PROJECT AND TERM PAPER

SOLID WASTE MANAGEMENT IN SMART CITIES

NAME – Shourya Bhattacharyya

ROLL – 002111701070

YEAR – 3rd

SEMESTER – 1st

DEPARTMENT OF PRODUCTION ENGINEERING

Acknowledgement

The success and final outcome of this report on term paper required a lot of guidance and assistance and I feel extremely privileged to have received this immense amount of support from the conception of this term paper up until its successful completion. All I have done is only due to such supervision and assistance and through this section I wish to express my outmost gratitude to everyone involved.

I respect and thank **Prof. Dr Sanjib Kumar Saren**, for providing me an opportunity to do the term paper work in the field of **"Solid Waste Management in Smart Cities"** and giving me all support and guidance, which helped me to complete the project in time. I am extremely thankful to him for taking time out of his busy schedule to assist me.

I am also grateful to the rest of the faculty and staff in the Department of Production Engineering, for providing me the necessary resources, opportunities and support.

Finally, I would like to express my heartfelt thanks to my family and friends, for their constant support, understanding and encouragement throughout the time it took to complete the term paper.

Shourya Bhattacharyya

B. Prod.E - III

CONTENTS

1. Machine Learning

Machine learning (ML) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks. It is seen as a part of artificial intelligence.

Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, agriculture, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A core objective of a learner is to generalize from its experience.[5][31] Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory via the Probably Approximately Correct Learning (PAC) model. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

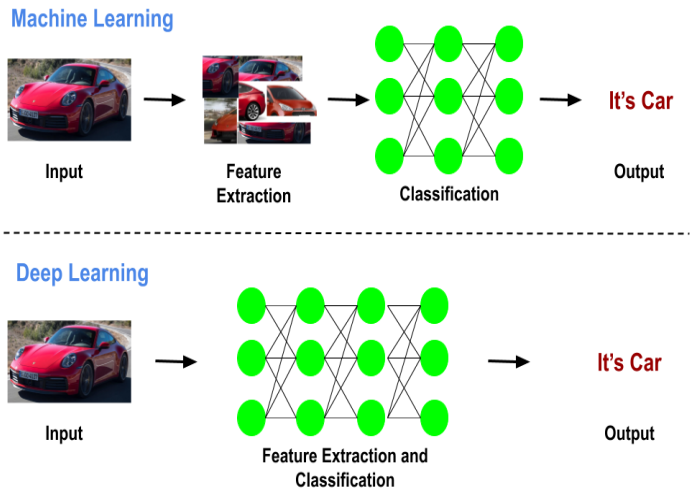
2. Deep Learning

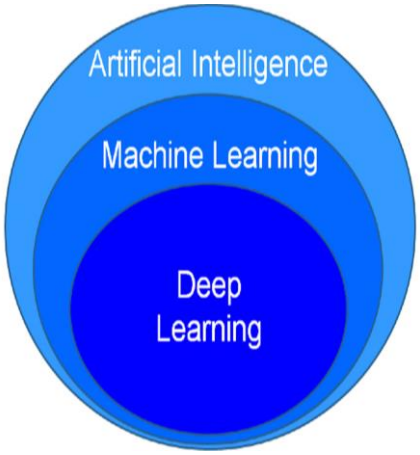
Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks, convolutional neural networks and Transformers have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, artificial neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analog.

3. Deep Learning vs Machine Learning





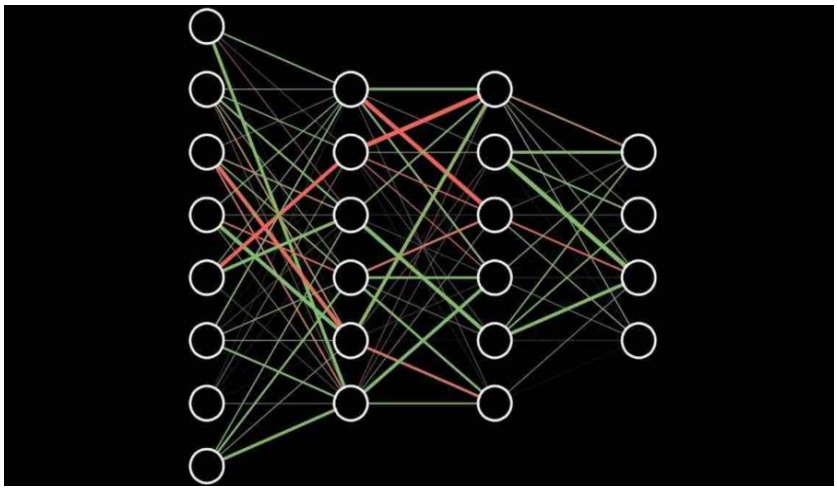
3.1 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is adapted for a second related task. In the context of deep learning, transfer learning involves using a pre-trained neural network on a large dataset for a specific task and then fine-tuning or using parts of that trained network on a different but related task. This approach leverages the knowledge gained from the source task to improve learning on the target task.

Some advantages are as follows:

1. **Reduced Training Time and Resources**: Pre-training a deep neural network on a large dataset for a source task is computationally expensive. Transfer learning allows you to reuse the pre-trained model's weights and architecture for a related target task, significantly reducing the amount of training time and computational resources required.
2. **Improved Generalization**: Transfer learning helps models generalize better to new tasks, especially when the source and target tasks share common features or patterns. The knowledge gained during pre-training on a large dataset helps the model capture generic features that can be useful across different tasks.
3. **Effective in Low Data Scenarios**: In scenarios where the target task has a limited amount of labeled data, transfer learning can be particularly beneficial. The pre-trained model brings knowledge from the source task, which can compensate for the scarcity of data in the target task.
4. **Avoidance of Overfitting**: Transfer learning can help mitigate overfitting, especially when the target task has a small dataset. The pre-trained model has already learned generic features from a large dataset, reducing the risk of overfitting to the limited data available for the target task.

4. What is a Neural Network?



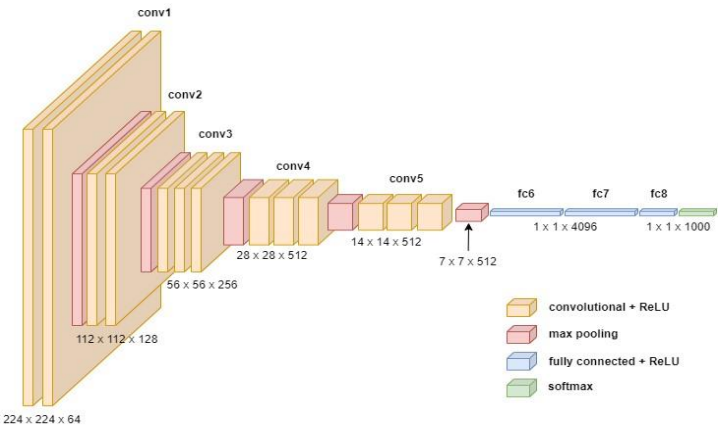
Neural nets are a means of doing machine learning, in which a computer learns to perform some task by analysing training examples.

Modelled loosely on the human brain, a neural net consists of thousands or even millions of simple processing nodes that are densely interconnected. Most of today's neural nets are organized into layers of nodes, and they're "feed-forward," meaning that data moves through them in only one direction. An individual node might be connected to several nodes in the layer beneath it, from which it receives data, and several nodes in the layer above it, to which it sends data.

To each of its incoming connections, a node will assign a number known as a "weight." When the network is active, the node receives a different data item — a different number — over each of its connections and multiplies it by the associated weight. It then adds the resulting products together, yielding a single number. If that number is below a threshold value, the node passes no data to the next layer. If the number exceeds the threshold value, the node "fires," which in today's neural nets generally means sending the number — the sum of the weighted inputs — along all its outgoing connections. This process of assigning the weights to each connection is called Training of the model.

4.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN or ConvNet) is a type of artificial neural network designed specifically for processing structured grid data, such as images. CNNs have proven highly effective in computer vision tasks, including image classification, object detection, and image recognition. The architecture of CNNs is inspired by the visual processing that occurs in the human brain.



Here are the key components and concepts of a Convolutional Neural Network:

1. **Convolutional Layers**: The core building block of a CNN is the convolutional layer. Convolution involves the use of filters (also called kernels) to scan across the input data (usually an image) to detect features such as edges, textures, or patterns.

2. **Stride and Padding**: Convolution is performed with a certain "stride," which determines the step size of the filter as it moves across the input. Stride affects the output size. Padding is often added to the input to preserve spatial information and ensure that the convolutional layers do not excessively reduce the size of the input.
3. **Activation Function**: Convolutional layers are typically followed by activation functions, with Rectified Linear Unit (ReLU) being a common choice. ReLU introduces non-linearity to the network, allowing it to learn complex patterns.
4. **Pooling (Subsampling)** Layers**:** Pooling layers are used to downsample the spatial dimensions of the input volume. In max pooling the maximum value in a region of the input is taken, reducing the dimensionality while preserving the most important information. Pooling helps in creating a more abstract and compressed representation of the input, reducing computation in deeper layers.
5. **Fully Connected Layers**: After several convolutional and pooling layers, the high-level reasoning in the neural network is often done through one or more fully connected layers. These layers connect every neuron to every neuron in the previous and subsequent layers.
6. **Flattening**: Before feeding the output of the convolutional and pooling layers into the fully connected layers, the data is usually flattened. This involves reshaping the multi-dimensional output into a vector.
7. **Output Layer**: The last layer of the CNN is the output layer, which produces the final predictions. The number of nodes in this layer corresponds to the number of classes in a classification task, and softmax activation is commonly used for multi-class classification.

5. Training of a Neural Network or Model

When a neural net is being trained, all of its weights and thresholds are initially set to random values. Training data is fed to the bottom layer — the input layer — and it passes through the succeeding layers, getting multiplied and added together in complex ways, until it finally arrives, radically transformed, at the output layer. During training, the weights and thresholds are continually adjusted until training data with the same labels consistently yield similar outputs.

The procedure used to carry out the learning process in a neural network is called the optimization algorithm (or optimizer).

There are many different optimization algorithms. All have different characteristics and performance in terms of memory requirements, processing speed, and numerical precision.

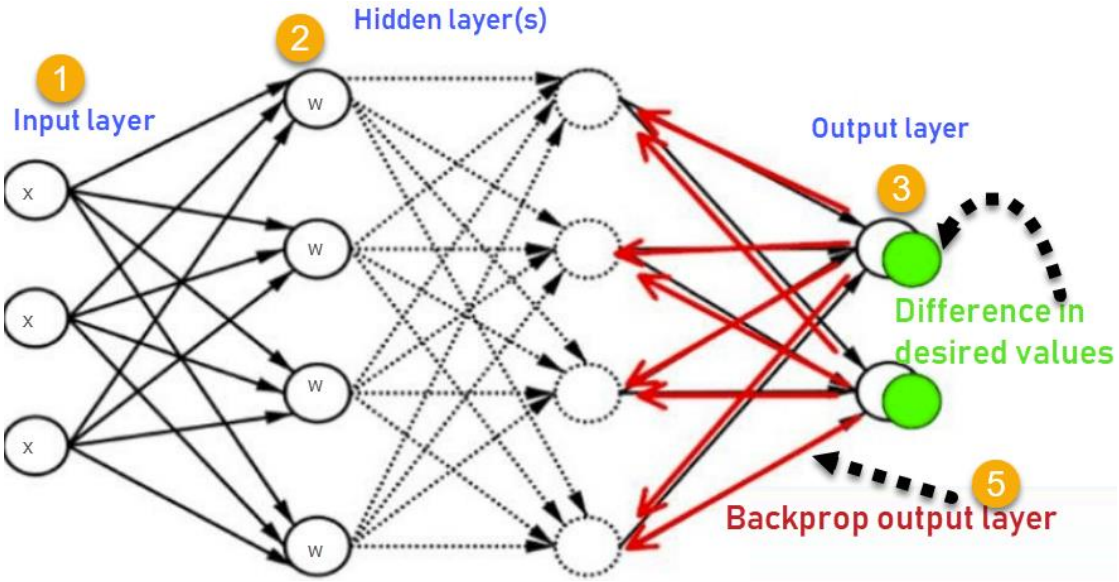
In this post, we formulate the learning problem for neural networks. Then, some important optimization algorithms are described. Finally, the memory, speed, and precision of those algorithms are compared. Some of these Algorithms are described below.

5.1 Backward Error Propagation

Backpropagation, short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using Gradient Descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights. It is a generalization of the delta rule for perceptrons to multilayer feedforward neural networks.

The "backwards" part of the name stems from the fact that calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately.

Backpropagation's popularity has experienced a recent resurgence given the widespread adoption of deep neural networks for image recognition and speech recognition. It is considered an efficient algorithm, and modern implementations take advantage of specialized GPUs to further improve performance.



6 Waste Management Model in Smart Cities

6.1 Introduction

The waste management system consists of disposal and treatment of different types of waste. Effective waste management technique can save much money which will lead to improve air quality and less environmental pollution.

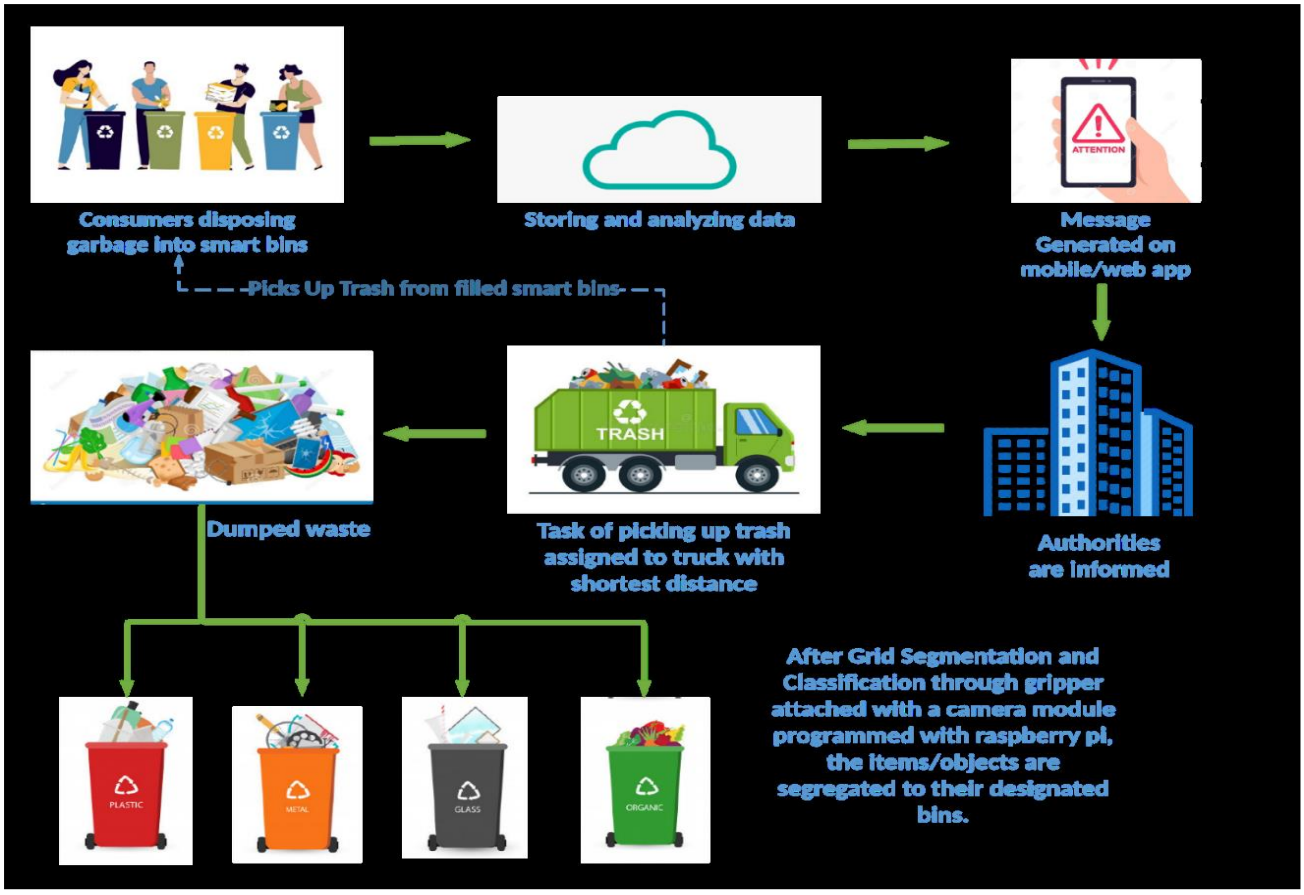
Due to rising population worldwide, there is an enormous increase in waste generation per day. Approximately 1.9 billion tons of waste is generated annually, with a minimum of 35% that is not treated securely. Income and generation of waste is directly proportional to each other.

Waste is a huge income source, so its treatment and disposal must be done in best possible way. The most effective technology to overcome the problems of environmental pollution is the use of the Internet of Things (IoT) and Machine learning (ML) based waste management system. These technologies can provide real time information about the waste and provide an optimized path for the waste collection trucks, reducing the cost and time for the overall process. The issues faced by current waste management systems are improper scheduling that is the waste collectors don't know that they have to pick the waste and they also don't know precisely about the drop off location.

The waste management system proposed in this paper will detect the waste level in the dustbins and the dumping grounds. Arduino UNO micro-controller is used in this system. Sensors are connected with the microcontroller for processing. Here we have deployed ultrasonic sensor and moisture sensor. The ultrasonic sensor will depict the assorted distance from waste in the dustbins, while the waste is dry or wet is detected by moisture sensor.

First the house-hold waste is collected in our smart waste bins, whose data is stored on the cloud and a message on the web/mobile application is generated when the bin gets full. A dumper truck database has been generated in the given system so that data and details of dumper truck ID, meeting date, meeting time of garbage collection and so on are collected. A mobile application will monitor the real time movement of the vehicles and will send the most optimised track to collect the waste from the waste bins and take it to the dumping area where segregation and classification of waste are performed.

For waste segmentation we use a grid segmentation mechanism that makes segments of the waste. Then a gripper with a camera connected with raspberry Pi starts picking waste items. After performing classification, it places the specific items in designated bins. In this model the waste is classified into six classes namely paper, metal, glass, trash.

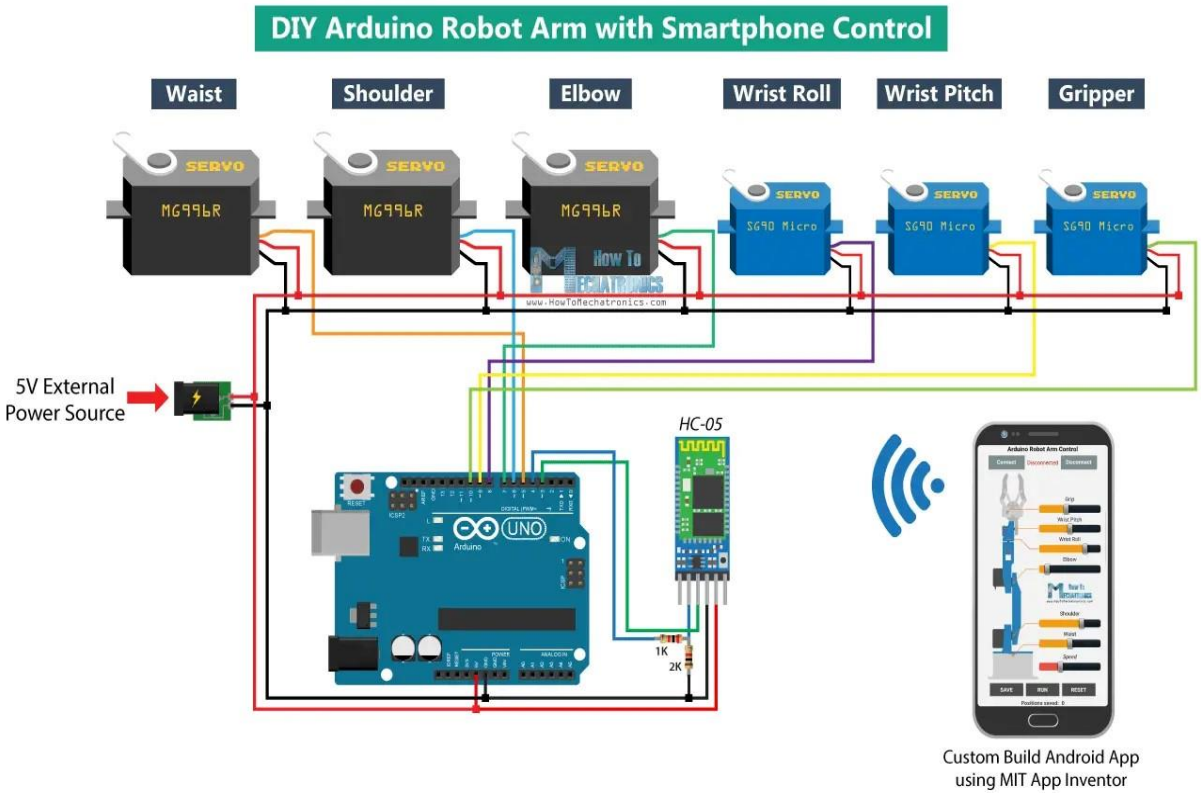


6.2 Experimental Setup

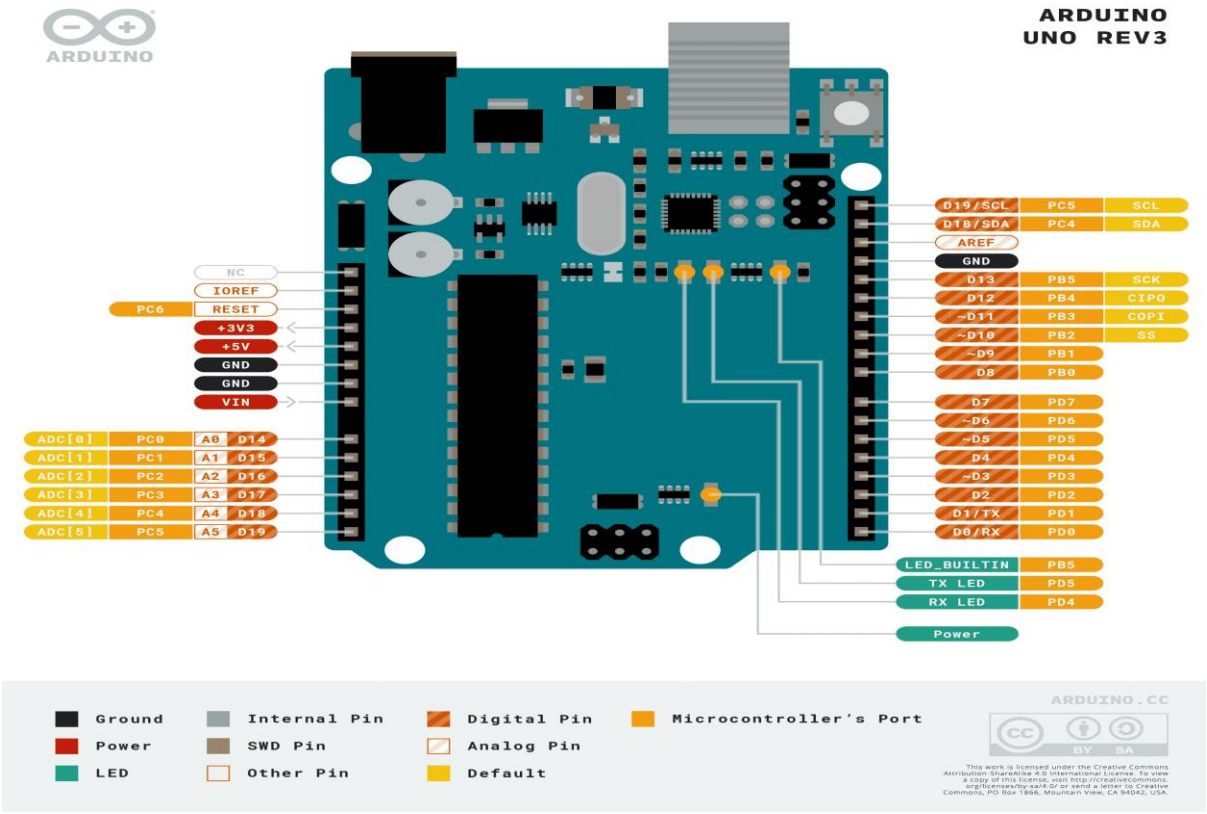
Specification of Each Component

Module Name	Hardware Requirements	Component Specs
Edge Node	Smartphone Pi-CAM	8GB (RAM) 5MP
Cloud Processing	Artificial Intelligence Module Pre-trained Deep learning Model	-- --
Power Source	DC Battery	6V, 4.5AH
Control Unit	Micro-controller Bluetooth Module Gripper Servo Motor	Arduino Mega 2560 HC 05 (Slave Mode) 5 Degree of Freedom MG996R, SG90 micro

The components are interconnected and embedded in a single module to make it a smart waste classification system. The waste items are picked from the dump and dropped in a specified bin by a 5-DOF gripper, which operates on the inputs received from phone. Arduino controls the gripper's movement and the movement of all motors associated with it, and the edge node (Smartphone) performs all the computation and classification based on the pie cam input image in a real-time environment. The figure below represents the visual assembly of the proposed system components. As for the experimental setup, pie cam mounted with edge node hanging at the distance of 45 cm. The overall waste dump area is 4*50 cm2, and the gripper is hanging in the middle of the mechanical assembly at a distance of 60 cm from the surface. It can move up and down through the steel string folded on the moveable pulley attached to the stepper motor. The waste dump is inside the mechanical model on the surface. It is picked after the grid segmentation technique and classification output



6.2.1 Arduino Uno



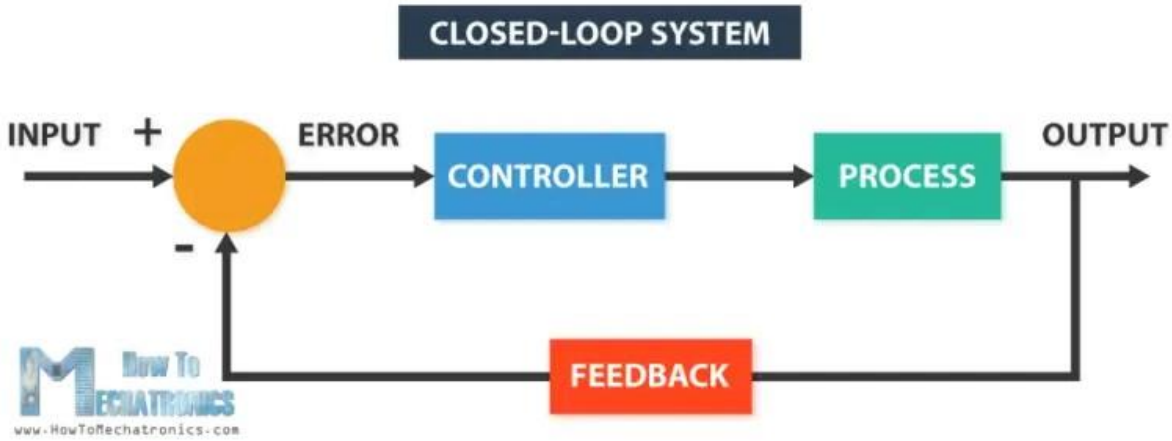
The ArduinoUno is an open source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available. The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software.

The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes pre-programmed with a bootloader that allows uploading new code to it without the use of an external hardware programmer.

While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

6.2.2 Servo-Motor

A servo motor is a closed-loop system that uses position feedback to control its motion and final position. There are many types of servo motors and their main feature is the ability to precisely control the position of their shaft.

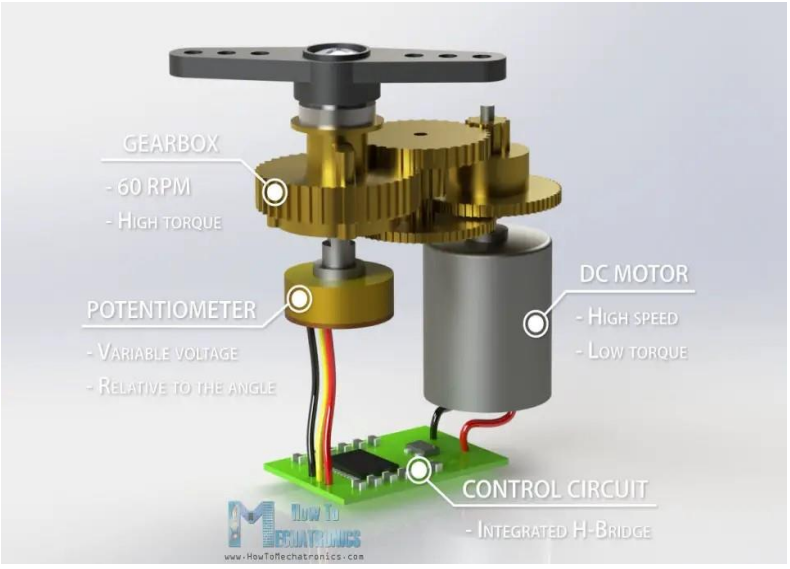


In industrial type servo motors the position feedback sensor is usually a high precision encoder, while in the smaller RC or hobby servos the position sensor is usually a simple potentiometer. The actual position captured by these devices is fed back to the error detector where it is compared to the target position. Then according to the error the controller corrects the actual position of the motor to match with the target position.

Working:

There are four main components inside of a hobby servo, a DC motor, a gearbox, a potentiometer and a control circuit. The DC motor is high speed and low torque but the gearbox reduces the speed to around 60 RPM and at the same time increases the torque.

The potentiometer is attached on the final gear or the output shaft, so as the motor rotates the potentiometer rotates as well, thus producing a voltage that is related to the absolute angle of the output shaft. In the control circuit, this potentiometer voltage is compared to the voltage coming from the signal line. If needed, the controller activates an integrated H-Bridge which enables the motor to rotate in either direction until the two signals reach a difference of zero.

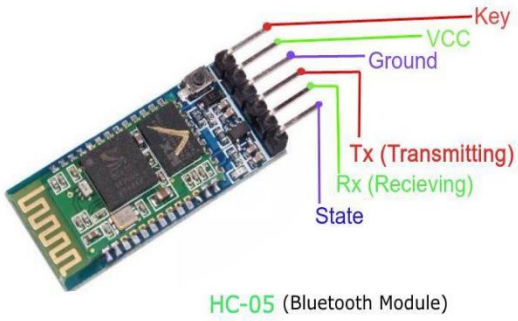


A servo motor is controlled by sending a series of pulses through the signal line. The frequency of the control signal should be 50Hz or a pulse should occur every 20ms. The width of pulse determines angular position of the servo and these type of servos can usually rotate 180 degrees (they have a physical limits of travel).

6.2.3 HC-05 Bluetooth Module

HC-05 is a Bluetooth module which is designed for wireless communication. This module can be used in a master or slave configuration. Bluetooth serial modules allow all serial enabled devices to communicate with each other using Bluetooth.

It has 6 pins:



1. **Key/EN:** It is used to bring Bluetooth module in AT commands mode. If Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is 38400bps and 9600 in data mode.

HC-05 module has two modes,

1. **Data mode:** Exchange of data between devices.
2. **Command mode:** It uses AT commands which are used to change setting of HC-05. To send these commands to module serial (USART) port is used.

2. **VCC:** Connect 5 V or 3.3 V to this Pin.

3. **GND:** Ground Pin of module.

4. **TXD:** Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)

5. **RXD:** Receive data serially (received data will be transmitted wirelessly by Bluetooth module).

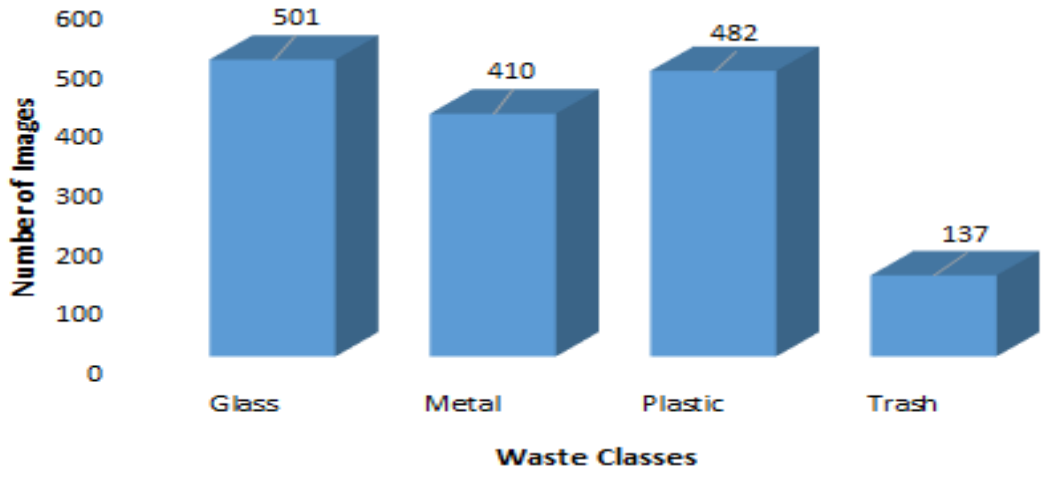
6. **State:** It tells whether module is connected or not.

HC-05 module Information

- HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not. Before connecting to HC-05 module this red LED blinks continuously in a periodic manner. When it gets connected to any other Bluetooth device, its blinking slows down to two seconds.
- This module works on 3.3V. We can connect 5V supply voltage as well since the module has on board 5 to 3.3 V regulator.
- As HC-05 Bluetooth module has 3.3V level for RX/TX and microcontroller can detect 3.3 V level, so, no need to shift transmit level of HC-05 module. But we need to shift the transmit voltage level from microcontroller to RX of HC-05 module.
- The data transfer rate of HC-05 module can vary up to 1Mbps is in the range of 10 meters.

6.3 Dataset Description:

In this study, the TrashNet dataset is used, which is publicly available on Github . It is a hand-collected dataset of a size of approximately 3.5GB. The dataset spans six classes: metal, paper, glass, cardboard, and trash. Together, these classes account for 99% of recycled material. Currently, this repository contains 2527 waste images. The dimensions of each image are 512 × 384, which can be changed or resized in the data. These images are captured by placing the object on a white poster board as a background using room lighting or sunlight. For this study, we use a subset of this dataset for four categories of waste: plastic, metal, glass, and trash, and we split it into two sets: training and testing, with ratios 80% and 20%, respectively. However, for real-time proposed model testing, we trained the whole data-set and test images captured in real-time particularly. The table reflects the sample distribution of classes for each subset. It can be noticed that the number of samples per class is not balanced. Figure below represents the dataset's number of samples among each label class.

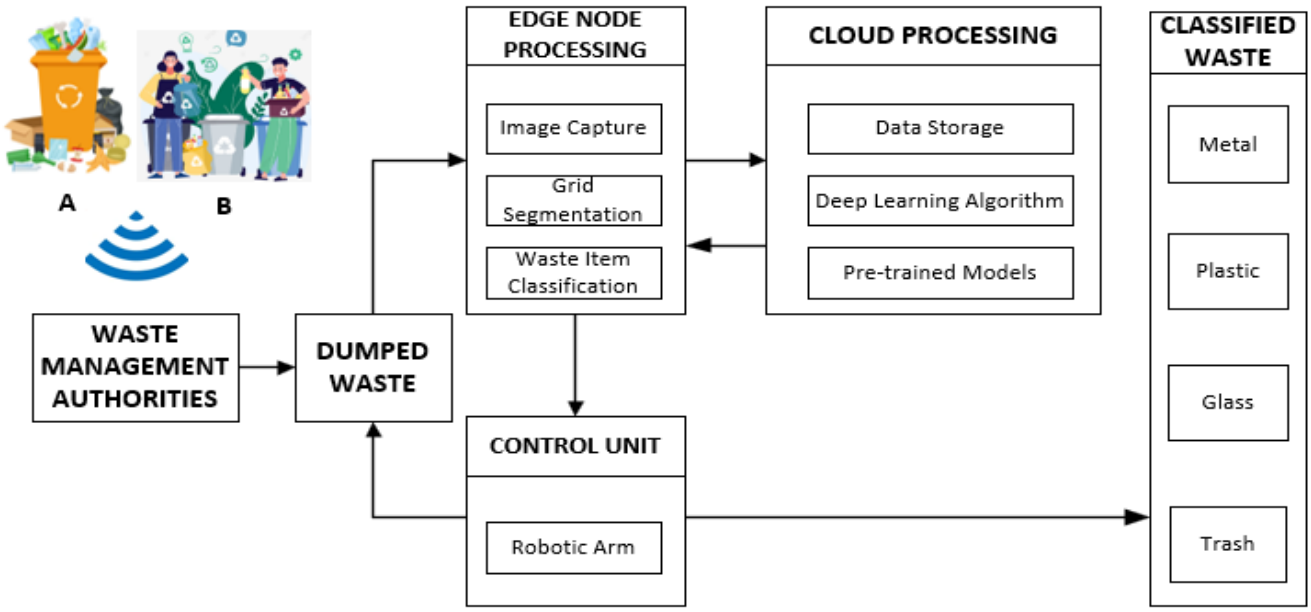


Sample Distribution For Training and Testing Subsets

Class Label	Training	Testing	Total
Metal	328	82	410
Glass	400	101	501
Plastic	385	97	482
Trash	109	28	137
Total	1222	308	1530

6.4 Proposed Methodology

The main focus of the system is to develop a system for segregation of solid waste, mainly recyclables. The proposed waste classification model is divided into three main modules i.e Edge Node, Cloud Processing and Control Unit.



6.4.1 Edge Node Processing

The edge node becomes the fundamental unit of the proposed system. It minimizes the overall response time of the system, as it is close to the source of data and efficiently uses computational resources to accomplish the desired tasks. We used a phone as an edge node to make it more effective in a real-time application. This stage is accomplished in three phases, i.e., image capturing, grid segmentation, and waste item classification.

- **Image Capture**

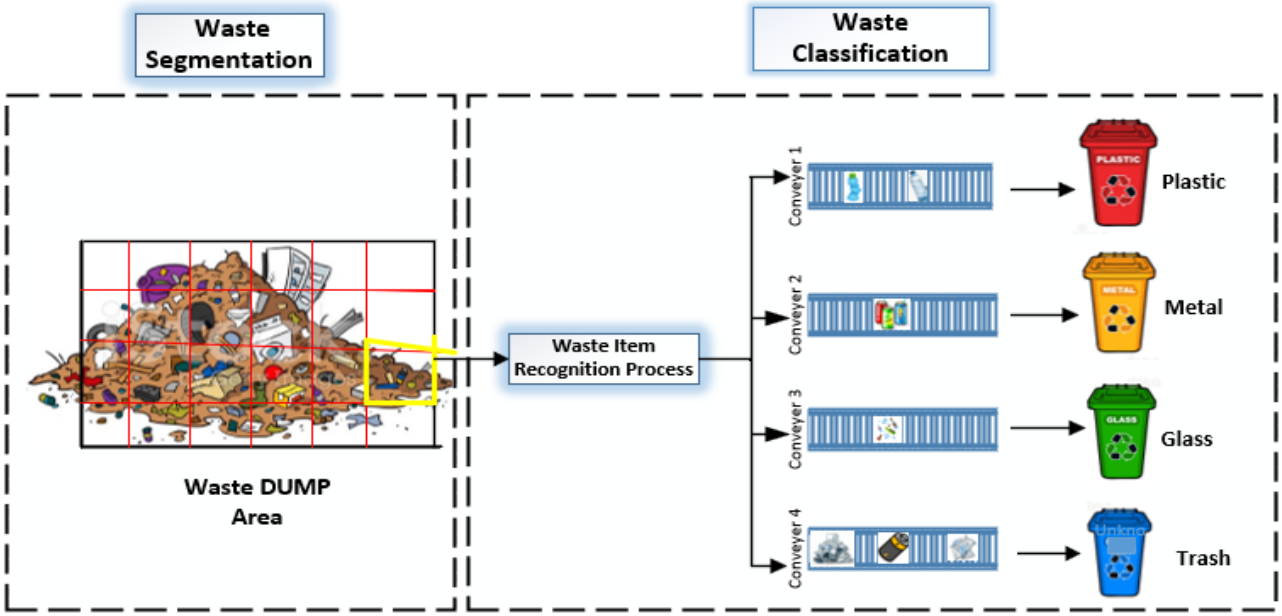
  The pie-cam is mounted on the edge node to captures a waste-dumped area image. In the centre of the pie cam sensor focus, the image size should be 1440 × 1080 pixels from a 5 mega-pixels pie cam. The captured image is further processed in the later phases to recognize the waste items effectively.

- **Grid Segmentation**

  Grid segmentation strategy is used to split the whole test image into grid-like cell structures. We took the test image from the pile of a waste dump in a controlled environment; as shown above, it contains a lot of waste objects in a single test image. We apply the grid-like cell segmentation in two steps:

  1. In the first step, we map the captured image of a waste dump on a grid-like cell structure, i.e., 5 × 6 matrix, of the same resolutions, represents the grid splits into segments and converts the segments into grayscale. The size of the grid and the total number of cells depend on the test image size.

2. Each cell segment is processed separately to recognize the waste items appropriately. After that, we move from one segment to another column-wise to the VGG16 algorithm to recognize the waste item effectively. Each cell contains one waste object at a time, which is ultimately picked by the robotic arm and placed in a respective bin. This process continues until the whole test image has been processed. Incorrect segments are put back, and final labelling or classification consists of the union of only correct segments.



- **Waste Item Classification**

Once the waste dumped image is converted into the grid-like structure, each grid cell is processed to recognize the appropriate waste item, as shown in the figure. We use the VGG16, a deep learning-based classification mechanism, to identify the waste item. The segregated waste item is managed with the help of the control unit, as discussed in the control unit module.

6.4.2 Cloud Processing

The cloud processing platform assists in computing heavy processing processes and algorithms. It can also facilitate managing and storing large datasets on which the whole system can be trained. The major processes of this module are data storage, deep learning algorithm, and pre-trained model

6.4.3 Control Unit

The control unit is the core part of the whole system. It can generate control signals based on the input received from the edge processing module. It controls the robotic arm movement according to its degree of freedom (DOF) specification. We use the five DOF robotic arm gripper to pick the recognized waste item to place it into the segregated waste bin as per the recognized waste item category, i.e., metal, plastic, glass, and trash, received from the edge processing module.

6.4.4 Data Storage

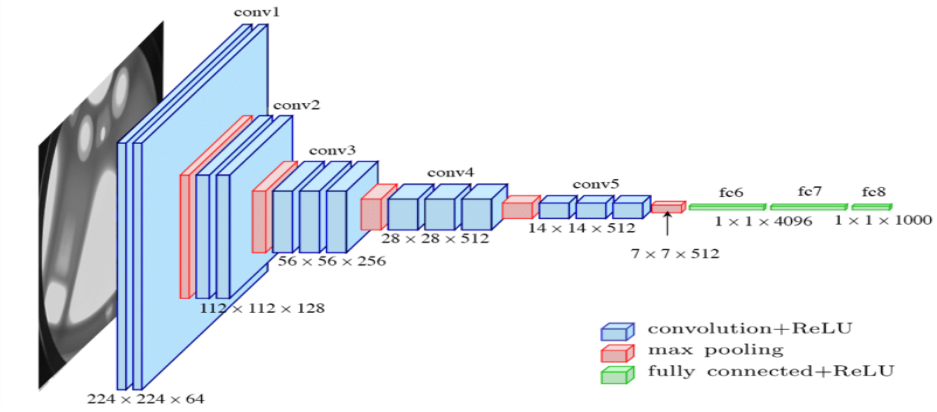
We use the “trashnet” waste items dataset. It comprises 2527 waste item image. To compute this dataset, we need to store it on a dedicated storage resource, as the edge node cannot handle this vast dataset. So, we use cloud storage resources to place this dataset, which was ultimately processed by the Google AI module for deep-learning model computations on the cloud.

6.5 Deep Learning Algorithm

In Computer vision, the trend is to design deeper more complicated and deeper networks to achieve higher accuracy. However deeper network are slower and takes more space, hence deeper networks are harder to optimize. In real time smart applications, such as in our Internet of Things (IoT)-based system, the object detection and recognition tasks must be able to be performed with computationally limited cost and time. **VGG16** alternative networks are computationally costly. The VGG16 network is characterized by its simplicity and deeper architecture with smaller kernel sizes; thus, it is suitable for our real-time smart waste classification process with greater efficiency and speed.

6.5.1 VGG-16 Pre-Trained Model

VGG16, short for Visual Geometry Group 16, is a convolutional neural network (CNN) architecture designed for image classification. It was proposed by the Visual Geometry Group at the University of Oxford and achieved second place in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. The architecture is characterized by its simplicity and uniformity.



Some features are as follows:

1. **Architecture:** VGG16 consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. The convolutional layers are followed by max-pooling layers to reduce spatial dimensions. The convolutional layers use small receptive fields (3x3) with a stride of 1, and the max-pooling layers use 2x2 pooling windows with a stride of 2.
2. **Layer Configuration**: The 16 layers are organized into blocks. The first two blocks consist of two convolutional layers each, and the last three blocks contain three

convolutional layers each. The convolutional layers are followed by rectified linear unit (ReLU) activation functions to introduce non-linearity into the network.

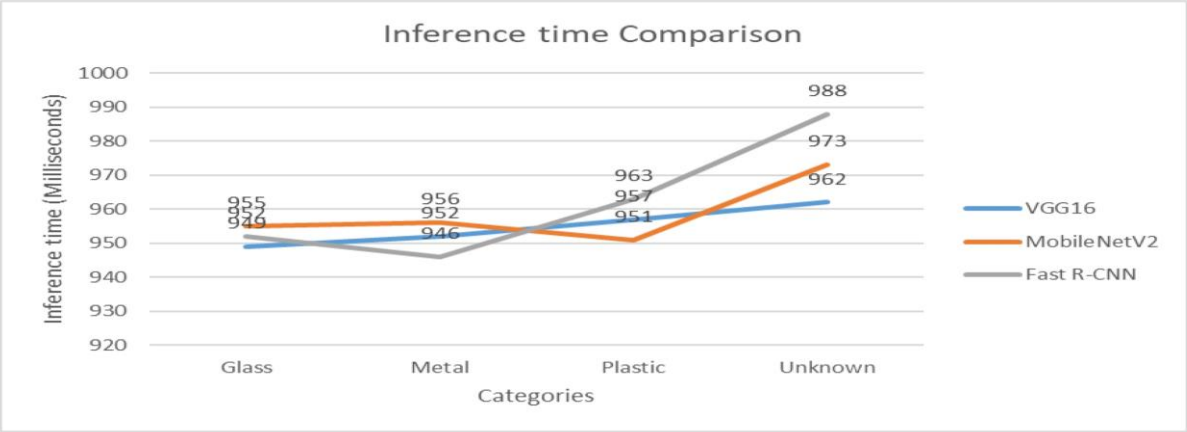
3. **Filter Size and Depth**: The small 3x3 convolutional filters are used to process the input data, and these filters are deepened to capture more complex patterns in the images. The depth of the filters (number of channels) increases as you go deeper into the network, from 64 in the early layers to 512 in the later layers.
4. **Pooling Layers**: VGG16 uses max-pooling layers with 2x2 pooling windows and a stride of 2. Max-pooling helps to downsample the spatial dimensions of the input, reducing the computational load and increasing the receptive field.
5. **Fully Connected Layers**: The final three layers are fully connected layers, which are responsible for combining the features learned by the convolutional layers to make predictions. The first two fully connected layers have 4,096 neurons each, and the final fully connected layer has 1,000 neurons for the 1,000 ImageNet classes. The final layer is often equipped with a softmax activation function for multi-class classification.
6. **Activation Function**: Throughout the network, rectified linear units (ReLU) are used as activation functions. ReLU helps introduce non-linearity to the model.
7. **Input Size and Pre-Processing**: The standard input size for VGG16 is 224x224 RGB images. Images are typically preprocessed by subtracting the mean pixel value computed on the training dataset from each pixel.

6.6 Performance Analysis of Pre-Trained Algorithms:

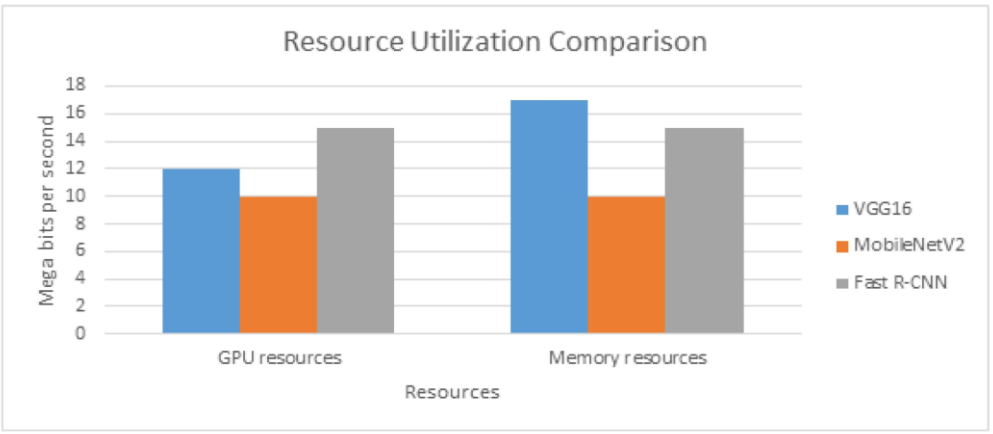
1. **Error Rate Comparison**: The figure below shows the error rate incurred during the training of VGG-16 which is used in the proposed system. The total error rate at the start (after 1 epoch) is around 0.68, which gradually decreases with the increase in the number of epochs, falling to approximately 0.48 after first 20 epochs. After that, it follows a similar downward trend, and the total error rate drops to 0.13 after 100 epochs. VGG-16 performs better in comparison to other algorithms like Mobile-Net v2 and Fast R-CNN.

2. **Inference Time Comparison:** The inference time is the total time a model takes to take a real-life input and deliver an actionable output. All three algorithms show

optimal performance, as shown. Minimal difference is observed in the algorithms, i.e., Fast R-CNN, VGG16, and Mobile-NetV2. Lighter Models generally have less inference time that deeper models and hence less computationally expensive, while object detection models like Fast R-CNN might have greater inference time than image detection models.



3. **Resource Utilization:** The figure below shows the comparison of resource utilization between VGG16, MobileNetV2, and Fast R-CNN. The GPU resources are used by Fast R-CNN much more than the other two models for training because of the large size of the model and fast training.



Accuracy of trained Model after 12 Epochs

```
Epoch 1/20
7/7 [==============================] - 564s - loss: 0.4193 - acc: 0.8143 - val_loss: 0.3700 - val_acc: 0.8667
Epoch 2/20
7/7 [==============================] - 540s - loss: 0.3735 - acc: 0.8762 - val_loss: 0.3287 - val_acc: 0.8889
Epoch 3/20
7/7 [==============================] - 531s - loss: 0.3773 - acc: 0.8571 - val_loss: 0.3704 - val_acc: 0.8667
Epoch 4/20
7/7 [==============================] - 538s - loss: 0.3807 - acc: 0.8524 - val_loss: 0.3384 - val_acc: 0.8667
Epoch 5/20
7/7 [==============================] - 531s - loss: 0.3425 - acc: 0.8667 - val_loss: 0.3577 - val_acc: 0.8778
Epoch 6/20
7/7 [==============================] - 538s - loss: 0.3263 - acc: 0.8714 - val_loss: 0.3285 - val_acc: 0.8556
Epoch 7/20
7/7 [==============================] - 531s - loss: 0.3623 - acc: 0.8762 - val_loss: 0.3339 - val_acc: 0.8556
Epoch 8/20
7/7 [==============================] - 540s - loss: 0.3084 - acc: 0.9000 - val_loss: 0.3309 - val_acc: 0.8778
Epoch 9/20
7/7 [==============================] - 534s - loss: 0.3326 - acc: 0.8571 - val_loss: 0.3221 - val_acc: 0.8889
Epoch 10/20
7/7 [==============================] - 532s - loss: 0.2886 - acc: 0.9048 - val_loss: 0.3160 - val_acc: 0.8667
Epoch 11/20
7/7 [==============================] - 538s - loss: 0.3361 - acc: 0.8476 - val_loss: 0.3144 - val_acc: 0.8556
Epoch 12/20
7/7 [==============================] - 531s - loss: 0.3031 - acc: 0.9000 - val_loss: 0.3158 - val_acc: 0.8556
```

6.7 Libraries Used In This Project

6.7.1 OpenCV and Computer Vision

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million.

6.7.2 TensorFlow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

6.7.3 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

The code is hosted on GitHub.

6.7.4 Streamlit

Streamlit is an open-source app framework that lets you create web apps from data scripts in pure Python, without front-end experience. You can deploy, share, and interact with your apps on Streamlit Community Cloud.

7. Critical Analysis

To achieve we use the VGG16 algorithm. It is the most appropriate algorithm in terms of computation and accuracy for the object detection and classification process. It has 16 convolutional layers and some flattening layers at the end. These convolutional layers are computed to determine the valuable features on the basis of weights assigned to each feature via a deep learning algorithm at the flattening output layer in order to calculate the overall performance metrics of the proposed system. The dataset is trained by the VGG16 deep learning algorithm up to 12 epochs and it achieves an accuracy level of 90%. After 12 epochs we stopped training as we can get the desired level of accuracy. The epochs are basically referred to as the total number of times the model is trained. The accuracy achieved after a single epoch is close to 81%,

8. Conclusion

Image classification for recycling has the potential to create significant social impact in a number of ways:

1. Environmental sustainability: By automating the sorting of recyclable materials, image classification can help increase the recovery of valuable materials and reduce waste. This can help promote environmental sustainability by conserving resources and reducing greenhouse gas emissions associated with the production of new materials.

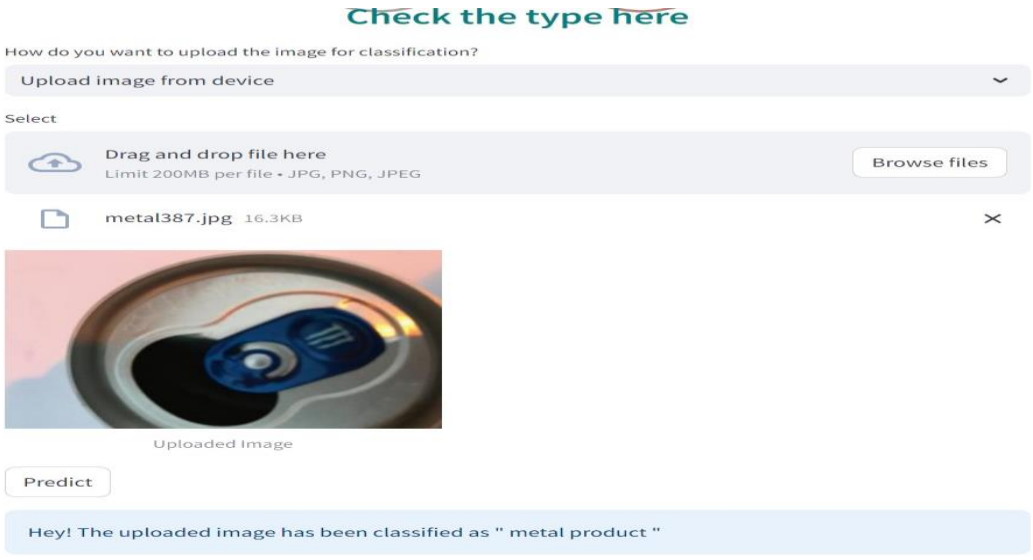
2. Job creation: The implementation of image classification technology in recycling facilities can create new job opportunities for individuals with skills in technology and data analysis. These jobs can help promote economic growth and provide new career pathways for individuals in the recycling industry.

3. Education and awareness: Image classification for recycling can also be used as a tool for education and raising awareness about the importance of recycling. By creating more efficient and accurate recycling processes, individuals and communities may become more engaged in sustainable practices and waste reduction efforts.

4. Equity and access: Image classification can also help promote equity and access to recycling services, particularly in communities that may have limited resources or access to recycling facilities. By making recycling processes more efficient, these communities may have greater access to recycling services and a reduced burden of waste disposal.

Overall, image classification for recycling has the potential to create significant social impact by promoting environmental sustainability, job creation, education and awareness, and equity and access.

9. Results:



10. References

1. NICHOLAS CHIENG ANAK SALLANG, MOHAMMAD TARIQUL ISLAM, (Senior Member, IEEE), MOHAMMAD SHAHIDUL ISLAM, (Member, IEEE), AND HASLINA ARSHAD 2A CNN-Based Smart Waste Management System Using TensorFlow Lite and LoRa-GPS Shield in Internet of Things Environment.

2. Olugboja Adedeji, Zenghui Wang. Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network.

3. Sehrish Munawar Cheema, Abdul Hannan and Ivan Miguel Pires**.** SmartWaste Management and Classification Systems Using Cutting Edge Approach.

4. Sayali Deepak Apte, Sayali Sandbhor, Rushikesh Kulkarni and Humera Khanum. Machine learning approach for automated beach waste prediction and management system: A case study of Mumbai.

5.Rijwan Khan , Santosh Kumar ,Akhilesh Kumar Srivastava, Niharika Dhingra, Mahima Gupta, Neha Bhati and Pallavi Kumari. Machine Learning and IoT-Based Waste Management Model.

6. Nagaveni C M. An Automated Machine Learning Approach For Smart Waste Management System.

7. Trashnet Waste Categories Dataset Github Link. Available online: https://github.com/garythung/trashnet (accessed on 10 August 2022).