SORTING AND SEGREGATION OF WASTE MATERIALS USING COMPUTER VISION

By: Bengaluru India Chapter
https://omdena.com/omdena-chapter-page-india/

AGENDA

INTRODUCTION

Problem Statement:

Many waste management facilities are facing an immense problem relating to recycling of waste without finding impurities (misclassified waste) , thus leading to high cost of waste processing and labour. Waste recycling becomes challenging if the waste products are not segregated. Recyclers have to setup methods to identify the waste recycling categories and it is often laborious process and time consuming.

The aim of this project is to find a way to segregate and categorise the waste to mitigate the problems mentioned above. With the help of computer vision, waste management facilities can build or implement robots that can detect the waste type based on product, shape, size, colour, etc. and automate the task of separation resulting in smart recycling. It is also expected to know the amount of a particular type of waste generation based on the modeling data and this can be utilized further to take correct actions by the authorities.

Solution:

This project focuses on solving this issue by creating solutions in multiple forefronts as below:

1. Using Computer Vision Techniques to identify and classify different types of waste materials
2. Generate Report/ Presentation on Findings

TIMELINE (July 7th - Aug 14th, 2022)

WEEK 1, **WEEK 2**

- Research/Resource exploration
- Web Scraping
- Data Collection
- Data Preprocessing

WEEK 3, **WEEK 4**

- Datasets Modification Based on the Requirements
- Models Resource Findings
- Models Creation for various Datasets

WEEK 5

- Model Deployment Using Frameworks & Tools
- Final Tuning
- Presentation

Sub-Tasks Outline

#1 Collection of Resources & Datasets

#2 Data Pre-processing & Preparation

#3 Model Building

#4 Deployment

Data Collection & Pre-processing

1. Initiated the process of researching and findings on the required datasets and readings about the white papers about segregating the waste collections using AI technologies.

2. Discussion about the collected datasets and came up with the specific datasets that are more relevant to the task.

3. Worked on merging & renaming of the specific categories from selected datasets.

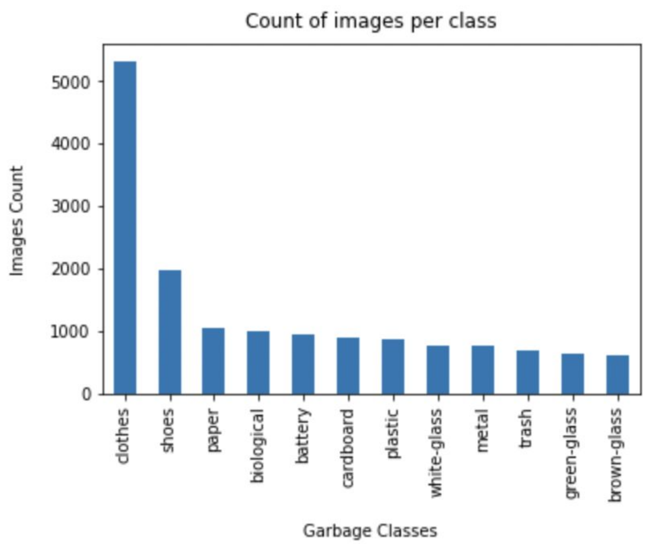
Hurdles in the Dataset

- While working with the Dataset 11,the common issues confronted by the team were:

1. The high no of glass categories mainly based on color
   I.e ‘green-glass’ , ‘brown-glass’ and ‘white-glass’.

2. Incorrect labelling of food wastes as ‘Trash’ and medical wastes as ‘biological’ which created confusion.

3. Non availability of plastic wrapper images which are a major contributor to the new age waste.



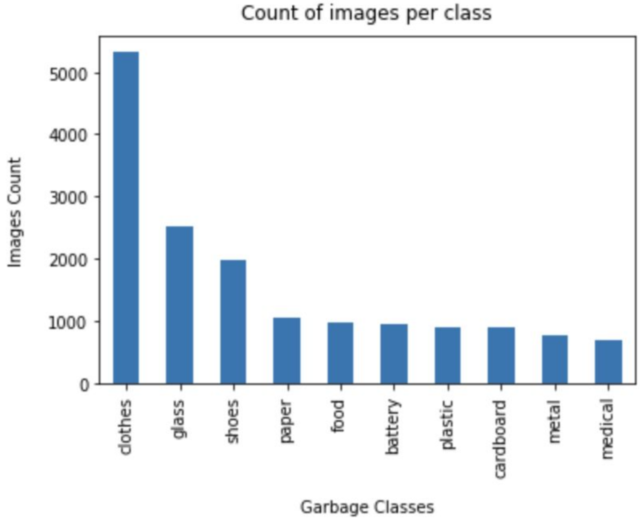
Original Distribution of Images and Classes of the Dataset

Key Modifications Made in the Dataset

1. Concatenated all glass images into one single category ‘glass’.

2. Renamed the labels to avoid confusion:
   - ‘trash’ as ‘food’
   - ‘biological’ as ‘medical’

3. Added plastic wrapper images like chips packets, toffee wrappers etc from other public datasets and also from the web.



Distribution of Images and Classes of the Dataset after Modifications

Image Samples from All Categories Used

1. **Battery :**

2. **Cardboard :**

3. **Clothes:**

4. **Food :**

5. **Glass :**

6. **Medical :**

7. **Metal :**

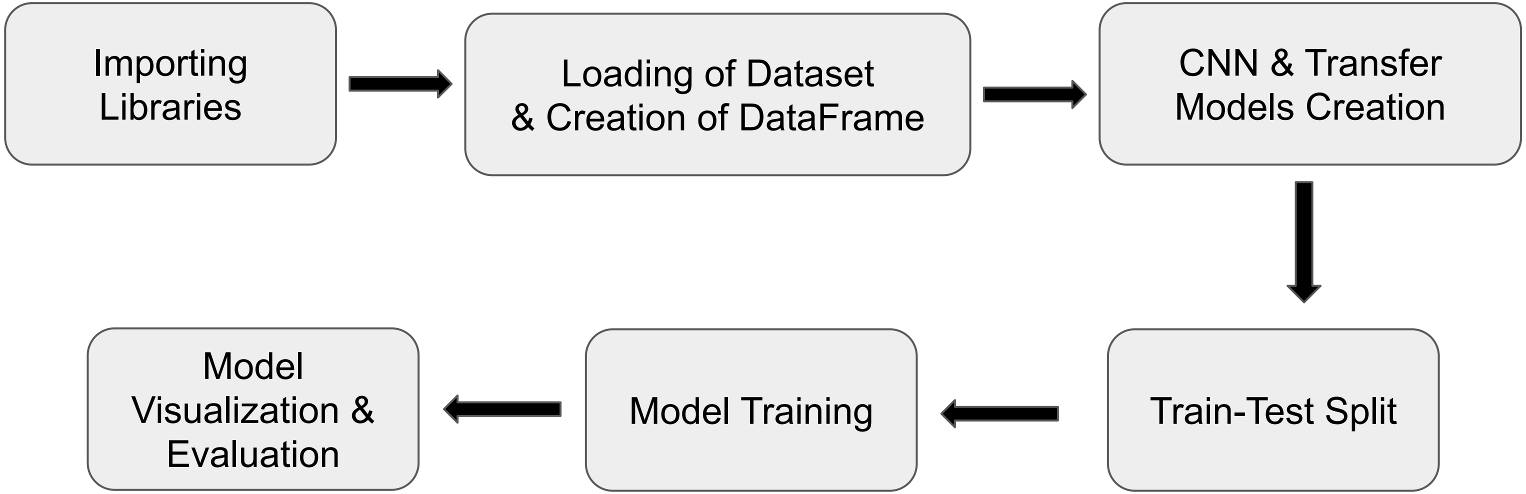
8. **Paper :**

9. **Plastic :**

10. **Shoes :**

MODEL BUILDING

Steps Followed in Model Building :



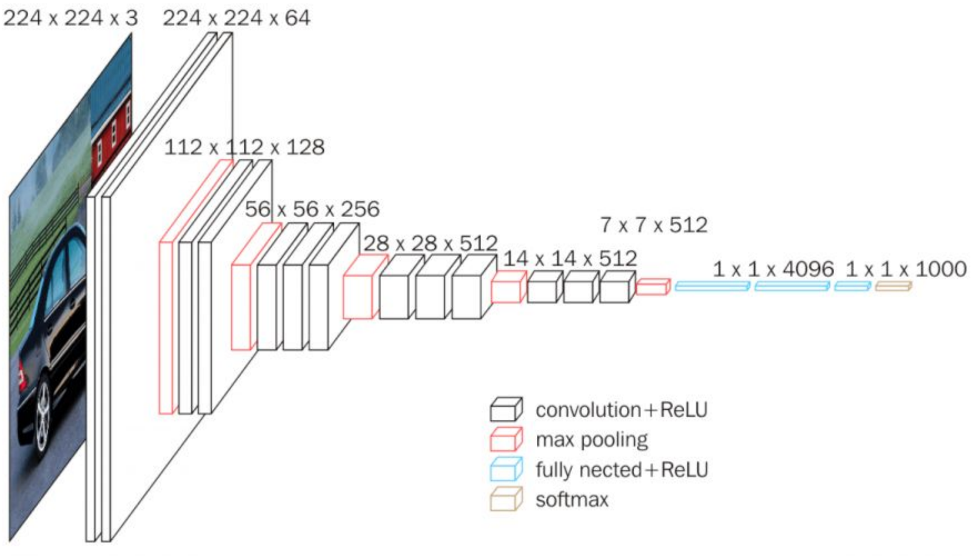
Model Performance Comparison

Sl No.	Model Architecture	Dataset Used	Training Accuracy	Testing Accuracy	No.of Epochs (Early Stopping)
1.	Custom CNN V1	Dataset 1	55%	53%	20 Epochs
2.	Custom CNN V2	Dataset 11	86%	76%	20 Epochs
3.	Custom CNN V3	Dataset 11	73%	70%	10 Epochs
4.	MobileNetV2	Dataset 11	96%	91%	20 Epochs(15)
5.	XceptionNet	Dataset 11	97%	93%	20 Epochs(13)
6.	VGG19	Dataset 11	94 %	91%	100 Epochs(16)
7.	VGG16 + SVM	Dataset 11	97%	93%	20 Epochs(16)

Final Model Architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda (Lambda)              (None, 224, 224, 3)       0
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 10)                5130
=================================================================
Total params: 14,719,818
Trainable params: 5,130
Non-trainable params: 14,714,688
_________________________________________________________________
```

Architecture of Final Model



VGG16 Architecture

Code Snippets

Import Required Libraries

```
import numpy as np
import pandas as pd
import random
import os
import matplotlib.pyplot as plt
import seaborn as sns
import zipfile
import sys
import time
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input

import tensorflow.keras as keras
import tensorflow as tf
import re

from PIL import Image
from keras.layers import Input, Conv2D, Dense, Flatten, MaxPooling2D, Input, GlobalAveragePooling2D
from tensorflow.keras.layers.experimental.preprocessing import Normalization
from keras.models import Model, Sequential
from keras.preprocessing import image
from tensorflow.keras.utils import to_categorical
from keras.layers import Lambda
from keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

print('setup successful!')
```

```
setup successful!
```

Define Constants

```
# Increasing the image size didn't result in increasing the training accuracy
IMAGE_WIDTH = 224
IMAGE_HEIGHT = 224
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS = 3


# Path where our data is located
base_path = "../input/garbage-seg-10-v5/Garbage Seg 10 V5/"

# Dictionary to save our 12 classes
categories = {0: 'paper', 1: 'cardboard', 2: 'plastic', 3: 'metal', 4: 'food', 5: 'battery',
              6: 'shoes', 7: 'clothes', 8: 'glass',9: 'medical'}

print('defining constants successful!')
```

```
defining constants successful!
```

Code Snippets

Create DataFrame

We want to create a data frame that has in one column the filenames of all our images and in the other column the corresponding category. We Open the directories in the dataset one by one, save the filenames in the filenames_list and add the corresponding category in the categories_list

```
[ ]
# Add class name prefix to filename. So for example "/paper104.jpg" become "paper/paper104.jpg"
def add_class_name_prefix(df, col_name):
    df[col_name] = df[col_name].apply(lambda x: x[:re.search("\d",x).start()] + '/' + x)
    return df

# list conatining all the filenames in the dataset
filenames_list = []
# list to store the corresponding category, note that each folder of the dataset has one class of data
categories_list = []

for category in categories:
    filenames = os.listdir(base_path + categories[category])

    filenames_list = filenames_list  +filenames
    categories_list = categories_list + [category] * len(filenames)

df = pd.DataFrame({
    'filename': filenames_list,
    'category': categories_list
})

df = add_class_name_prefix(df, 'filename')

# Shuffle the dataframe
df = df.sample(frac=1).reset_index(drop=True)

print('Number of Elements = ' , len(df))
```

```
Number of Elements =  16059
```

Code Snippets

Create the model

The steps are:

1. Create an mobilenetv2 model without the last layer and load the ImageNet pretrained weights
2. Add a pre-processing layer
3. Add a pooling layer followed by a SVM at the end

```
[ ]
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation, BatchNormalization
import keras.applications.vgg16 as vgg16
from tensorflow.keras.regularizers import l2


vgg16_layer = VGG16(include_top = False, input_shape = (IMAGE_WIDTH, IMAGE_HEIGHT,IMAGE_CHANNELS),weights=None)

vgg16_layer.load_weights("../input/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5")
# IF we don't want to train the imported weights
vgg16_layer.trainable = False


model = Sequential()
model.add(keras.Input(shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)))

#create a custom layer to apply the preprocessing
def vgg16_preprocessing(img):
  return vgg16.preprocess_input(img)

model.add(Lambda(vgg16_preprocessing))

model.add(vgg16_layer)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Dense(len(categories),kernel_regularizer=tf.keras.regularizers.l2(0.01),activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])

model.summary()
```

```
2022-08-09 09:44:19.216202: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had
2022-08-09 09:44:19.337676: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had
```

Code Snippets

Split the Data Set

We split the training set into three separate sets:

1. **The training set:** used to train our model.
2. **The validation set**: used to double check that our model is not overfitting the training set, i.e. it can also generalise to other data other than the train data
3. **The Test set:** Used to estimate the accuracy of the model on new data other than the ones the model used for training For a competition or for some other cases, you can split the data only to training and validation sets in order to achieve the highest possible accuracy, without the need to properly estimate how accurate the model really is.

We split the data set as follows: 80% train set, 10% cross_validation set, and 10% test set

```
[ ]
#Change the categories from numbers to names
df["category"] = df["category"].replace(categories)

# We first split the data into two sets and then split the validate_df to two sets
train_df, validate_df = train_test_split(df, test_size=0.2, random_state=42)
validate_df, test_df = train_test_split(validate_df, test_size=0.5, random_state=42)

train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
test_df = test_df.reset_index(drop=True)

total_train = train_df.shape[0]
total_validate = validate_df.shape[0]

print('train size = ', total_validate , 'validate size = ', total_validate, 'test size = ', test_df.shape[0])
```

```
train size =  1606 validate size =  1606 test size =  1606
```

Code Snippets

Train the model

We will first create the training data generator, that will get the images from the input data directory to train on them. We will also create a generator for the validation set.

Applying Data Augmentation on the training set was taking too long to be executed and the initial results didn't show much improvement than the results without augmentation, so I commented the augmentation to make the training faster. However fell free to uncomment the Data Augmentation lines in the following cell and play a bit with it.

```
[ ]
batch_size=64

train_datagen = image.ImageDataGenerator()

train_generator = train_datagen.flow_from_dataframe(
    train_df,
    base_path,
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

```
Found 12847 validated image filenames belonging to 10 classes.
```

Code Snippets

```python
validation_datagen = image.ImageDataGenerator()

validation_generator = validation_datagen.flow_from_dataframe(
    validate_df,
    base_path,
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

Python

```python
EPOCHS = 20
history = model.fit_generator(
    train_generator,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=total_validate//batch_size,
    steps_per_epoch=total_train//batch_size,
    callbacks=callbacks
)
```

Python

Code Snippets

Evaluate the test

To evaluate the performance of our model we will create a test generator to load the images from the input data directory and evaluate the results.

```
[ ]
test_datagen = image.ImageDataGenerator()

test_generator = test_datagen.flow_from_dataframe(
    dataframe= test_df,
    directory=base_path,
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    color_mode="rgb",
    class_mode="categorical",
    batch_size=1,
    shuffle=False
)
```

```
Found 1606 validated image filenames belonging to 10 classes.
```

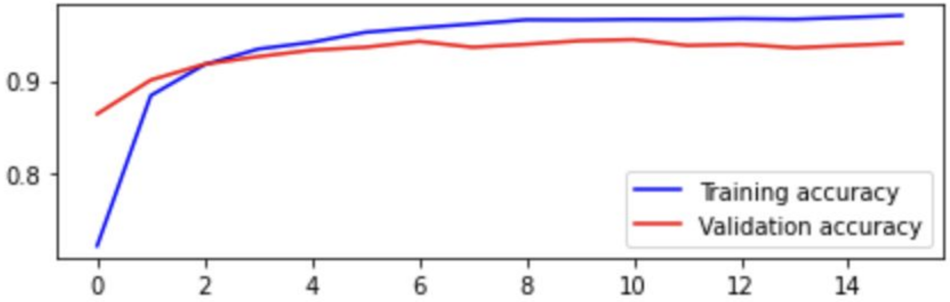
```
[ ]
filenames = test_generator.filenames
nb_samples = len(filenames)

_, accuracy = model.evaluate_generator(test_generator, nb_samples)

print('Accuracy on test set = ',  round((accuracy * 100),2 ), '% ')
```

```
/opt/conda/lib/python3.7/site-packages/keras/engine/training.py:2006: UserWarning: `Model.evaluate_generator` is deprecated
  warnings.warn('`Model.evaluate_generator` is deprecated and '
Accuracy on test set =  93.15 %
```

Performance Report and Graph of Final Model

	precision	recall	f1-score	support
battery	0.99	0.93	0.96	90
cardboard	0.91	0.92	0.92	89
clothes	0.99	0.98	0.98	533
food	0.99	0.93	0.96	98
glass	0.94	0.89	0.91	245
medical	0.88	0.94	0.91	72
metal	0.70	0.88	0.78	72
paper	0.89	0.89	0.89	98
plastic	0.77	0.74	0.75	93
shoes	0.94	0.99	0.96	216
accuracy			0.93	1606
macro avg	0.90	0.91	0.90	1606
weighted avg	0.93	0.93	0.93	1606

Performance Report



Training vs Validation Acc. Graph

MODEL DEPLOYMENT
Tools Used

gradio

Streamlit

How the Deployment Works

1. **GRADIO**

- Train, Save and Export Model
- Load the Model
- Creation of app.py to write the interface and working function.
- Run the Gradio App Interface on local host.
- Uploading the app with all the files to Hugging Faces for hosting.

2. **STREAMLIT**

- Train, Save and Export Model
- Load the Model
- Creation of app.py to write the app interface and working function
- Import libraries like ngrok to connect local host to internet.
- Run it to see the app running on internet

For easy and time saving,app building process and for reliable hosting through hugging face. We decided to go ahead with Gradio over Streamlit for final deployment.

Gradio Code Snippets

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

app.py 9

Users > dinoking > Garbage-Classification-VGG19 > app.py > ...

```python
import gradio as gr
import tensorflow as tf
import numpy as np
from PIL import Image
import tensorflow.keras as keras
import keras.applications.vgg16 as vgg16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.models import load_model

# load model
model = load_model('model520.h5')

#prediction classes
#classnames = ['paper', 'cardboard', 'plastic', 'metal', 'food', 'battery', 'shoes', 'clothes', 'glass', 'medical']
classnames = ['battery','cardboard','clothes','food','glass','medical','metal','paper','plastic','shoes']

#prediction function
def predict_image(img):
  img_4d=img.reshape(-1,224, 224,3)
  prediction=model.predict(img_4d)[0]
  return {classnames[i]: float(prediction[i]) for i in range(len(classnames))}


#Gradio interface
image = gr.inputs.Image(shape=(224, 224))
label = gr.outputs.Label(num_top_classes=3)
article="<p style='text-align: center; font-weight:bold;'>Model based on the VGG-16 CNN</p>"
examples = ['battery.jpeg', 'clothes.jpeg', 'plastic.jpg']

gr.Interface(fn=predict_image, inputs=image,  title="Garbage Classifier VGG-19",
    description="This is a Garbage Classification Model Trained using VGG-19 architecture. Deployed to Hugging Face using Gradio.", outputs=label, examples=examples, article=article,
    enable_queue=True, interpretation='default').launch(share="True")
```

Importing the Required Libraries

Loading the Model

Providing the Predcdiction Classes

Writing Prediction Function

Writing the App Interface

Gradio App Interface

Step 1: Go to Hugging Face App Link and open your space
Step 2:You will be greeted with the App Interface
Step 3: Follow the steps given the image to load the desired picture for prediction.

Step 4: Follow the steps in the image to see the prediction for your desired image

POTENTIAL NEXT STEPS

1. Most Images used had single object in them and had a plain background but in real day usage the images are going to be cluttered and shall not have a plain background.So better images with varied objects and backgrounds can be added to the training set for better predictions in real day world.

2. Ensure proper scaling of model to a final production stage by adding more prediction classes as per need.

3. Also the model can be extended to a connected robot or other IOT device to automatically sort the waste into classes. It can also be connected to a live feed camera for continuous classification for manual sorting .

4. Appropriate data analysis of the classification data to create proper awareness of public and for authorities to make adequate concrete policies for sustainability.

REFERENCES

Below are some of the resources which were helpful for the project :

- **Datasets**:

  - TrashNet

  - Garbage Classification (12 classes)[Kaggle- Dataset 11]

  - Garbage Classification(6 Classes)[Kaggle- Dataset 1]

- **Research Papers and Projects** :

  - Fine-Tuning Models Comparisons on Garbage Classification for Recyclability(Umut Ozkaya and Levent Seyfi)

  - Image Classification Using SVM

  - Building a Image Classifier using SVM Blog

CONCLUSION

When the three major outcomes of these projects are implemented at full scale, we could see following benefits:

- The waste classification model will scale down the task of waste segregation for the public authorities which is one of the most exorbitant tasks in waste collection.It shall cost less manual labour and shall be time-saving.

- Additionally, proper future modifications with time can be done for adding of more categories of waste based on the particular area’s livelihood.

- The waste classification lets us know about the high usage of non-recyclable waste such as plastic bags,bubble wrap,glass etc and shall help raise awareness to shift focus sustainable materials in daily life.

OUR TEAM

- ➔ Ramya N
- ➔ Sudhanva Satish
- ➔ Aditya Narendra
- ➔ Disha Aggarwal
- ➔ Naga Karthik Jetti
- ➔ Priyank
- ➔ Nandar
- ➔ Pradyumn

CHAPTER CO-LEADS:-

HARDIK SEJU
PRATHIMA KADARI

CHAPTER LEAD:-

MUHAMMAD YAHIYA