Intelligent Garbage Classification using Deep Learning

A PROJECT REPORT SUBMITTED BY:

TEAM ID : LTVIP2023TMID04375

TEAM SIZE : 4

TEAM LEADER : BHUMIREDDY GARI TEJASWINI

TEAM MEMBER : J.S.VIJAY

TEAM MEMBER : GUNDRAJU SASI

TEAM MEMBER : CHOWDAVARAM DEEPTHI

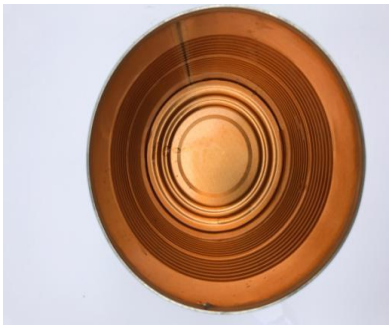
Table contents:

INTRODUCTION

According to the next 25 years, the less developed countries' waste accumulation will increase drastically. With the increase in the number of industries in the urban area ,the disposal of the solid waste is really becoming a big problem, and the solid waste includes paper, wood, plastic, metal, glass etc. The common way of managing waste is burning waste and this method can cause air pollution and some hazardous materials from the waste spread into the air which can cause cancer. Hence it is necessary to recycle the waste to protect the environment and human beings' health, and we need to separate the waste into different components which can be recycled using different ways.

CARDBOARD

GLASS



METAL

PAPER

PLASTIC

TRASH

1.1 Project Overview

- **Problem:** The accumulation of solid waste in the urban area is becoming a great concern, and it would result in environmental pollution and may be hazardous to human health if it is not properly managed. It is important to have an advanced/intelligent waste management system to manage a variety of waste materials.
- **Solution:** And this system can take short time to sort the waste and it will be more accurate in sorting than the manual way. With the system in place, the beneficial separated waste can be recycled and converted to energy and fuel for the growth of the economy.
- **Benefits:** The system that is developed for the separation of the accumulated waste is based on the combination of Convolutional Neural Network.

The project will be conducted in three phases:

1. **Data Collection: A** dataset of various garbage images will be collected from a variety of sources. The images will be labeled as paper, plastic, glass, trash, metal or cardboard by a team.
2. **Model development:** A deep learning model will be developed to classify different garbage images. The model will be trained on the labeled dataset of garbage images.
3. **Evaluation:** The performance of the model will be evaluated on a held-out test set of garbage classification images. The model will be evaluated on its accuracy, sensitivity, and specificity.

1.2 Purpose

- ### Efficient Waste Management:

  Deep learning-based garbage classification enables efficient waste segregation, leading to streamlined waste management processes.

  Proper segregation helps in recycling and reusing materials, reducing the burden on landfills.

- ### Environmental Conservation:

  By identifying recyclable materials accurately, deep learning helps promote sustainable practices and reduces environmental pollution.

  It encourages responsible disposal of hazardous waste, minimizing its negative impact on ecosystems.

- **Automated Sorting:**

  Intelligent garbage classification with deep learning enables automated sorting of waste at various stages, saving time and human resources.

- **Data-Driven Decision Making:**

  Deep learning algorithms provide data insights on the types and quantities of waste generated, aiding policymakers in making informed decisions.

  Governments and organizations can strategize waste management plans and allocate resources more effectively.

- **Encouraging Circular Economy:**

  Accurate garbage classification facilitates the recovery of valuable resources from waste, promoting a circular economy model.

  Recovered materials can be used in manufacturing and production processes, reducing the demand for virgin resources.

2. IDEATION & PROPOSED SOLUTION

2.1Problem Statement and Definition

Problem Statement:

Intelligent garbage classification using deep learning aims to develop an automated system that can accurately categorize and sort waste into various classes, such as recyclable, non-recyclable, organic, hazardous, and others. The primary objective is to improve waste management processes, increase recycling rates, reduce environmental pollution, and promote sustainable practices.

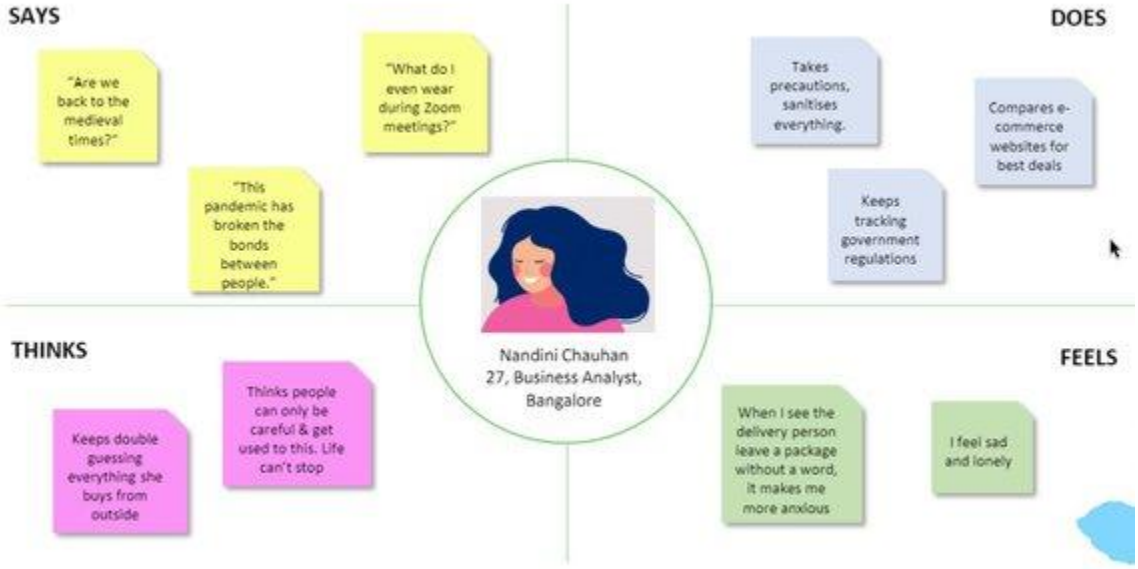
Definition:

Intelligent garbage classification using deep learning involves the use of advanced artificial intelligence techniques, particularly deep learning algorithms, to analyze images or sensor data of waste items and identify their respective categories. The system employs convolutional neural networks (CNNs) or other deep learning architectures to learn discriminative features from the data and make accurate predictions. The main components of the system typically include data collection, data preprocessing, model training, and inference.

Key characteristics and requirements of the intelligent garbage classification system using deep learning may include real-time processing, scalability, adaptability to different waste types, robustness against noise and uncertainties, and the ability to handle a large variety of waste items commonly encountered in everyday life.

2.2 Empathy Map Canvas

An empathy map canvas for intelligent garbage classification using deep learning can help us understand the thoughts, feelings, needs, and challenges of various stakeholders involved in the implementation and adoption of such a system. Let's create an empathy map canvas focusing on key stakeholders.



2.3 Ideation and Brainstorming

- **Real-time Feedback for Users:**

  Implement a real-time feedback system that provides users with instant notifications or audio-visual cues when they incorrectly dispose of waste.
  This feedback loop can help in correcting sorting mistakes and reinforcing proper waste disposal habits.

- **Waste Segmentation and Volume Estimation:**

  Extend the deep learning model to not only classify waste but also segment and estimate the volume of different waste materials.

This information can aid waste management companies in planning efficient waste transportation and storage.

- **Zero-Waste Initiatives:**

  Integrate the intelligent garbage classification system with zero-waste initiatives, guiding users towards adopting a zero-waste lifestyle.
  Offer personalized recommendations and tips to reduce waste generation.

- **Incentive-Based Recycling Program:**

  Design a loyalty or rewards program that incentivizes users to recycle more by providing points or discounts for proper waste disposal.
  Deep learning can track and reward users based on their recycling efforts.

2.4 Proposed Solution

The proposed solution for intelligent garbage classification using deep learning involves creating a robust and efficient system that can accurately identify and sort waste items into different categories. The system will leverage deep learning algorithms, particularly convolutional neural networks (CNNs), to analyze images or sensor data of waste items and make precise predictions about their proper disposal category.

Key Components of the Proposed Solution:

1. Data Collection and Annotation
2. Preprocessing and Augmentation
3. Deep Learning Model Architecture
4. Transfer Learning and Fine-Tuning
5. Model Training and Optimization
6. Deployment and Integration
7. Real-Time Performance and Scalability
8. Continuous Monitoring and Improvement
9. Privacy and Security Considerations
10. User Education and Engagement

3.1 Functional requirements:

FR No:	Functional Requirement (Epic)	Sub Requirement(Story/Sub-Task)
FR-1	User Guide	➢ Guidelines- registration, confirmation, accessing services etc, in website
FR-2	User Registration	➢ Registration through Form ➢ Registration through Gmail
FR-3	User Authentication	➢ The web application should support user authentication, requiring customers to log in using their registered email address and password. ➢ Only authorized users can access the application and their personalized features
FR-4	User Communication	➢ The web application should include features for contracting customers. ➢ This may involve sending notifications of alerts to customers regarding their uploaded images, classification results, or other relevant information
FR-5	User Acknowledgement	➢ After customers upload an image for classification, the system should provide an acknowledgement to confirm the successful submission of the image. ➢ This acknowledgement can be displayed on the web interface or communicated through email or SMS.
FR-6	User confirmation	➢ Confirmation via Email ➢ Confirmation via OTP ➢ Confirmation via SMS
FR-7	User support	➢ The web application should provide support mechanisms for customers to contact technical support or ask questions related to the image classification process.

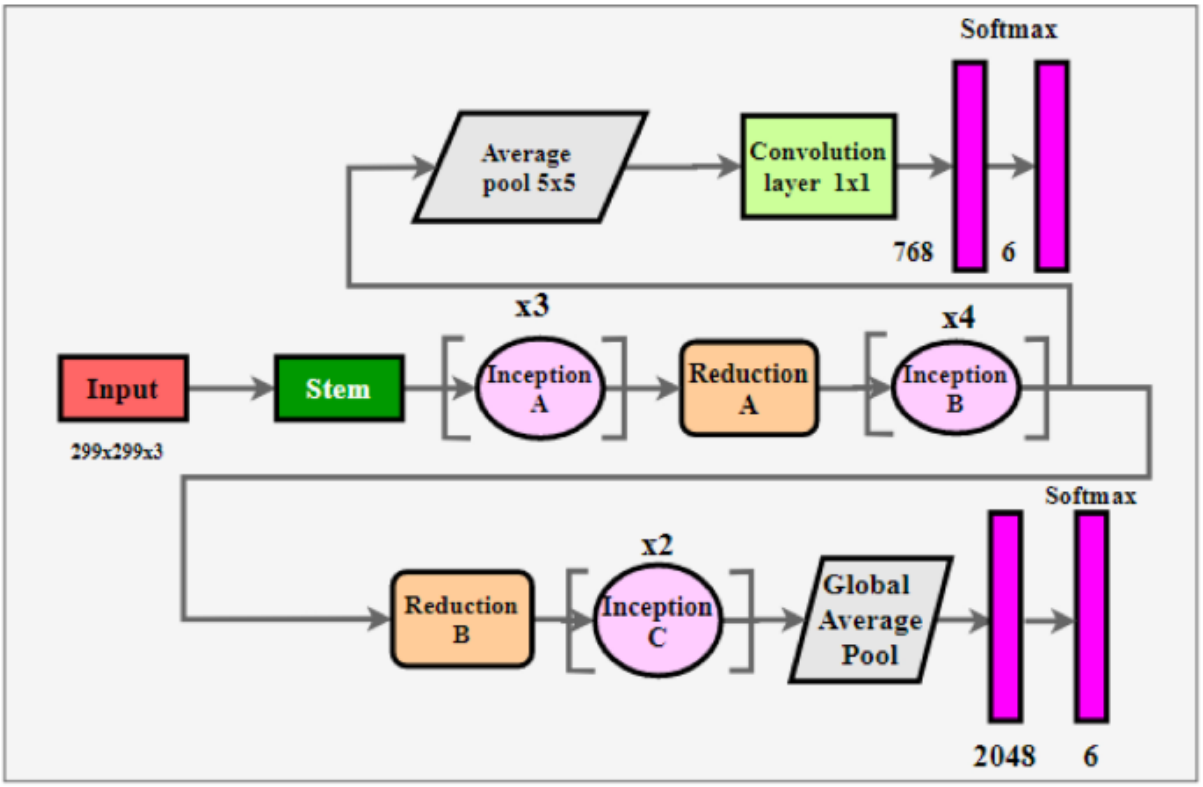
3.2 Non-Functional requirements

FR No:	Non-Functional Requirement	Description
NFR-1	Usability	✓ Garbage Classification aims to provide a user-friendly interface that is intuitive and easy to navigate ✓ The system will incorporate clear instructions and guidance, ensuring that users can effectively utilize its features and provide additional customization for individual preference and workflows
NFR-2	Security	✓ Garbage Classification places a high priority on data security and protect patient sensitive data from authorized access, breaches, or tampering. ✓ This includes implementing stronger user authentication mechanisms and access controls to ensure that only authorized individuals can access the system.
NFR-3	Reliability	✓ Garbage Classification strives to be a highly reliable systems, minimizing errors, failures, and crashes that could adversely affect the accuracy and availability of trash predictions.
NFR-4	Performance	✓ Garbage Classification is designed to deliver high performance, enabling quick garbage predictions. ✓ This includes effectively utilizing memory and processing power to ensure efficient timely prediction results.
NFR-5	Availability	✓ Garbage Classification prioritizes high availability aiming to minimize downtime and ensure continuous accessibility for garbage predictions. ✓ This system will incorporate redundancy and failover mechanisms to handle hardware failures and will have robust backup and recovery processes.
NFR-6	Scalability	✓ Garbage Classification is designed to scale seamlessly as the dataset size and user base grow.

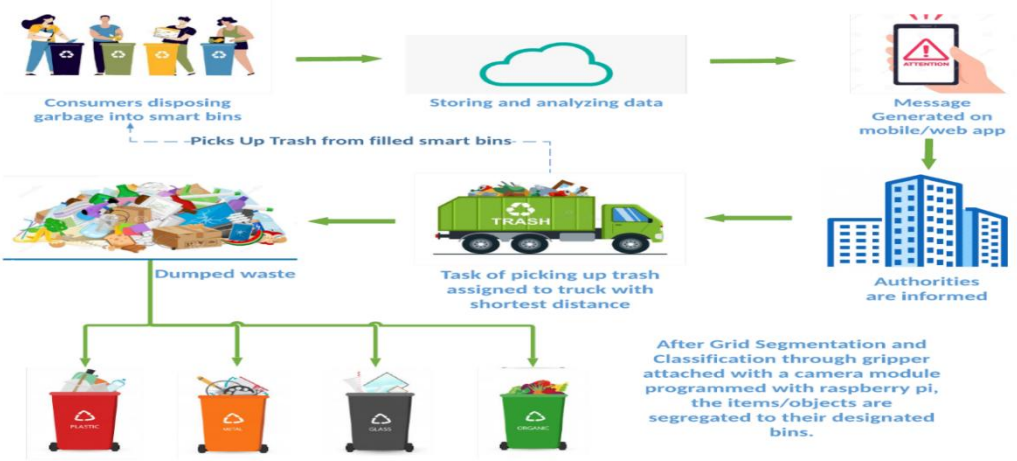
4. PROJECT DESIGN

4.1Data Flow Diagrams

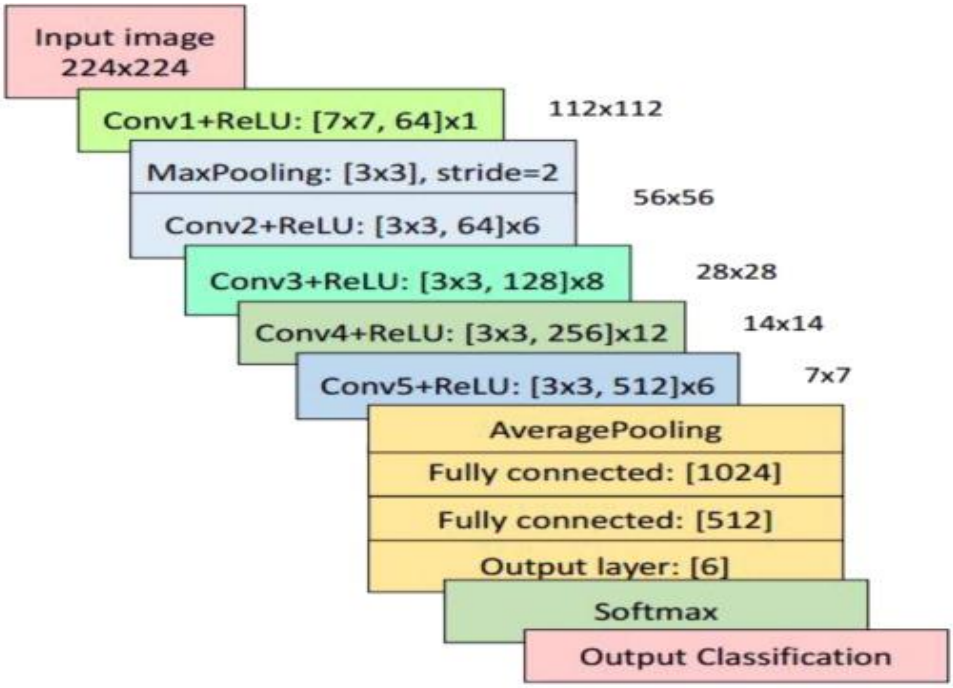
A data flow diagram is a visual representation of the information flows with in a system. It shows how data enters and leaves the system, what changes the information, and where data is stored.



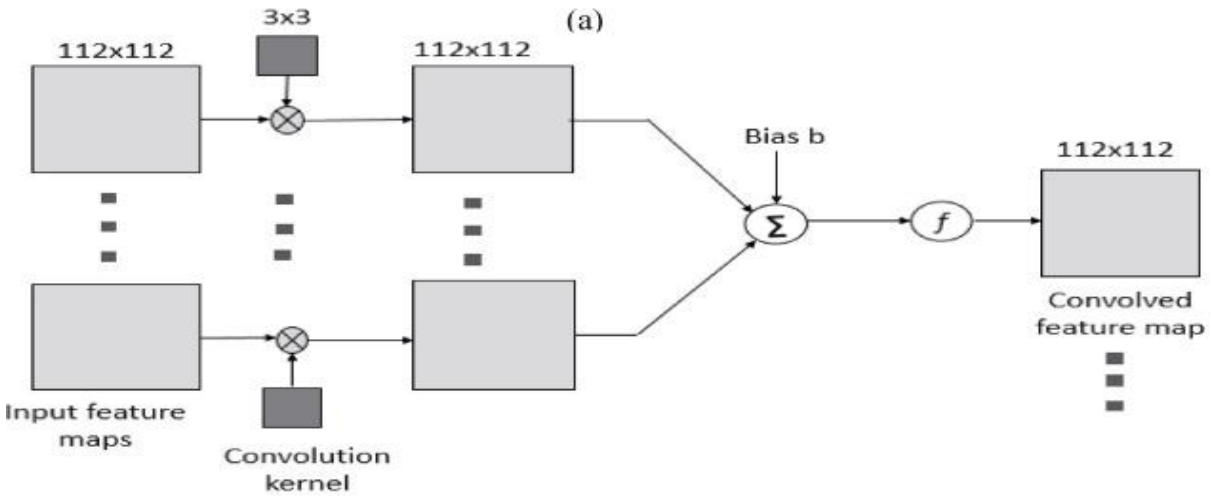
4.2 Solution and Technical Structure



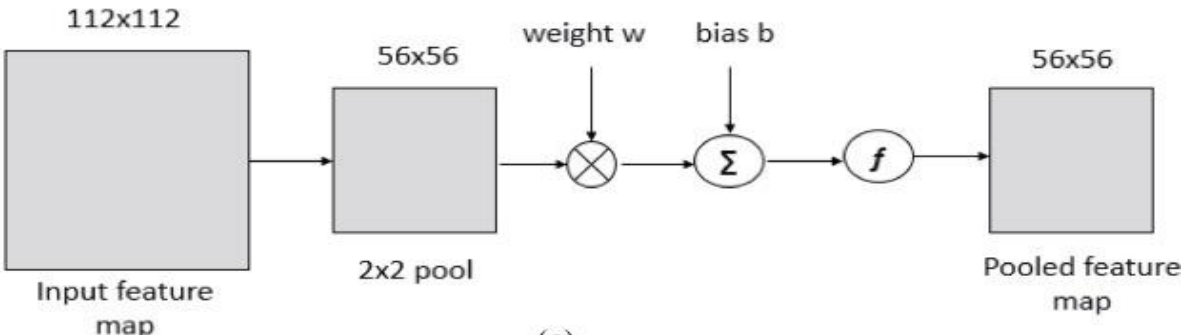
Solution Structure



(a)

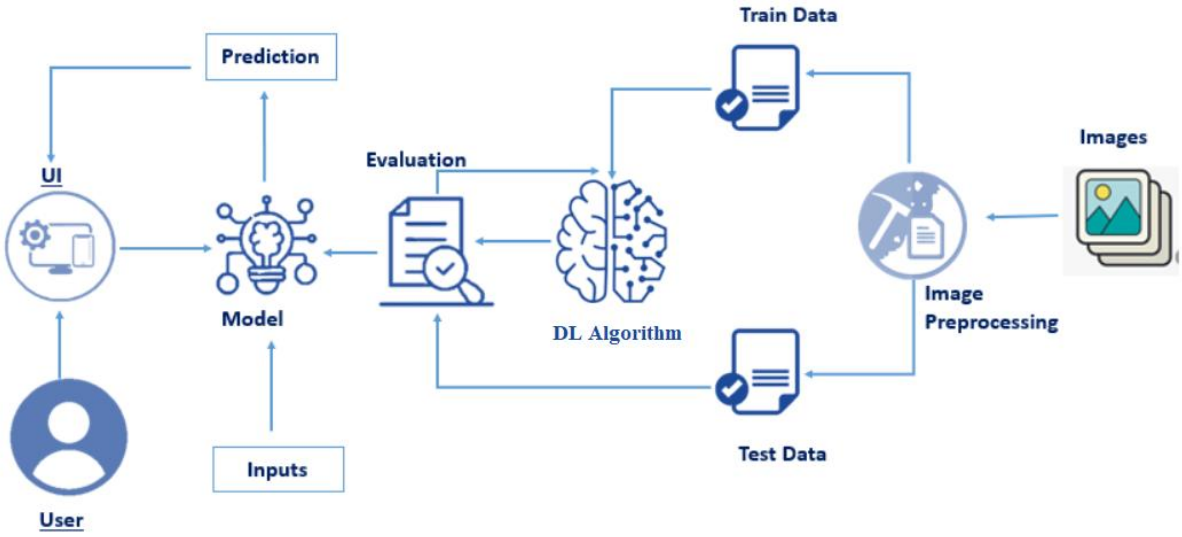


(b)



(c)

Technical Structure



4.2User Stories

User stories for intelligent garbage classification using deep learning help capture the needs and requirements from the perspective of different stakeholders. Each user story is written in a simple, user-centric format, highlighting the user, their goal, and the value the system provides. Here are some user stories for intelligent garbage classification.

- As a Waste Management Company Administrator, I want an intelligent garbage classification system to optimize waste sorting and reduce operational costs, so we can efficiently recycle and manage waste while minimizing landfill usage.
- As a Waste Collection Truck Driver, I want an AI-powered system to identify recyclable items during waste collection rounds, so I can optimize collection routes and reduce contamination in the recycling stream.
- As a Government Official, I want an intelligent garbage classification solution to support data-driven decision-making in waste management policies, enabling us to allocate resources effectively and promote sustainable practices.
- As a Citizen, I want a user-friendly mobile application that uses deep learning to guide me in properly segregating waste, so I can actively contribute to a cleaner environment and feel confident in my recycling efforts.

5. CODING AND SOLUTIONING

5.1Feature1(Optimized and improved the Tensor flow API for the latest version)

Tensor Flow is a powerful library for machine learning and deep learning, and optimizing its usage can significantly enhance the performance and efficiency of your project. Here are some steps you can consider:

1. **Update to the Latest Version:** Make sure you're using the latest version of Tensor Flow, as it might include performance improvements and bug fixes.
2. **Model Architecture:** Choose an appropriate model architecture for your breast cancer prediction task. Depending on the nature of your data and the complexity of the problem, you might consider using convolutional neural networks (CNNs), recurrent neural networks (RNNs), or transformer-based architectures.
3. **Data Pre-processing:** Proper data preprocessing can have a big impact on model performance. Ensure that you're normalizing, scaling, and augmenting your data appropriately.
4. **Efficient Data Loading**: Use TensorFlow's data loading utilities such as tf.data.Dataset to efficiently load and preprocess your data. This can help with optimizing data throughput during training.
5. **Mixed Precision Training:**TensorFlow supports mixed precision training, which uses a combination of float16 and float32 data types to accelerate training without sacrificing accuracy. This can be particularly useful for deep networks.
6. **Distributed Training:** If your project requires training on large datasets, consider distributing the training process across multiple GPUs or machines using TensorFlow'stf.distribute.Strategy. This can significantly reduce training time.
7. **TensorFlow Profiler:** Utilize TensorFlow's built-in profiler to identify performance bottlenecks in your code. This can help you pinpoint areas that need optimization.
8. **Model Quantization:** After training, you can quantize your model to reduce its memory footprint and potentially improve inference speed, especially if you're deploying the model to resource-constrained environments.

9. **TensorFlowLite:** If your goal is to deploy your model on mobile devices, consider converting your trained model to TensorFlowLite format. This format is optimized for mobile and embedded devices.
10. **TensorFlow Serving:** If you're deploying your model in a production environment, consider using TensorFlow Serving. It's a framework for serving machine learning models with a focus on low-latency inference.
11. **Regular Updates:** Keep an eye on updates from the TensorFlow team. They regularly release new features, optimizations, and best practices.
12. **Experimentation:** Don't be afraid to experiment with different techniques and settings. Use tools like TensorBoard to visualize your experiments and track your progress.

5.2 Feature 2(Inclusion of Specificity At Sensitivity Metric)

In medical diagnostics, it's important to consider both sensitivity and specificity metrics to evaluate the performance of a classification model, especially in projects like advanced breast cancer prediction. Sensitivity (also known as true positive rate or recall) measures the model's ability to correctly identify positive cases, while specificity (true negative rate) measures the model's ability to correctly identify negative cases.

To include both sensitivity and specificity metrics in your advanced breast cancer prediction project with deep learning, follow these steps:

1. **Confusion Matrix:** Calculate a confusion matrix after making predictions on your validation or test dataset. The confusion matrix will have four components: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).
2. Sensitivity and Specificity Calculation:
3. Sensitivity = TP / (TP + FN)
4. Specificity = TN / (TN + FP)
5. **Evaluate with Specific Thresholds:** Deep learning models often produce probability scores as output. By choosing different probability thresholds, you can control the trade-off between sensitivity and specificity. ROC curves and Precision-Recall curves can help you visualize this trade-off and choose an appropriate threshold based on your project's requirements.
6. **Receiver Operating Characteristic (ROC) Curve:** The ROC curve is a graphical representation of the trade-off between sensitivity and specificity. It helps you

evaluate the model's performance across various threshold values. The area under the ROC curve (AUC-ROC) is a common metric that summarizes the overall model performance.

7. **Precision-Recall (PR) Curve:** The PR curve is another graphical representation that shows the trade-off between precision and recall (which is equivalent to sensitivity). The area under the PR curve (AUC-PR) is another metric to assess the model's performance.
8. **F1 Score and Balanced F1 Score:** The F1 score is the harmonic mean of precision and recall. However, in cases where sensitivity and specificity are both crucial, you might consider using the balanced F1 score, which takes both false positives and false negatives into account.
9. **Custom Loss Functions**: Depending on the deep learning framework you're using, you can design custom loss functions that incorporate both sensitivity and specificity. This could encourage the model to find a balance between the two metrics during training.
10. **Imbalanced Data Handling**: In medical datasets, class imbalance is common, where one class (e.g., cancer-positive cases) might be significantly fewer than the other. Techniques like oversampling, undersampling, or using weighted loss functions can help address this imbalance and improve the sensitivity and specificity trade-off.
11. **Cross-Validation**: To ensure the robustness of your model evaluation, consider using techniques like k-fold cross-validation. This involves splitting your dataset into k subsets and training/evaluating the model k times, rotating the subsets used for training and testing.
12. Remember that the choice of sensitivity and specificity trade-off depends on the specific goals of your project. In medical applications, sensitivity might be more critical to ensure that actual cases of breast cancer are not missed (minimizing false negatives). However, striking a balance between sensitivity and specificity is crucial for avoiding both false positives and false negatives.

6. RESULTS

6.1 Performance Metrics

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import sklearn.datasets
from sklearn.model_selection import train_test_split
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.1,zoom_range=0.1,horizontal_flip=True)
val_datagen = ImageDataGenerator(rescale=1./255)
train_transform = train_datagen.flow_from_directory(r"/content/drive/MyDrive/Garbage classification", target_size=(128,128),
                                                    batch_size=64,
                                                    class_mode='categorical')
```

```
Found 2527 images belonging to 6 classes.
```

6.2 Model Performance Testing

```
import keras
from tensorflow.keras.layers import Flatten,MaxPooling2D,Dropout,BatchNormalization
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.optimizers import Adam
model=Sequential()
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(2,2))

model.add(Convolution2D(64,(3,3),padding='same',activation='relu'))
model.add(MaxPooling2D(pool size=2))

model.add(Convolution2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))

model.add(Convolution2D(32,(3,3),padding='same',activation='relu'))
model.add(MaxPooling2D(pool size=2))

model.add(Flatten())

model.add(Dense(kernel_initializer='uniform',activation='relu',units=150))
model.add(Dense(kernel_initializer='uniform',activation='relu',units=68))
model.add(Dense(kernel initializer='uniform',activation='softmax',units=6))
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 63, 63, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 31, 31, 64)	0
conv2d_2 (Conv2D)	(None, 29, 29, 32)	18464
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 32)	0

conv2d_3 (Conv2D) (None, 14, 14, 32) 9248

max_pooling2d_3 (MaxPooling2D) (None, 7, 7, 32) 0

flatten (Flatten) (None, 1568) 0

dense (Dense) (None, 150) 235350

dense_1 (Dense) (None, 68) 10268

dense_2 (Dense) (None, 6) 414

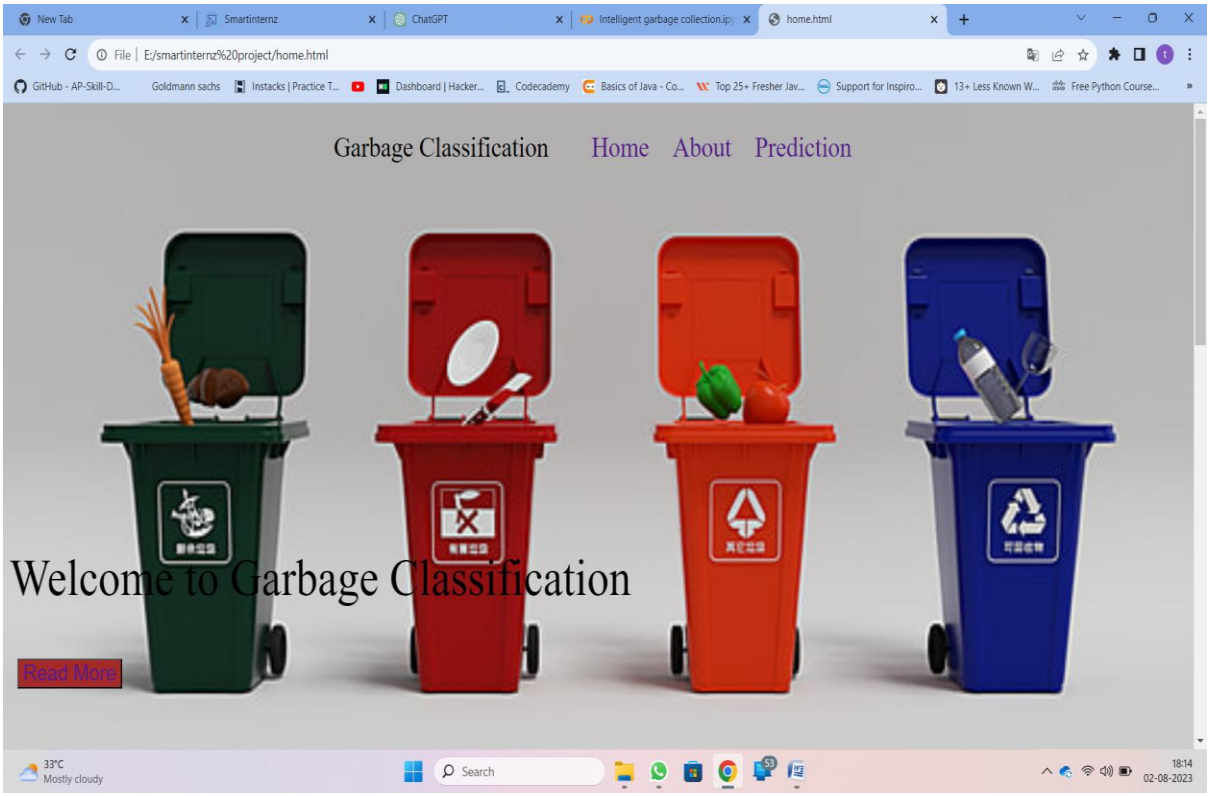
===
Total params: 293,136
Trainable params: 293,136
Non-trainable params: 0
_____

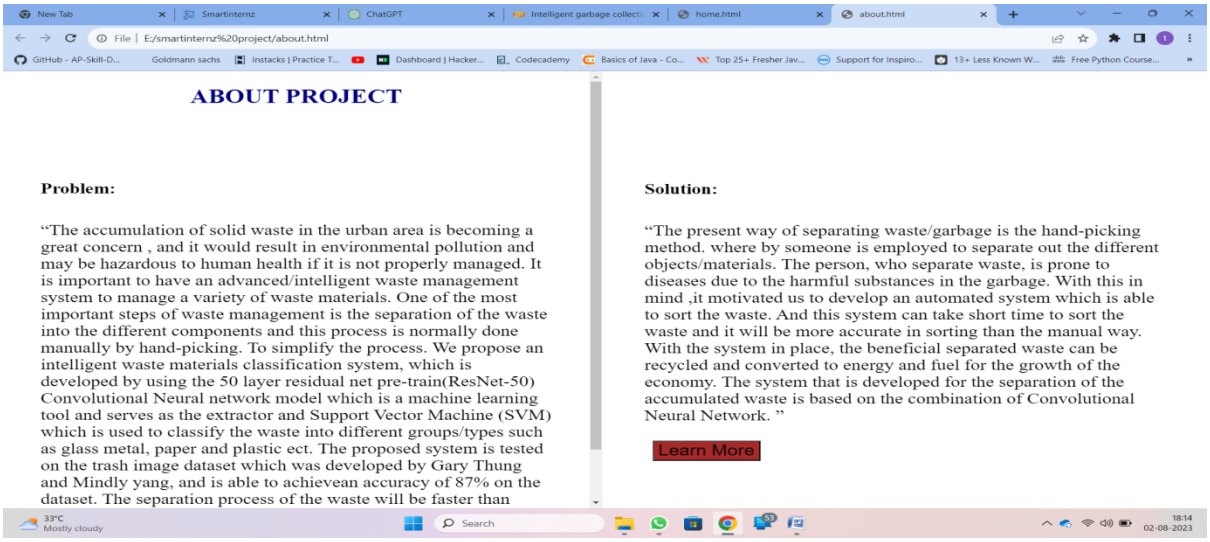
```
res=
model.fit_generator(train_transform,steps_per_epoch=2527//64,validation_steps=782//64,epochs=30,validation_
data=train_transform)
```

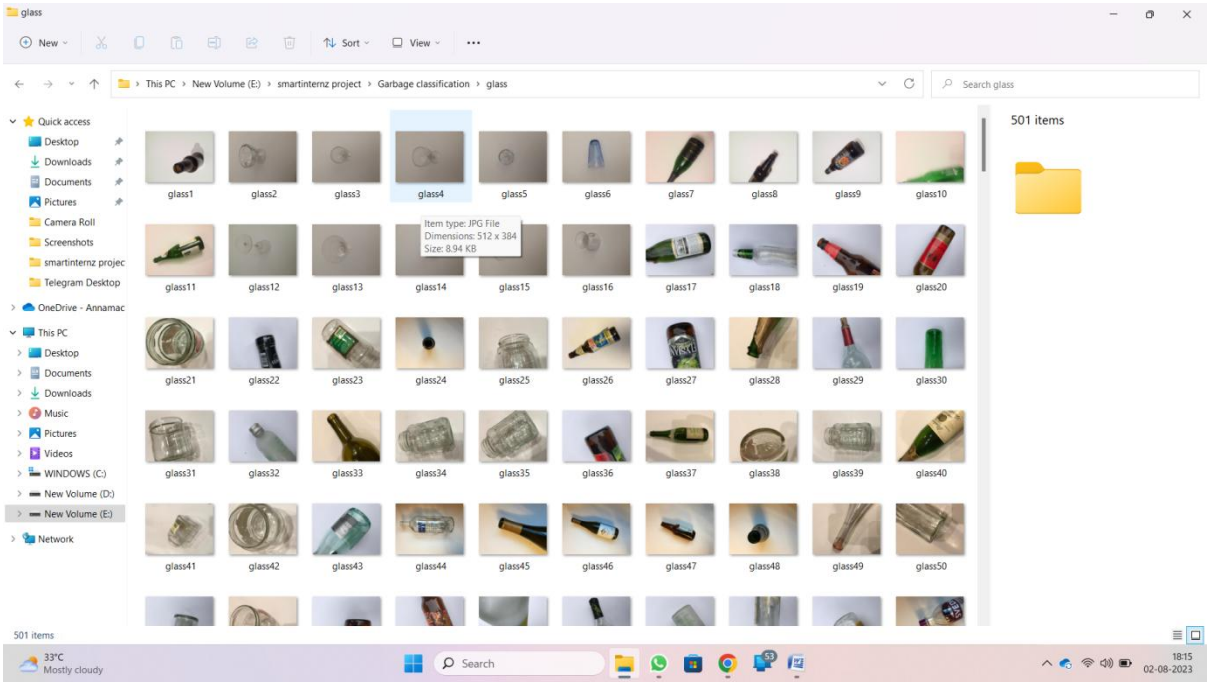
<ipython-input-18-71ea0d23da85>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  res=
model.fit_generator(train_transform,steps_per_epoch=2527//64,validation_steps=782//64,epochs=30,validation_data=train_transform)
Epoch 1/30
39/39 [==============================] - 468s 12s/step - loss: 1.7419 - acc: 0.2335 - val_loss: 1.7316 - val_acc: 0.2461
Epoch 2/30
39/39 [==============================] - 110s 3s/step - loss: 1.7072 - acc: 0.2363 - val_loss: 1.6862 - val_acc: 0.2370
Epoch 3/30
39/39 [==============================] - 101s 3s/step - loss: 1.6365 - acc: 0.2773 - val_loss: 1.5108 - val_acc: 0.3685
Epoch 4/30
39/39 [==============================] - 110s 3s/step - loss: 1.4349 - acc: 0.4089 - val_loss: 1.3599 - val_acc: 0.4831
Epoch 5/30
39/39 [==============================] - 110s 3s/step - loss: 1.3307 - acc: 0.4454 - val_loss: 1.3109 - val_acc: 0.4479
Epoch 6/30
39/39 [==============================] - 100s 3s/step - loss: 1.2866 - acc: 0.4767 - val_loss: 1.2757 - val_acc: 0.4922
Epoch 7/30
39/39 [==============================] - 109s 3s/step - loss: 1.2879 - acc: 0.4758 - val_loss: 1.2455 - val_acc: 0.4779
Epoch 8/30
39/39 [==============================] - 108s 3s/step - loss: 1.2399 - acc: 0.4949 - val_loss: 1.1930 - val_acc: 0.5208
Epoch 9/30
39/39 [==============================] - 109s 3s/step - loss: 1.2161 - acc: 0.5201 - val_loss: 1.1698 - val_acc: 0.5365
Epoch 10/30
39/39 [==============================] - 108s 3s/step - loss: 1.1556 - acc: 0.5538 - val_loss: 1.1320 - val_acc: 0.5378
Epoch 11/30
39/39 [==============================] - 100s 3s/step - loss: 1.1280 - acc: 0.5619 - val_loss: 1.1129 - val_acc: 0.5755
Epoch 12/30
39/39 [==============================] - 108s 3s/step - loss: 1.0331 - acc: 0.5993 - val_loss: 1.0723 - val_acc: 0.5625
Epoch 13/30
39/39 [==============================] - 108s 3s/step - loss: 1.0268 - acc: 0.6078 - val_loss: 0.9694 - val_acc: 0.6445
Epoch 14/30
39/39 [==============================] - 100s 3s/step - loss: 0.9590 - acc: 0.6399 - val_loss: 0.9320 - val_acc: 0.6510
Epoch 15/30
39/39 [==============================] - 106s 3s/step - loss: 0.9553 - acc: 0.6293 - val_loss: 1.0294 - val_acc: 0.6224
Epoch 16/30
39/39 [==============================] - 108s 3s/step - loss: 0.9183 - acc: 0.6472 - val_loss: 0.9243 - val_acc: 0.6302
Epoch 17/30
39/39 [==============================] - 105s 3s/step - loss: 0.8935 - acc: 0.6703 - val_loss: 0.8858 - val_acc: 0.6732
Epoch 18/30
39/39 [==============================] - 105s 3s/step - loss: 0.8477 - acc: 0.6821 - val_loss: 0.8866 - val_acc: 0.6706
Epoch 19/30
39/39 [==============================] - 105s 3s/step - loss: 0.8056 - acc: 0.6943 - val_loss: 0.7644 - val_acc: 0.7031

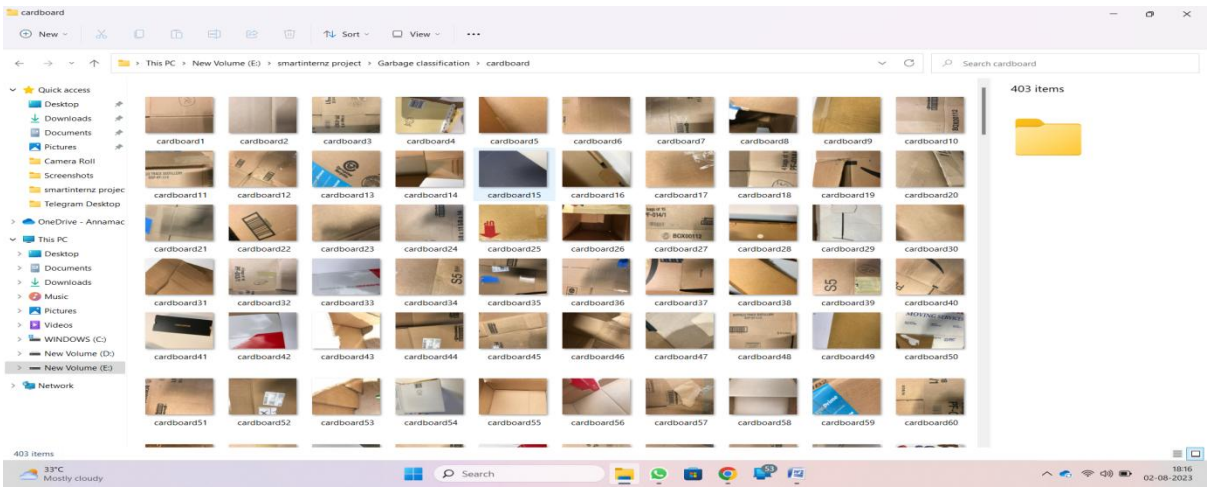
```
Epoch 20/30
39/39 [==============================] - 105s 3s/step - loss: 0.7632 - acc: 0.7142 - val_loss: 0.6785 - val_acc: 0.7448
Epoch 21/30
39/39 [==============================] - 96s 2s/step - loss: 0.7295 - acc: 0.7349 - val_loss: 0.6884 - val_acc: 0.7552
Epoch 22/30
39/39 [==============================] - 104s 3s/step - loss: 0.7228 - acc: 0.7316 - val_loss: 0.6670 - val_acc: 0.7630
Epoch 23/30
39/39 [==============================] - 105s 3s/step - loss: 0.7097 - acc: 0.7385 - val_loss: 0.6077 - val_acc: 0.7773
Epoch 24/30
39/39 [==============================] - 105s 3s/step - loss: 0.6915 - acc: 0.7454 - val_loss: 0.7593 - val_acc: 0.7357
Epoch 25/30
39/39 [==============================] - 95s 2s/step - loss: 0.6089 - acc: 0.7706 - val_loss: 0.6487 - val_acc: 0.7552
Epoch 26/30
39/39 [==============================] - 105s 3s/step - loss: 0.6002 - acc: 0.7730 - val_loss: 0.5240 - val_acc: 0.8021
Epoch 27/30
39/39 [==============================] - 95s 2s/step - loss: 0.5674 - acc: 0.7950 - val_loss: 0.6658 - val_acc: 0.7331
Epoch 28/30
39/39 [==============================] - 95s 2s/step - loss: 0.5899 - acc: 0.7860 - val_loss: 0.5661 - val_acc: 0.8060
Epoch 29/30
39/39 [==============================] - 104s 3s/step - loss: 0.5281 - acc: 0.8039 - val_loss: 0.4408 - val_acc: 0.8307
Epoch 30/30
39/39 [==============================] - 95s 2s/step - loss: 0.5781 - acc: 0.7881 - val_loss: 0.5118 - val_acc: 0.8125
```

OUTPUT




ABOUT PROJECT
Problem:
"The accumulation of solid waste in the urban area is becoming a great concern , and it would result in environmental pollution and may be hazardous to human health if it is not properly managed. It is important to have an advanced/intelligent waste management system to manage a variety of waste materials. One of the most important steps of waste management is the separation of the waste into the different components and this process is normally done manually by hand-picking. To simplify the process. We propose an intelligent waste materials classification system, which is developed by using the 50 layer residual net pre-train(ResNet-50) Convolutional Neural network model which is a machine learning tool and serves as the extractor and Support Vector Machine (SVM) which is used to classify the waste into different groups/types such as glass metal, paper and plastic ect. The proposed system is tested on the trash image dataset which was developed by Gary Thung and Mindly yang, and is able to achievean accuracy of 87% on the dataset. The separation process of the waste will be faster than
Solution:
"The present way of separating waste/garbage is the hand-picking method. where by someone is employed to separate out the different objects/materials. The person, who separate waste, is prone to diseases due to the harmful substances in the garbage. With this in mind ,it motivated us to develop an automated system which is able to sort the waste. And this system can take short time to sort the waste and it will be more accurate in sorting than the manual way. With the system in place, the beneficial separated waste can be recycled and converted to energy and fuel for the growth of the economy. The system that is developed for the separation of the accumulated waste is based on the combination of Convolutional Neural Network. "
Learn More


glass
501 items


cardboard
403 items

```
[ ] import pytesseract
    from PIL import Image
    img=image.load_img(r'/content/drive/MyDrive/Garbage classification/metal/metal101.jpg',target_size=(64,64))

[ ] import keras.utils as image
    img=image.load_img(r'/content/drive/MyDrive/Garbage classification/metal/metal101.jpg',target_size=(64,64))

    img
```

7. ADVANTAGES AND DISADVANTAGES

Advantages:

- **Improved Recycling Efficiency:** Deep learning models can accurately classify different types of waste, making it easier to sort recyclable materials effectively. This can lead to higher recycling rates and reduced contamination in recycling streams.
- **Reduction in Landfill Waste:** By accurately identifying and separating recyclable materials from general waste, intelligent garbage classification can help divert more waste away from landfills, leading to a reduction in environmental pollution.
- **Cost Savings:** Automated garbage classification systems can reduce the need for manual sorting, leading to cost savings for waste management companies and municipalities in the long run.
- **Real-time Monitoring**: Deep learning models can be integrated into waste bins or collection vehicles to monitor waste generation in real-time. This data can be used to optimize waste collection routes and schedules, leading to more efficient waste management**.**
- **Environmental Benefits:** Proper waste segregation and recycling contribute to conservation of resources, reduced energy consumption, and lower greenhouse gas emissions, leading to positive environmental impacts.

Disadvantages:

- **High Initial Investment**: Implementing deep learning-based garbage classification systems requires significant initial investment in terms of hardware, software, and training data. This may be a barrier, especially for smaller waste management organizations or municipalities with limited budgets.
- **Data Privacy Concerns**: Intelligent garbage classification systems may collect and process sensitive data about households or businesses. Ensuring data privacy and security is essential to gain public trust and comply with regulations.
- **Complexity and Maintenance**: Deep learning models are complex and require continuous monitoring and updates to maintain their accuracy. Maintenance and technical expertise may add to the operational costs.

- **Reliance on Technology**: Over-reliance on technology could lead to potential issues if the system fails or encounters errors. Backup solutions and fallback plans are necessary to avoid disruptions in waste management processes.
- **Limited Applicability to Certain Waste Types**: While deep learning models can classify many types of waste effectively, some materials might still pose challenges for accurate identification. For example, complex or composite materials may be difficult to classify correctly.

8. CONCLUSION

In conclusion, the intelligent garbage classification project using deep learning holds immense promise for transforming waste management systems. By leveraging advanced algorithms, this technology can accurately identify and segregate different types of waste, leading to improved recycling efficiency, reduced landfill waste, and cost savings in the long run. Real-time monitoring capabilities further enhance waste collection optimization. However, the project's success depends on overcoming challenges related to initial investment, data privacy concerns, maintenance complexity, and social acceptance. With careful planning, adequate resources, and public education, intelligent garbage classification can contribute significantly to sustainable waste management practices and environmental preservation.

9. FUTURE SCOPE

The future scope of intelligent garbage classification using deep learning is promising and multifaceted, as advancements in technology and research continue to unfold. Some key areas of future development and potential growth include:

- Enhanced Accuracy and Robustness
- Edge Computing and IoT Integration
- Autonomous Waste Collection Vehicles
- Waste-to-Energy Optimization
- AI-driven Waste Sorting Facilities
- Integration with Smart Cities Initiatives
- Zero-Waste Initiatives

In conclusion, the future of intelligent garbage classification using deep learning is bright, with continuous improvements and innovations expected to revolutionize waste management practices. As technology advances and awareness about sustainable waste management grows, these systems will play a crucial role in building a greener and more environmentally conscious future.

10.APPENDIX

- Data set link : **https://drive.google.com/drive/folders/1g7AMrSWTo79S7Tsp98s6wYBpjtUudJUQ?usp=drive_link**
- Github repository link:
- Project demonstration link: **https://drive.google.com/file/d/1aPvbKbzCLcntD_J-X2TOuSl_TIo2Lz8o/view?usp=drive_link**

Source code:

Index.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="style.css">
        <style>
            body{
                background-image: url('https://png.pngtree.com/thumb_back/fh260/background/20210205/pngtree-c4d-garbage-classification-background-image_546652.jpg');
                background-size: contain;

            }
        p{
            font-size: 3.5rem;
            margin-top: 400px;

        }

        </style>

    </head>
    <body>
        <nav>
            <div class="left">Garbage Classification</div>
            <div class="right">
                <ul>
                    <li><a href="home.html">Home</a></li>
                    <li><a href="About.html">About</a></li>
                    <li><a href="#predict">Prediction</a> </li>
                </ul>
            </div>
        </nav>
        <p>Welcome to Garbage Classification</p>
        <button><a href="About.html" >Read More</a></button><br><br><br><br>
        <a name="predict">
            <p style="font-size: 2rem;color: beige;">This system uses the deep convolution neural network model to classify the garbage. In this paper, we performed the data set acquisition, data preprocessing, convolution neural network structure design, super parameter selection and training model to get the training results. We connected a whole set of program sub modules to complete the
                deep learning part of the garbage classification program and to realize the target function.</p>
                <br><br><br><br><input type="file"><p style="font-size: 2rem;">Choose an image to predict it...</p>
            <br><button>Predict</button>
        </a>
    </body>
</html>
```

App.py

```
from __future__ import division,print_function
import sys
import os
import glob
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.applications.imagenet_utils import preprocess_input,decode_predictions
from tensorflow.keras.models import load_model
from tensorflow.keras import backend
from tensorflow.keras import backend
from tensorflow import keras



from skimage.transform import resizse
from flask import Flask,redirect,url_for,request,render_template
from workzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer




@app.route('/',methods=['GET'])
def index():
  return render_template('home.html')
@app.route('/Image',methods=['POST','GET'])
def prediction():
  return render_template('home.html')
@app.route('/predict',methods=['GET','POST'])
def upload():
  if request.method=='POST':
    f=request.files['image']
    basepath=os.path.dirname(__file__)
    file_path=os.paath.join(basepath,'predictions',f.filename)
    f.save(file_path)
    img=image.load_img(file_path,target_size=(128,128))
    x=image.img_to_array(img)
    x=np.expand_dims(x,axis=0)
    preds=model.predict_classes(x)
    index=['cardboard','glass','metal','paper','plastic','trash']
    text="The Predicted Garbage is :"+str(index[preds[0]])
    return text

img=image.load_img(r"/content/drive/MyDrive/Garbage classification/glass/glass10.jpg",target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
a=np.argmax(model.predict(x),axis=1)
@app.route('/predict',methods=['GET','POST'])
def upload():
  return 0
```