbish photos into groups like recyclable and non-recyclable items is therefore becoming increasingly popular. Transfer learning, which enables the reuse of previously trained models on new datasets, has demonstrated particular promise for deep learning techniques in this field. In this project, we want to create a garbage classification system using transfer learning and the ResNet architecture. On a batch of garbage photos, we will polish the pre-trained ResNet model and assess how well it performs inappropriately categorizing the images.

1.1. Purpose of the requirements document

The requirements document's primary purpose is to specify the system's functional and non-functional needs for the garbage classification project utilising transfer learning and ResNet. The aims of the project, the system's scope, and the particular features and functionalities that the system must have to achieve those goals are all outlined in this document. The specifications document will also act as a base for system testing and validation.

1.2. Scope of the product

The goal of the project is to create an automated system that can categorise rubbish photographs into several groups, such as recyclable and non-recyclable elements. The project will use ResNet and transfer learning to classify garbage. On a dataset of useless photos, the research will involve fine-tuning a pre- trained ResNet model. The system must accurately classify the photos with high recall rates and precision. The creation of a user-friendly user interface that enables users to interact with the system and acquire categorization results is also included in the project's scope. Additionally, the project will only take into account trash that is often found in domestic waste and will not take industrial or hazardous waste into account.

1.3. Definitions, acronyms, and abbreviations

CNN

The Convolutional Neural Network (CNN) is a deep learning neural network that is widely utilised in image andvideo recognition tasks. CNNs use many convolutional layers to apply filters to the input data. These filters recognize local features and patterns in the input image and apply them over the entire image. This allows the CNN to learn spatial feature hierarchies from the input data. CNNs have demonstrated substantial success in a variety of applications, including object detection, facial recognition, and picture classification, and are now an important component of computer vision and deep learning research.

SIFT

SIFT (Scale-Invariant Feature Transform) is a computer vision method that detects and describes local features in images. The algorithm is intended to find key features or places of interest in an image that are insensitive to scale, orientation, and affine distortion. SIFT does this by analysing the image's scale-space extrema with difference-of-Gaussian filters. SIFT computes a descriptor vector for each keypoint that encapsulates information about the surrounding image patch once the keypoints have been recognised.

SVM

Support vector machines include different supervised learning methods for data classification, outlier detection and regression analysis. Support vector machines are efficient in high-dimensional environments. SVM, or Support Vector Machine, is a popular supervised machine learning algorithm for classification and regression. SVM can handle linear and nonlinear data by transforming it into a higher-dimensional space where the data points can be separated linearly using kernel functions. Because of its capacity to handle high-dimensional and complicated data, SVM has been effectively employed in a variety of domains, including computer vision, natural language processing, and bioinformatics.

AlexNet

AlexNet is eight layer deep which is a convolutional neural network. AlexNet is used for the object detection, and it also solves the artificial intelligence problems. Convolutional layers employ a wide receptive field size and a short stride to allow the network to learn high-level picture information. To improve its accuracy, AlexNet employs techniques like as data augmentation, dropout, and ReLU activation. AlexNet's architecture influenced the development of deep learning and convolutional neural networks, paving the door for many later discoveries in computer vision.

ResNet50
Residual networks are CNNs which have more than 50 layers, such as ResNet 50. Residual Network is referred to as ResNet.The ResNet model consists of a fully linked layer and a number of ResNet blocks (combined convolution and identity blocks). The model was trained using the 1000-category Imagenet dataset; we will then delete the final fully connected layer and add a new fully connected layer, which produces six categories that indicate the likelihood that an image belongs to the category or type of garbage.

2. General description

A computer vision project that uses deep learning techniques to accurately classify different sorts of waste. The project entails developing a Residual Neural Network (ResNet) architecture for transfer learning that uses pre-trained models to minimise training time and enhance accuracy. The goal is to create a model that can categorise garbage into different types such as paper, plastic, metal, glass, and organic waste. For training the model, the project requires a huge dataset of rubbish photos that have been labelled and tagged. The finished model can then be used to automate waste management systems' rubbish sorting processes, minimising human labour and enhancing efficiency. Overall, the goal of this project is to address the global issue of incorrect garbage disposal and to encourage sustainable waste management practises.

2.1. Product perspective

It has the potential to revolutionize the waste management industry by providing an efficient and automated garbage sorting solution. The project's main objective is to develop a deep learning model that can accurately classify different types of garbage, such as paper, plastic, metal, glass, and organic waste. This model can be integrated into existing waste management systems to automate the garbage sorting process, reducing human labor and increasing efficiency. The project's success depends on the availability of a large dataset of labeled and annotated garbage images to train the ResNet model. Once trained, the model can be easily deployed on various hardware and software platforms. This makes the project highly scalable and adaptable to different waste management environments, from small-scale recycling centers to large municipal waste facilities.

2.2. Product functions

- Image recognition: The system should be capable of recognising photographs of various waste materials such as plastic, paper, metal, and organic garbage.
- Classification: The system should categorise the recognised photos based on the type of waste material.
- Transfer learning: The system should use the pre-trained ResNet model to train the CNN model for garbage categorization faster and more accurately.
- Accuracy: To increase trash management and recycling operations, the system should achieve high accuracy in rubbish classification.
- Data analysis: The system should analyse waste material data to find trends and patterns that can aid in waste management decision making.
- Integration: To boost overall efficiency, the system should be integrated with other waste management systems.

2.3. User characteristics

- Waste Management Authorities: Waste management authorities are the primary users of this project. They require an efficient system to identify, sort, and manage different types of waste materials to improve recycling efforts and minimize the environmental impact of waste disposal
- Recycling Companies: Recycling companies can also benefit from this project as they require high-quality and sorted waste materials for their recycling processes. The accurate classification of waste materials can enable them to streamline their operations and reduce cost Environmentalists: Environmentalists are users who are passionate about preserving the environment and reducing waste. They may use this project to monitor waste management practices and advocate for sustainable waste management policies.

2.4 General constraints

1. Computing Resources: The ResNet model used in this project is computationally intensive, which requires high computing resources. The availability of computing resources may be a constraint, especially for small-scale waste management authorities or recycling companies.
2. Limited Data: The availability of data on different types of waste materials may be limited, which can affect the accuracy of the model. This can be addressed by using data augmentation techniques to generate more training data.
3. Time Constraints: The training of the model may take a considerable amount of time, especially if the dataset is large. This can be addressed by using pre-trained models and transfer learning techniques to reduce the training time.
4. Cost: The cost of developing and deploying the system may be a constraint, especially for small-scale waste management authorities or recycling companies. This can be addressed by using open-source tools and technologies to reduce costs.
5. Accessibility: The system needs to be accessible to all users, including people with disabilities. The design and development of the system should take into account accessibility guidelines and standards.

2.5. Assumptions and dependencies

Assumptions

- Sufficient and representative data will be available for training and testing the ResNet model.
- The pre-trained ResNet model will be compatible with the software and hardware used for the project.
- The accuracy of the ResNet model will be sufficient for garbage classification purposes.
- The training and testing datasets will be balanced, and each class will have sufficient data.
- The waste materials' images will be of high quality and resolution, and the images will contain only the waste material to be classified.

Dependencies

- Availability of computing resources to train and test the ResNet model.
- Availability of software and hardware compatible with the pre-trained ResNet model.
- Access to a reliable and high-speed internet connection for data acquisition and deployment of the system.
- Availability of trained personnel with expertise in machine learning, deep learning, and computer vision.
- Support from the waste management authorities or recycling companies to provide data and insights into waste management practices and policies.
- Adherence to ethical guidelines and principles in the development and deployment of the system.

3. Specific requirements
3.1. Software Requirements

-> Operating system

- Windows 10,11 can be used

-> Coding language

- Python 3.8 is used in this project.Many useful analytics libraries are already installed in the Python 3 environment.

-> Coding environment

- A Python and R programming language distribution called Anaconda aims to make package deployment and administration in scientific computing simpler. Anaconda distribution consists of Jupyter Notebooks application.

- Jupyter Notebook is python IDE used for Machine Learning

- Data visualization, data cleaning, machine learning (ML), numerical simulation, and statistical modelling are all part of the platform

3.2. Hardware Requirements

The minimum hardware requirements differ greatly depending on the software Python, or VS Code user is working on. Quicker CPU is required.

-> Processor- Intel i3 6th gen and above

-> Speed- 1.1 GHz and above

-> RAM- 4GB (minimum),8GB (recommended)

-> Display – 1280*720 pixels

-> GPU- 4 GB

- GPUs perform faster calculations than CPU. So, we perform on GPU rather on CPU.

3.3. Functional Requirements

-The system shall use ResNet (Transfer Learning) to train the model for accurate garbage classification.
-The system shall be able to classify the garbage into different categories such as paper, plastic, glass, organic waste, and metal.
-The system shall be able to recognize and classify garbage images in real-time.
- The system shall provide an accurate percentage score for each garbage category.
- The system shall provide a user-friendly interface to interact with the garbage classification system.
- The system shall provide an option to add new categories of garbage to the classification system.

3.4. Non-Functional Requirements

Performance: The system shall have a fast response time for image recognition and classification.

Security: The application shall comply with data privacy regulations and protect user data from unauthorized access.

Compatibility: The system's shall be compatible with various operating systems like Windows, macOS, and Linux

Reliability: The application shall be able to detect emotions accurately at least 90% of the time.

3.5. Feasibility Study

3.5.1. Technical Feasibility

To train the ResNet model, the project will require hardware resources such as a computer with a powerful graphics processing unit (GPU). It is critical to guarantee that such resources are available and easily accessible in order to complete the project successfully. For implementation, software technologies such as Python, TensorFlow, and Keras will be required. These tools must be readily available and compatible with the hardware resources chosen. For training the ResNet model, the project will require a big dataset of rubbish photos. It is critical to guarantee that the dataset is accessible and free of legal or copyright concerns.

3.5.2. Economic Feasibility

By making the garbage classification process more effective, the initiative can assist cut waste management costs. With accurate waste type identification, resources may be allocated more efficiently, and the recycling process can be optimised, resulting in lower costs. Garbage classification and recycling are critical for reducing waste's negative environmental impact. The project has the potential to encourage sustainable practises and minimise pollution, resulting in lower health-care expenditures and higher productivity. The selling of recycled materials from garbage sorting has the potential to produce cash. The project can boost the efficiency of the recycling process by detecting different categories of waste, resulting in larger volumes of recycled materials that can be sold for a profit.

3.5.3. Operational Feasibility

To train the ResNet algorithm, the project requires a vast amount of data. Fortunately, several open-source datasets containing photos of various forms of garbage are available, making it easier to collect and identify the data required to train the model. ResNet (Transfer Learning), the technology employed in the project, is widely used in picture classification and is easily available through libraries such as TensorFlow and PyTorch. This accessibility enables for simple algorithm implementation and makes it possible to apply the technology to garbage classification.The initiative can be connected with existing waste management systems, making it easier to execute and requiring less infrastructure.

4. DESIGN

4.1. Project Description

Step 1: Importing Libraries

We will be using PyTorch library for Classifying.
PyTorch is based on the python programming languages and the Torch library. It is a machine learning framework(open source). One of the most in demand platforms used for research in deep learning areas. A faster transition from research prototyping to implementation is intended by the framework.
- Some of the libraries are torchvision, torch.nn.

Step 2: Loading the Dataset and Transforming

We load the Dataset and see the classes present in the Dataset. Then we apply Transformations for the Dataset.

- Data is transformed to improve its organization. It's possible that modified data is easier to machines.
- It not only improves data consistency but also protects from possible problems such as duplicates,null values, incompatible formats and wrong indexing, are two benefits of properly structured and verified data.
- transforms. Resize () and transform.ToTensor() modules are used.

transforms.Resize() : The input image is resized to the specified size by the Resize() transform. The torchvision.transforms module includes it as one of its transforms. Resize() takes tensor and PIL pictures.

transform.ToTensor(): A PyTorch Float Tensor of C, H, W shape with [0.0, 1.0] of range is created from a PIL picture consists of a pixel range of (0, 255).

Dataset

We use the Dataset from Kaggle which contain 2500 images of waste material considering different categories like cardboard, glass, plastic, paper, metal, and Trash.

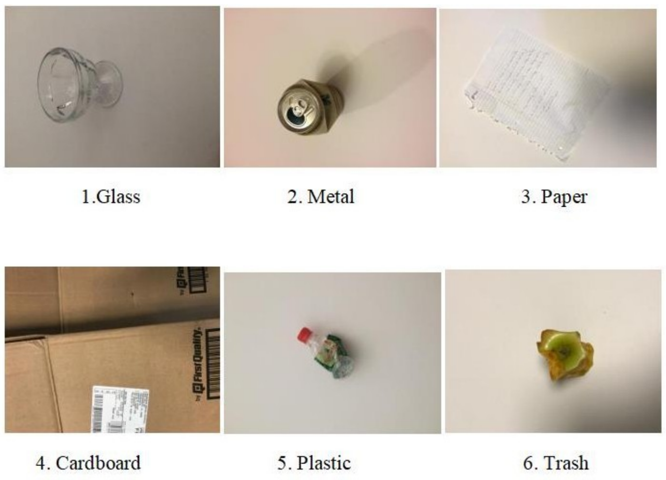
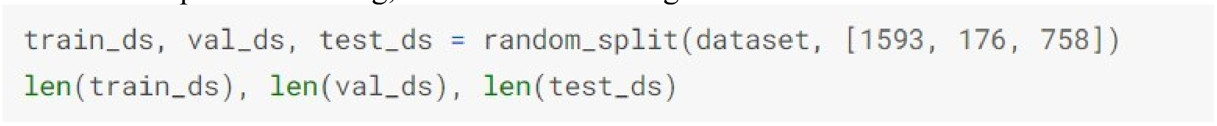


Fig 1.Dataset

Step 3: Splitting the Data

Data will be split into training, validation and testing datasets.



```
train_ds, val_ds, test_ds = random_split(dataset, [1593, 176, 758])
len(train_ds), len(val_ds), len(test_ds)
```

Fig 2. data split

Create training and validation dataloaders

One sample at a time, the Dataset retrieves the features and labels from our dataset. We often wish to pass samples in "mini batches" when training a model, reorder the data at each epoch to prevent overfitting, and utilise Python's multiprocessing to speed up data retrieval. DataLoader takes away this complexity for us and provides a simple API.

Porting to GPU

Graphics processing unit perform faster calculations than CPU. We can make use of the GPU for a faster speed of the training model.

Step 4: Training the Model

Fitting the model

model. train()

- The model. train() places the network's modules in training mode. It informs our model that we are now in the training phase, so the model keeps several layers active that behave differently depending on the phase, including dropout and batch-normalization.

--Training and validating the model for fixed number of epochs and for every epoch, the train losses and validation losses are calculated using loss functions and stored during the training and validation phases.

Optimizers
- To optimise our weights with the gradients, we will use Stochastic Gradient Descent (SGD). In our situation, we merely update the weights of the final layer.

backward() method.

The gradient is computed in the neural network's backward pass using Pytorch's backward() method. Gradients for the tensors are not calculated if this backward() method is not used. When requires grad is set to True, a tensor's gradient will be computed.

\

Forward() methodical
- Any real integer can be compressed using the element-wise PyTorch sigmoid function into a range between 0 and 1.

Step 5: Visualizing and Predicting results:

We will be visualizing the accuracies and losses for each epoch by creating graphs for them. This will help us understand the accuracy over every epoch. Visualizing the results helps to understand how the model is behaving during training and validation, and to identify any patterns or trends in the performance of the model. One way to visualize the results is by plotting the accuracy and loss for each epoch, which can help to identify overfitting or underfitting.The fine-tuning of the pre-trained model on the waste material classification task has improved its performance as compared to training the model from scratch and it has performed well on the six categories of waste materials.

We decreased the batch size to be more suitable for the size of the dataset. Lastly, we employed a weight initialization technique to enhance the model's learning ability. These changes were made to optimize the performance of the model and improve its accuracy.

Step 6: Generating predictions for test dataset

The test set images will be loaded and pre-processing procedures similar to that of the training set will be applied.
Transformations similar to those used on the training and validation datasets should also be applied to the test dataset.
We will then generate predictions for the test set. We can also generate predictions for other external images using this trained model.

4.2. Unified Modeling Language

4.2.1. Class Diagram

Class diagrams represents different types of objects and relationship in the system. The class diagram is used to demonstrate, visualize, describe different aspects in the system. It has different types of attributes. It is also used to design and analyze the view of application. Class diagrams are also using the reverse and forward engineering. Complex systems use Class diagrams to represent. Different classes have their functionalities.

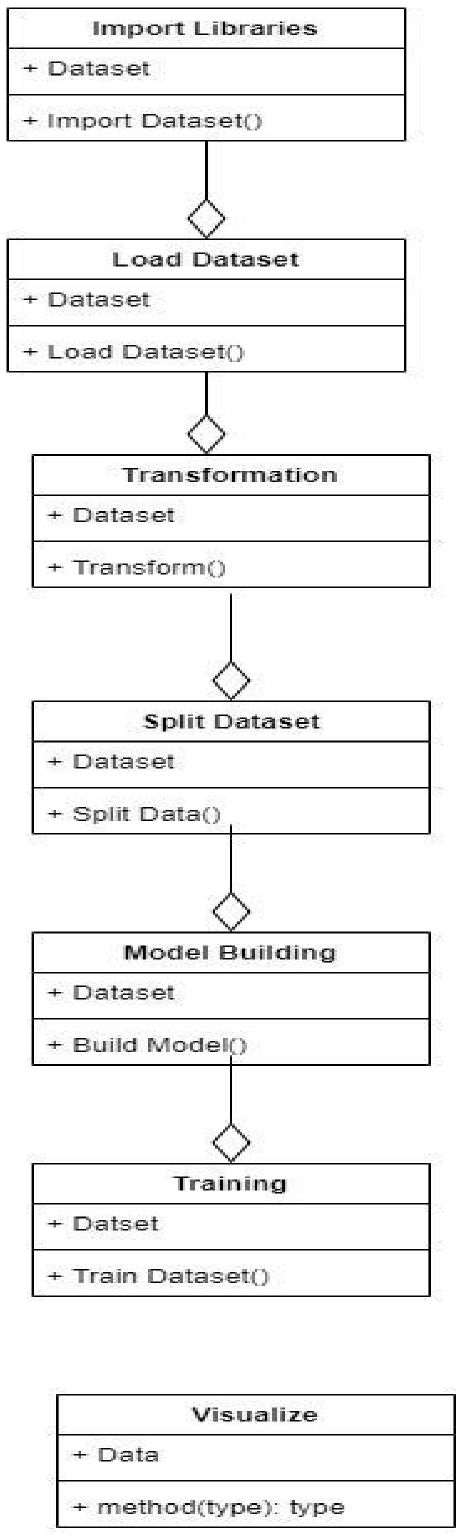


Fig 3. Class Diagram

4.2.2. Use case Diagram

A use Case is a Behavioural diagram in the Unified Modelling language (UML) used to form the use cases between the user and elements or use case analysis. Its primary task a is to present a graphical representation of the functionality of a system in terms of actors, their goals (Represented as use cases), and any interdependencies between those use cases. The main goal of a use case diagram is to show which actor performs which system functions. Roles can serve as representations of the system's actors.

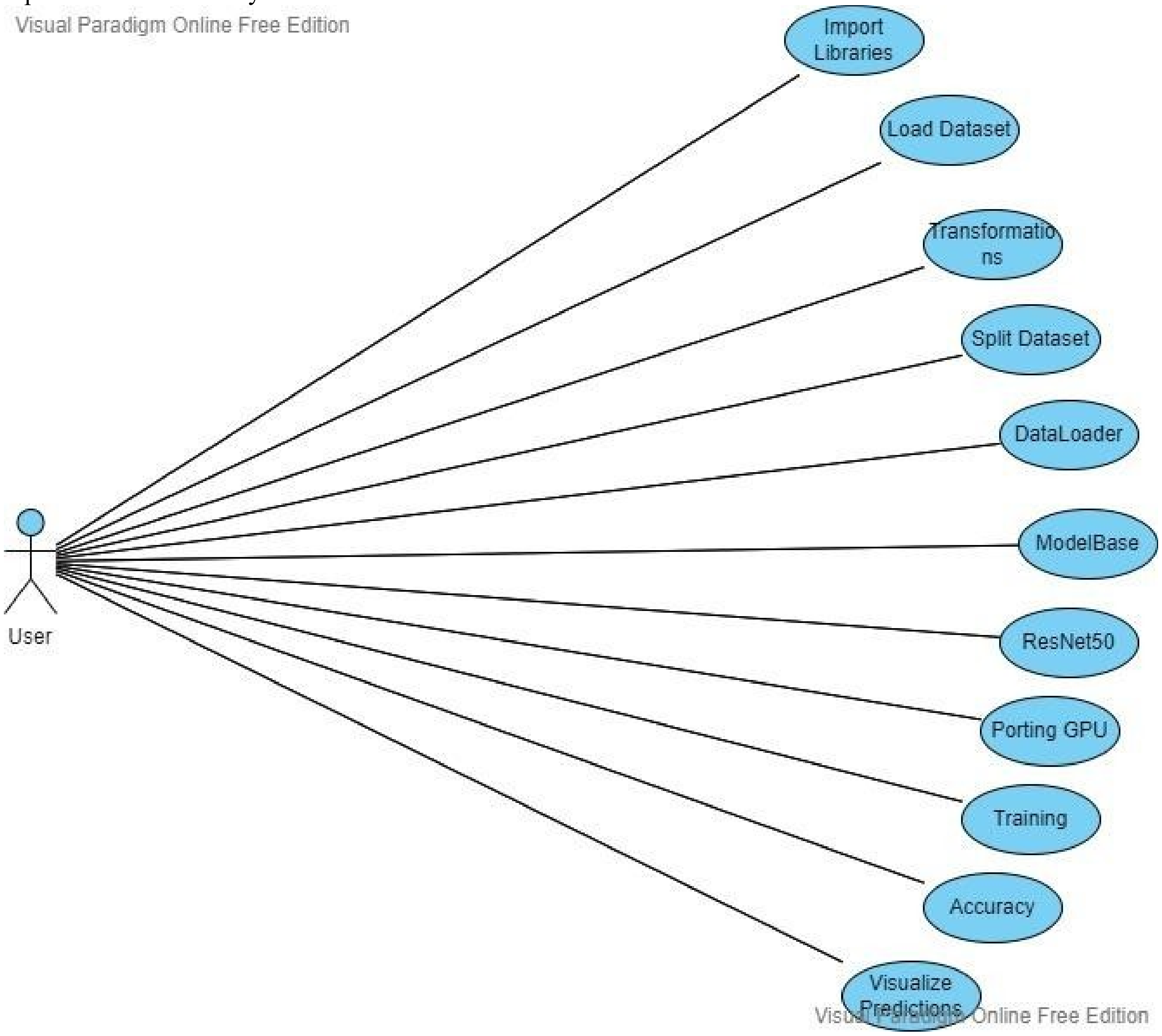


Fig 4. Use case Diagram

4.2.3. Object Diagram

Object diagram is a type of class diagram. A class's instances are objects. In essence, this means that while the system is functioning, an object represents the state of a class at a specific time. The object diagram represents the current state of the many classes in the system, as well as any interactions or associationsbetweenthem.

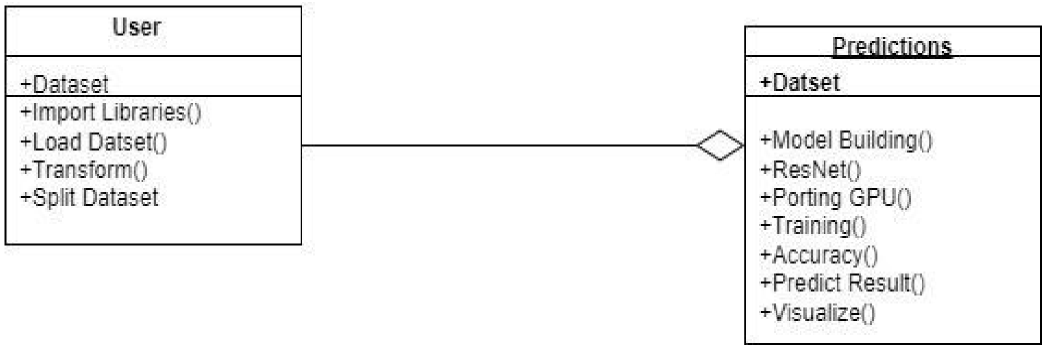


Fig 5. Object Diagram

4.2.4. Collaboration Diagram

Collaboration diagrams are also known as Communication diagram. It is used to analyze the relationship between different objects. Notations of collaboration diagram are actor, object, link and message.

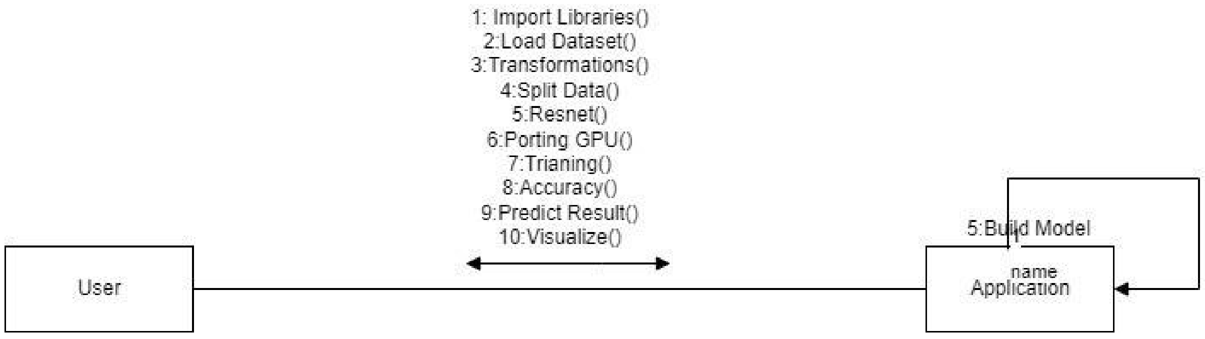


Fig 6. Collaboration Diagram

4.2.5. State Diagram

A state diagram describes the various states that system objects experience over their lifetimes. The properties of system objects change as a result of events. A state diagram also describes how an object's state changes in response to system events, moving from its starting state to its final.

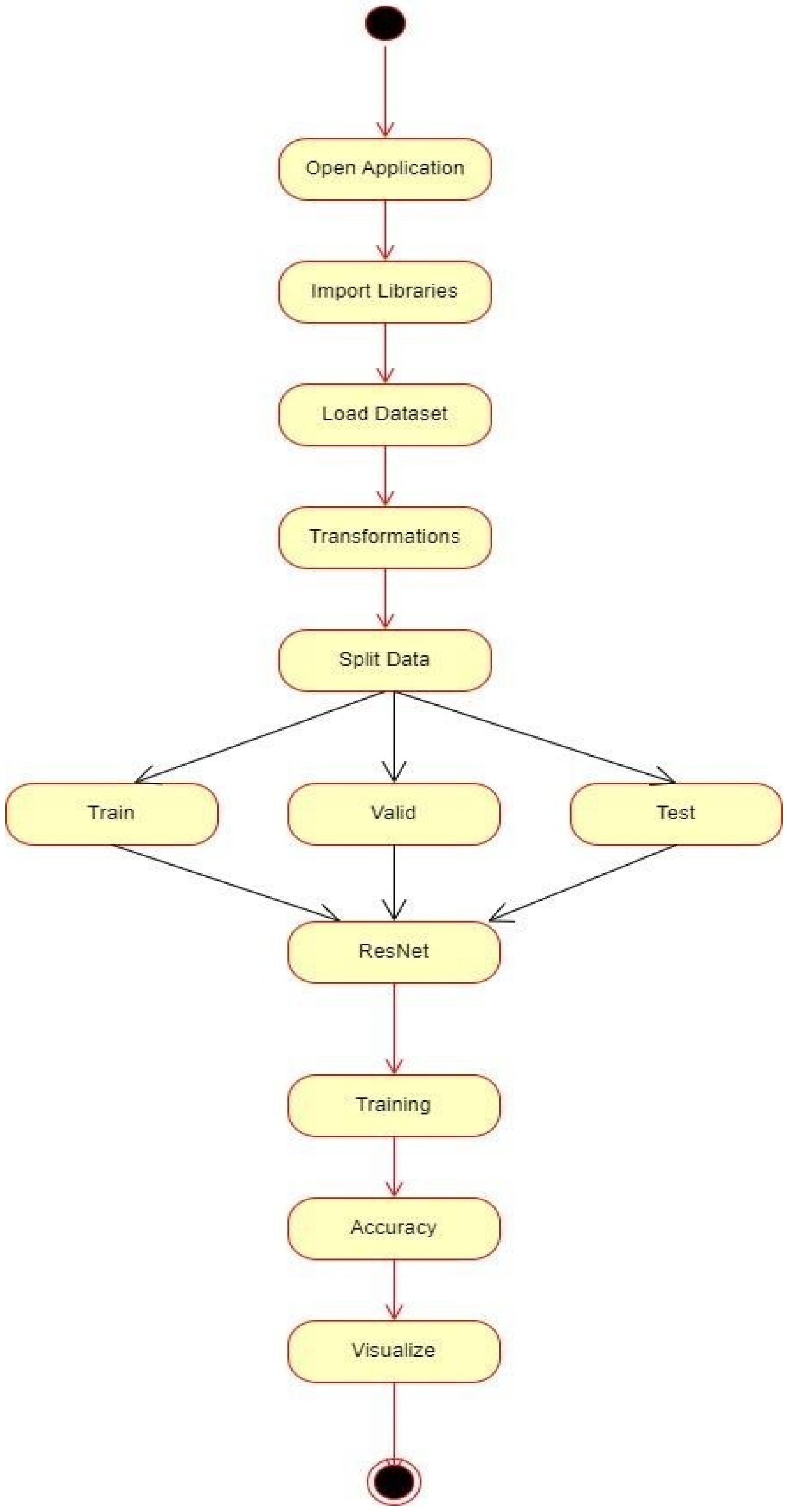


Fig 7. State Diagram

5. IMPLEMENTATION

5.1 Source code (Main)

```
In [4]: import os
        import torch
        import torchvision
        from torch.utils.data import random_split
        import torchvision.models as models
        import torch.nn as nn
        import torch.nn.functional as F
```

```
In [7]: data_dir  = 'C:/Users/ragipani sai ramana/OneDrive/Desktop/Garbage classification'

        classes = os.listdir(data_dir)
        print(classes)

        ['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']
```

```
In [8]: from torchvision.datasets import ImageFolder
        import torchvision.transforms as transforms

        transformations = transforms.Compose([transforms.Resize((256, 256)), transforms.ToTensor()])

        dataset = ImageFolder(data_dir, transform = transformations)
```

```
In [9]: import matplotlib.pyplot as plt
        %matplotlib inline

        def show_sample(img, label):
            print("Label:", dataset.classes[label], "(Class No: "+ str(label) + ")")
            plt.imshow(img.permute(1, 2, 0))
```

```
In [10]: img, label = dataset[12]
         show_sample(img, label)

         Label: cardboard (Class No: 0)
```

```
In [12]: random_seed = 42
         torch.manual_seed(random_seed)

Out[12]: <torch._C.Generator at 0x209b298eed0>
```

```
In [13]: train_ds, val_ds, test_ds = random_split(dataset, [1593, 176, 758])
         len(train_ds), len(val_ds), len(test_ds)

Out[13]: (1593, 176, 758)
```

```
In [14]: from torch.utils.data.dataloader import DataLoader
         batch_size = 32
```

```
In [15]: train_dl = DataLoader(train_ds, batch_size, shuffle = True, num_workers = 4, pin_memory = True)
         val_dl = DataLoader(val_ds, batch_size*2, num_workers = 4, pin_memory = True)
```

```
In [16]: from torchvision.utils import make_grid

         def show_batch(dl):
             for images, labels in dl:
                 fig, ax = plt.subplots(figsize=(12, 6))
                 ax.set_xticks([])
                 ax.set_yticks([])
                 ax.imshow(make_grid(images, nrow = 16).permute(1, 2, 0))
                 break
```

```
In [17]: show_batch(train_dl)
```

```
In [18]: def accuracy(outputs, labels):
             _, preds = torch.max(outputs, dim=1)
             return torch.tensor(torch.sum(preds == labels).item() / len(preds))

         class ImageClassificationBase(nn.Module):
             def training_step(self, batch):
                 images, labels = batch
                 out = self(images)                  # Generate predictions
                 loss = F.cross_entropy(out, labels) # Calculate loss
                 return loss

             def validation_step(self, batch):
                 images, labels = batch
                 out = self(images)                    # Generate predictions
                 loss = F.cross_entropy(out, labels)   # Calculate loss
                 acc = accuracy(out, labels)           # Calculate accuracy
                 return {'val_loss': loss.detach(), 'val_acc': acc}

             def validation_epoch_end(self, outputs):
                 batch_losses = [x['val_loss'] for x in outputs]
                 epoch_loss = torch.stack(batch_losses).mean()   # Combine losses
                 batch_accs = [x['val_acc'] for x in outputs]
                 epoch_acc = torch.stack(batch_accs).mean()      # Combine accuracies
                 return {'val_loss': epoch_loss.item(), 'val_acc': epoch_acc.item()}

             def epoch_end(self, epoch, result):
                 print("Epoch {}: train_loss: {:.4f}, val_loss: {:.4f}, val_acc: {:.4f}".format(
                     epoch+1, result['train_loss'], result['val_loss'], result['val_acc']))
```

```
In [19]: class ResNet(ImageClassificationBase):
             def __init__(self):
                 super().__init__()
                 # Use a pretrained model
                 self.network = models.resnet50(pretrained=True)
                 # Replace last layer
                 num_ftrs = self.network.fc.in_features
                 self.network.fc = nn.Linear(num_ftrs, len(dataset.classes))

             def forward(self, xb):
                 return torch.sigmoid(self.network(xb))

         model = ResNet()

         C:\anaconda\lib\site-packages\torchvision\models\_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.1
         3 and will be removed in 0.15, please use 'weights' instead.
           warnings.warn(
         C:\anaconda\lib\site-packages\torchvision\models\_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for
         'weights' are deprecated since 0.13 and will be removed in 0.15. The current behavior is equivalent to passing `weights=ResNet5
         0_Weights.IMAGENET1K_V1`. You can also use `weights=ResNet50_Weights.DEFAULT` to get the most up-to-date weights.
           warnings.warn(msg)
         Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to C:\Users\ragipani sai ramana/.cache\torch\hub\check
         points\resnet50-0676ba61.pth

           0%|          | 0.00/97.8M [00:00<?, ?B/s]
```

```
In [30]: def get_default_device():
             """Pick GPU if available, else CPU"""
             if torch.cuda.is_available():
                 return torch.device('cuda')
             else:
                 return torch.device('cpu')

         def to_device(data, device):
             """Move tensor(s) to chosen device"""
             if isinstance(data, (list,tuple)):
                 return [to_device(x, device) for x in data]
             return data.to(device, non_blocking=True)

         class DeviceDataLoader():
             """Wrap a dataloader to move data to a device"""
             def __init__(self, dl, device):
                 self.dl = dl
                 self.device = device

             def __iter__(self):
                 """Yield a batch of data after moving it to device"""
                 for b in self.dl:
                     yield to_device(b, self.device)

             def __len__(self):
                 """Number of batches"""
                 return len(self.dl)
```

```
In [31]: device = get_default_device()
         device

Out[31]: device(type='cpu')
```

```
In [22]: train_dl = DeviceDataLoader(train_dl, device)
         val_dl = DeviceDataLoader(val_dl, device)
         to_device(model, device)

                 (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                 (relu): ReLU(inplace=True)
                 (downsample): Sequential(
                   (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
                   (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                 )
               )
               (1): Bottleneck(
                 (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
                 (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                 (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
                 (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                 (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
                 (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                 (relu): ReLU(inplace=True)
               )
               (2): Bottleneck(
                 (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
                 (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

In [23]:
```
@torch.no_grad()
def evaluate(model, val_loader):
    model.eval()
    outputs = [model.validation_step(batch) for batch in val_loader]
    return model.validation_epoch_end(outputs)

def fit(epochs, lr, model, train_loader, val_loader, opt_func=torch.optim.SGD):
    history = []
    optimizer = opt_func(model.parameters(), lr)
    for epoch in range(epochs):
        # Training Phase
        model.train()
        train_losses = []
        for batch in train_loader:
            loss = model.training_step(batch)
            train_losses.append(loss)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()
        # Validation phase
        result = evaluate(model, val_loader)
        result['train_loss'] = torch.stack(train_losses).mean().item()
        model.epoch_end(epoch, result)
        history.append(result)
    return history
```

In [24]:
```
model = to_device(ResNet(), device)
```

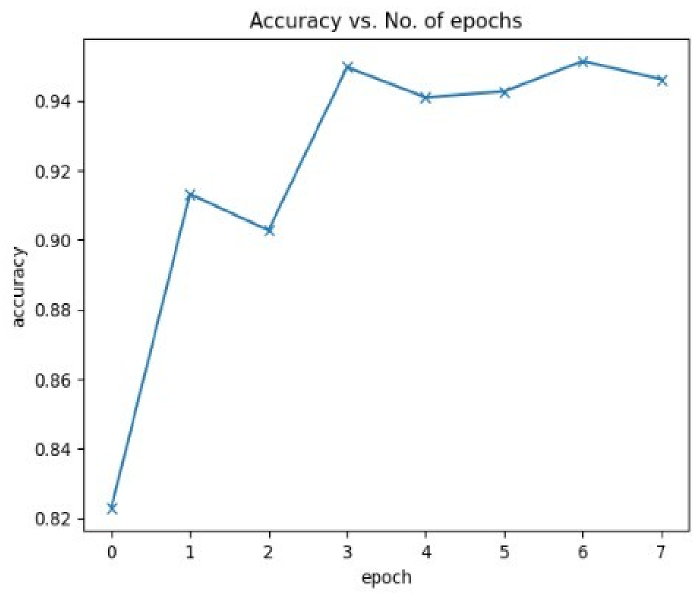
In [25]:
```
evaluate(model, val_dl)
```

Out[25]:
```
{'val_loss': 1.8006811141967773, 'val_acc': 0.1371527761220932}
```

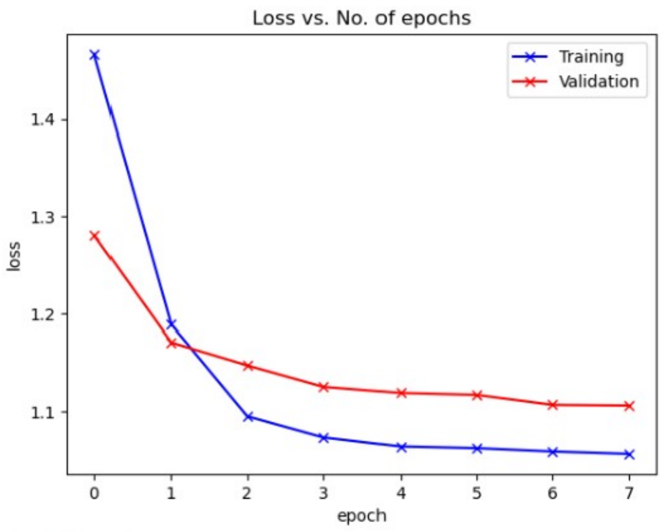
In [26]:
```
num_epochs = 8
opt_func = torch.optim.Adam
lr = 5.5e-5

history = fit(num_epochs, lr, model, train_dl, val_dl, opt_func)
```

```
Epoch 1: train_loss: 1.4671, val_loss: 1.2809, val_acc: 0.8229
Epoch 2: train_loss: 1.1896, val_loss: 1.1710, val_acc: 0.9132
Epoch 3: train_loss: 1.0950, val_loss: 1.1474, val_acc: 0.9028
Epoch 4: train_loss: 1.0731, val_loss: 1.1254, val_acc: 0.9497
Epoch 5: train_loss: 1.0638, val_loss: 1.1191, val_acc: 0.9410
Epoch 6: train_loss: 1.0620, val_loss: 1.1170, val_acc: 0.9427
Epoch 7: train_loss: 1.0585, val_loss: 1.1068, val_acc: 0.9514
Epoch 8: train_loss: 1.0560, val_loss: 1.1061, val_acc: 0.9462
```

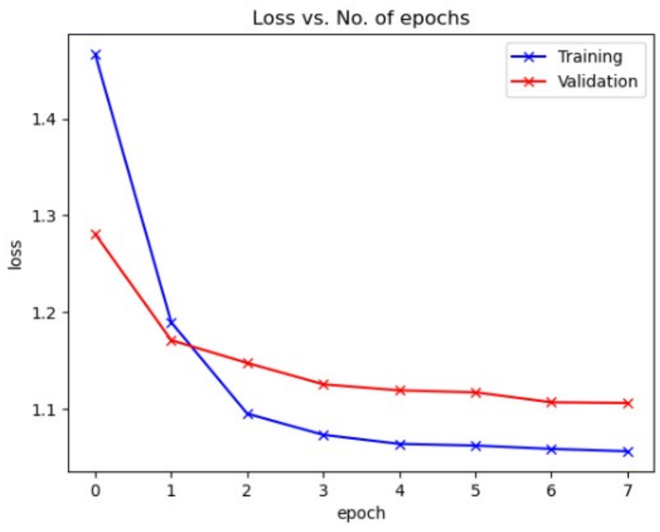
In [32]:
```
def plot_accuracies(history):
    accuracies = [x['val_acc'] for x in history]
    plt.plot(accuracies, '-x')
    plt.xlabel('epoch')
    plt.ylabel('accuracy')
    plt.title('Accuracy vs. No. of epochs');

plot_accuracies(history)
```



```
In [33]: def plot_losses(history):
             train_losses = [x.get('train_loss') for x in history]
             val_losses = [x['val_loss'] for x in history]
             plt.plot(train_losses, '-bx')
             plt.plot(val_losses, '-rx')
             plt.xlabel('epoch')
             plt.ylabel('loss')
             plt.legend(['Training', 'Validation'])
             plt.title('Loss vs. No. of epochs');

         plot_losses(history)
```



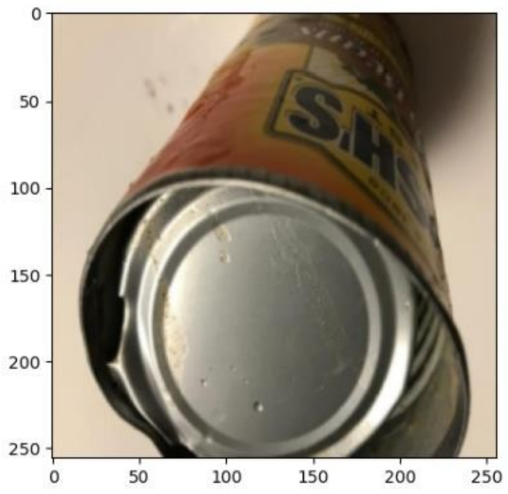
```
In [33]: def plot_losses(history):
             train_losses = [x.get('train_loss') for x in history]
             val_losses = [x['val_loss'] for x in history]
             plt.plot(train_losses, '-bx')
             plt.plot(val_losses, '-rx')
             plt.xlabel('epoch')
             plt.ylabel('loss')
             plt.legend(['Training', 'Validation'])
             plt.title('Loss vs. No. of epochs');

         plot_losses(history)
```



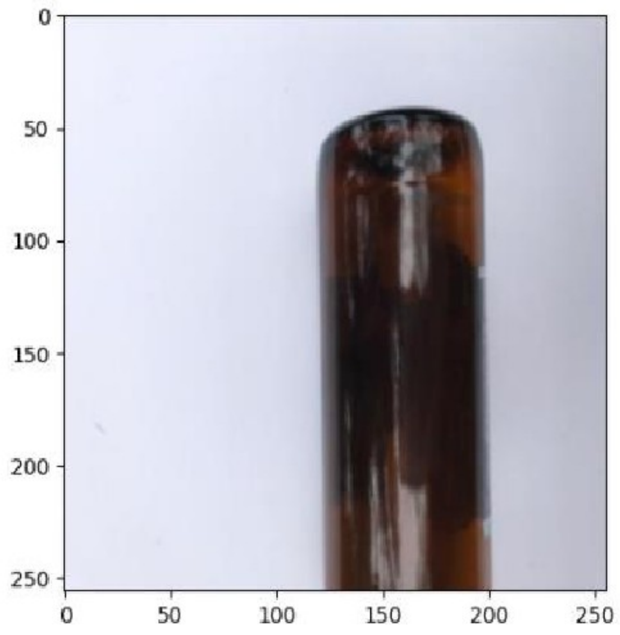
```
In [34]: def predict_image(img, model):
             # Convert to a batch of 1
             xb = to_device(img.unsqueeze(0), device)
             # Get predictions from model
             yb = model(xb)
             # Pick index with highest probability
             prob, preds  = torch.max(yb, dim=1)
             # Retrieve the class label
             return dataset.classes[preds[0].item()]
```

```
In [34]: def predict_image(img, model):
             # Convert to a batch of 1
             xb = to_device(img.unsqueeze(0), device)
             # Get predictions from model
             yb = model(xb)
             # Pick index with highest probability
             prob, preds  = torch.max(yb, dim=1)
             # Retrieve the class label
             return dataset.classes[preds[0].item()]
```

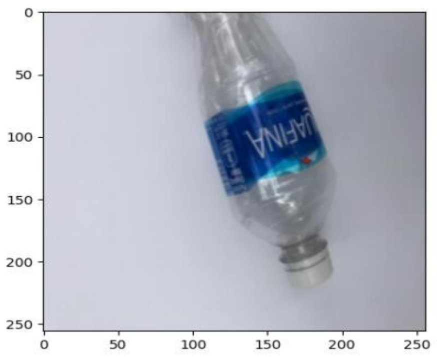
```
In [35]: img, label = test_ds[17]
         plt.imshow(img.permute(1, 2, 0))
         print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))

         Label: metal , Predicted: metal
```



```
In [36]: img, label = test_ds[23]
         plt.imshow(img.permute(1, 2, 0))
         print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))

         Label: glass , Predicted: glass
```



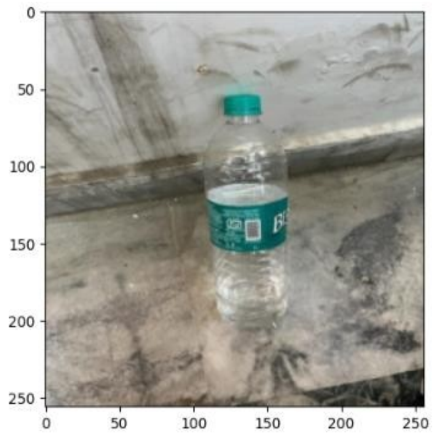
```
In [26]: img, label = test_ds[51]
         plt.imshow(img.permute(1, 2, 0))
         print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))

         Label: plastic , Predicted: plastic
```



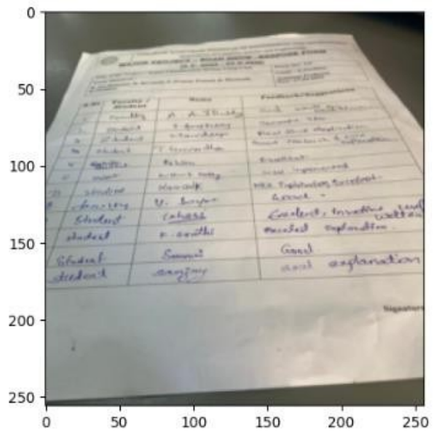
```
In [27]: loaded_model = model

In [28]: from PIL import Image
         from pathlib import Path

         def predict_external_image(image_name):
             image = Image.open(Path('./' + image_name))

             example_image = transformations(image)
             plt.imshow(example_image.permute(1, 2, 0))
             print("The image resembles", predict_image(example_image, loaded_model) + ".")
```

5.1 . Results

```
In [40]: predict_external_image('C:/Users/ragipani sai ramana/OneDrive/Desktop/bottle.jpg')

         The image resembles plastic.
```



```
In [43]: predict_external_image('C:/Users/ragipani sai ramana/OneDrive/Desktop/paper1.jpg')

         The image resembles paper.
```



6. TESTING

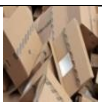
Test Cas e No.	Test Scenario	Test Data	Expected Result	Actual Result	Status Pass/Fail
1	Garbage Image		Plastic	Plastic	Pass
2	Garbage Image		Paper	Paper	Pass
3	Garbage Image		Metal	Metal	Pass
4	Garbage Image		Cardboar d	Cardboard	Pass
5	Garbage Image		glass	glass	Pass
6	Garbage Image		trash	trash	Pass

Table 2

7. CONCLUSION AND FUTURE SCOPE

CONCLUSION

The use of machine learning and computer vision algorithms allows for sorting trash into different recycling categories. However, the diverse range of possible data can make it difficult to create a highly accurate system. To improve accuracy, it is necessary to have a vast and continually expanding dataset. The project's aims to solve waste management by precisely categorising various sorts of garbage. The system trains the model with ResNet, a deep learning method, to obtain high accuracy in garbage categorization.

This project has the potential to greatly enhance trash management and lessen garbage's negative environmental impact. Better trash disposal, recycling, and reuse practises will result from proper rubbish classification, resulting in a cleaner and greener environment.

FUTURE SCOPE

We are looking to broaden the scope of this project by including the capability to recognize and categorize multiple objects within a single image or video. This would increase the efficiency of recycling facilities by allowing them to process a greater volume of materials. Additionally, we want to incorporate multiple object detection and classification to improve the accuracy of large-scale recycling material classification. Furthermore, we plan to continue expanding our dataset by adding more images and potentially more categories, with the ultimate goal of releasing it for public use.

8. REFERENCES

1. Thung, Gary and M. Yang. "Classification of Trash for Recyclability Status."

2. J. Donovan, "No Title," Auto-trash sorts garbage automatically at the techerunch disrupt hackathon, 2016. [Online]. Available:https://techcrunch.com/2016/09/13/auto-trash-sortsgarbage-%0Aautomatically-at-the-techcrunch-disrupt-hackathon/.

3. B. Batinic, S. Vukmirovic, G. Vujic, N. Stanisavljevic, D. Ubavin, and G. Vukmirovic, "Using ANN model to determine future waste characteristics in order to achieve specific waste management targets - case study of Serbia," J. Sei. Ind. Res. (India)., 2011.

4. C. Liu, L. Sharan, E. H. Adelson, and R. Rosenholtz, "Exploring features in a bayesian framework for material recognition," in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010, pp. 239–246.

5.Smartphone-app-to-detect-garbage-using-Mittal-Yagnik

6.Towardsdatascience.com/advanced-waste-classification-with-machine-learning.

7. iq.opengenus.org/resnet50-architecture/.

8. Trash Classification Using Convolutional Neural Networks Project Category: Computer Vision

9. www.researchgate.net/publication/34693751Trash_Classification_Classifying_garbage_using_Deep_Learning.

10.www.ibm.com/in-en/topics/convolutional-neural-networks.

11.Design of convolutional Neural Network Based Smart Waste Disposal System-Md. Samiul Haque Sunny, Debopriya Roy Dipta, Shifat Hoss

CSE-C9-1

ORIGINALITY REPORT

7%	4%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.scilit.net Internet Source	2%
2	Submitted to Gitam University Student Paper	1%
3	"ICASERT 2019 Table of Contents", 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), 2019 Publication	1%
4	www.researchgate.net Internet Source	<1%
5	jcreview.com Internet Source	<1%
6	Abdul Rajak A. R., Shazia Hasan, Bushra Mahmood. "Automatic waste detection by deep learning and disposal system design", Journal of Environmental Engineering and Science, 2020 Publication	<1%
7	Submitted to University of South Australia	

	Student Paper	<1 %
8	Submitted to Higher Education Commission Pakistan Student Paper	<1 %
9	Submitted to Sim University Student Paper	<1 %
10	Submitted to American University of the Middle East Student Paper	<1 %
11	Submitted to Queen's University of Belfast Student Paper	<1 %
12	hugrypiggykim.com Internet Source	<1 %
13	Submitted to ABES Engineering College Student Paper	<1 %
14	ryanwingate.com Internet Source	<1 %
15	Submitted to Al Quds University Student Paper	<1 %
16	www.coursehero.com Internet Source	<1 %