1.INTRODUCTION

1.1 Overview

With the increase in the number of industries in the urban area, the disposal of solid waste is really becoming a big problem, and 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. In this project, we will be building a deep learning model that can detect and classify types of garbage.

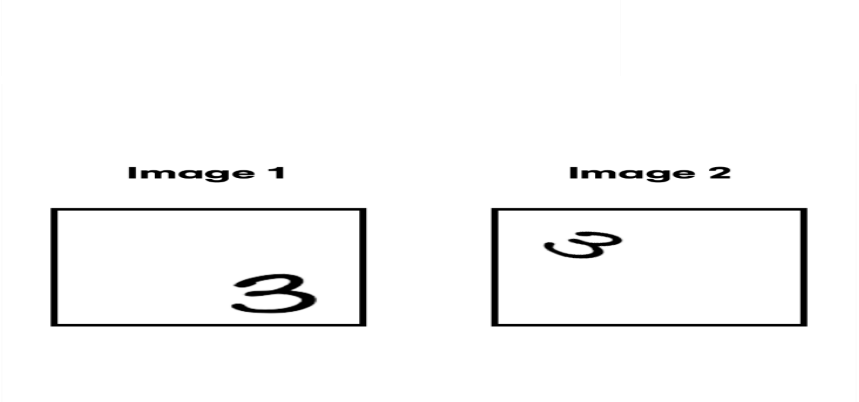
1.2 Purpose

The main aim of project is to building a model is used for classifying five types of waste. Garbage Waste Classification can reduce the cost of terminal waste disposal and improve the efficiency of the overall disposal process, as well as prevent harmful substances from polluting water and land resources and avoiding landfill pollution.

2.Literature Survey

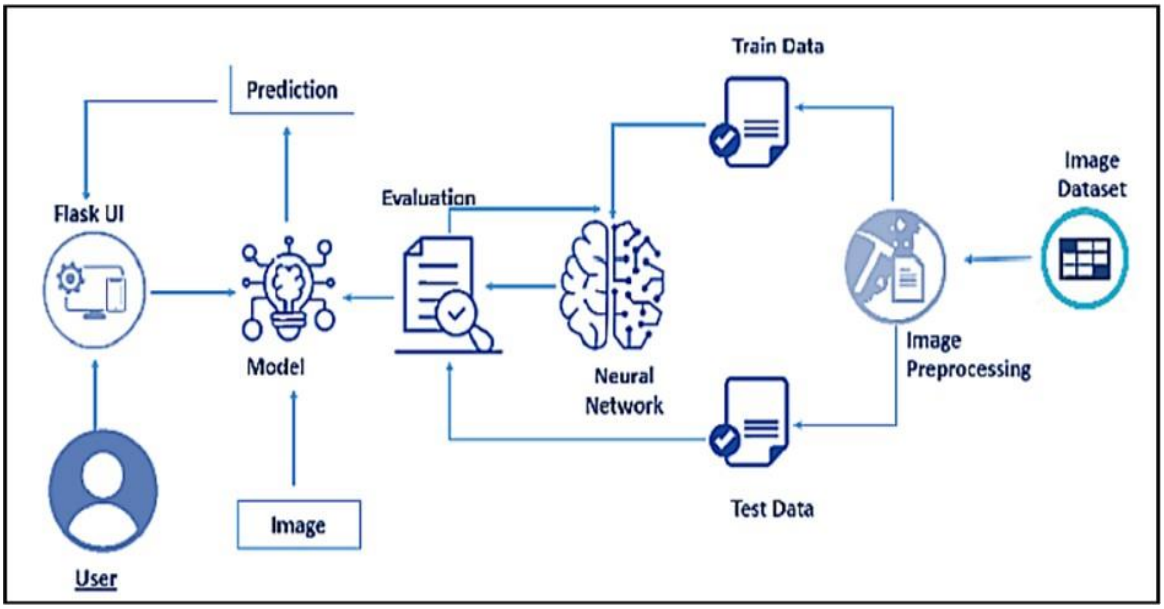
2.1 Proposed Solution

The proposed aim is to build a classifier that sorts out different types of waste, we will need a model architecture denominated as Convolutional Neural Network since our dataset will be composed mainly of labeled images, meaning that each image has a corresponding label that indicates the prediction (type of waste material) the model will have to provide as output. At the same time, the model will also need a fully-connected network after the convolutional module to transform an arbitrary response given by it to a set of values with a particular structure that will allow us to determine the class predicted by the model. The aim of using a Convolutional Network to process an image resides in its ability to extract certain patterns or features from images with an invariance in position, rotation, and scale. To understand the power of these properties when detecting features in images, let's consider an example.



3.Theoritical Analysis

3.1 Block diagram



3.2 Hardware/Software Specifications

➤ Hardware

Operating System : Windows, Mac, Linux

CPU : Multi Core Processors(i3 or above)

Requirements : Anaconda Navigator, Jupyter Notebook, Spyder

➢ Software

Python : v3.9.0 or above

Python Packages : tensorflow, flask, keras, numpy, pandas,

Web Browser : Mozilla Firefox, Internet Explorer, Google Chrome

IBM Cloud : Watson Studio - Model Training & Deployment as Machine Learning Instance

4. Experimental Investigation

Training and Testing code using given dataset

In [1]:
```
import tensorflow as tf
```

Importing libraries

In [2]:
```
from keras.models import Sequential
from keras.layers import Convolution2D,Flatten,Dense,MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
```

Loading Images

In [3]:
```
train_datagen=ImageDataGenerator(horizontal_flip=True,rescale=1./255,zoom_range=0.2)
#rescale=1./255 means transform every pixel value from range [0,255] -> [0,1].
```

In [4]:
```
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [5]:
```
X_train=train_datagen.flow_from_directory(r'C:\Users\Dell\Downloads\garbage collection\dataset\Training',

                                          target_size=(128,128),class_mode='categorical',batch_size=100)
```

```
Found 2508 images belonging to 6 classes.
```

In [6]:
```
X_test=test_datagen.flow_from_directory(r'C:\Users\Dell\Downloads\garbage collection\dataset\Testing',
                                        target_size=(128,128),class_mode='categorical',batch_size=100)
```

```
Found 464 images belonging to 6 classes.
```

In [7]:
```
X_train.class_indices
```

Out[7]:
```
{'cardboard': 0, 'glass': 1, 'metal': 2, 'paper': 3, 'plastic': 4, 'trash': 5}
```

ModelBuilding

In [8]:
```
model=Sequential()
```

In [9]:
```
#1)convolution layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu',padding='same'))
```

In [10]:
```
#1)maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [11]:
```
#2)convolution layer
model.add(Convolution2D(32,(3,3),activation='relu',padding='same'))
```

In [12]:
```
#2)maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [13]:
```
#Flatten layer
model.add(Flatten())
```

In [14]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      896

 max_pooling2d (MaxPooling2D  (None, 64, 64, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 64, 64, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 32, 32, 32)       0
 2D)


 flatten (Flatten)           (None, 32768)             0

=================================================================
Total params: 10,144
Trainable params: 10,144
Non-trainable params: 0
_________________________________________________________________
```

In [15]:
```
model.add(Dense(300,activation='relu'))#hidden layer
model.add(Dense(150,activation='relu'))#hidden layer
model.add(Dense(6,activation='softmax'))#output layer
```

In [16]:
```
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=['accuracy'])
```

In [17]:
```
#training
#Note :increse epochs number
model.fit_generator(X_train,validation_data=X_test,epochs=40)
```

```
Epoch 18/40
26/26 [==============================] - 82s 3s/step - loss: 0.6153 - accuracy: 0.7819 - val_loss: 1.0557 - val_accuracy: 0.6
918
Epoch 19/40
26/26 [==============================] - 82s 3s/step - loss: 0.6845 - accuracy: 0.7592 - val_loss: 0.9559 - val_accuracy: 0.7
198
Epoch 20/40
26/26 [==============================] - 81s 3s/step - loss: 0.5983 - accuracy: 0.7847 - val_loss: 0.9378 - val_accuracy: 0.7
134
Epoch 21/40
26/26 [==============================] - 82s 3s/step - loss: 0.4746 - accuracy: 0.8353 - val_loss: 1.0419 - val_accuracy: 0.6
918
Epoch 22/40
26/26 [==============================] - 81s 3s/step - loss: 0.4757 - accuracy: 0.8274 - val_loss: 0.9492 - val_accuracy: 0.7
349
Epoch 25/40
26/26 [==============================] - 82s 3s/step - loss: 0.4064 - accuracy: 0.8573 - val_loss: 1.0956 - val_accuracy: 0.7
026
Epoch 26/40
26/26 [==============================] - 83s 3s/step - loss: 0.4014 - accuracy: 0.8600 - val_loss: 0.9277 - val_accuracy: 0.7
241
Epoch 27/40
26/26 [==============================] - 82s 3s/step - loss: 0.3637 - accuracy: 0.8700 - val_loss: 1.0288 - val_accuracy: 0.6
961
Epoch 28/40
26/26 [==============================] - 76s 3s/step - loss: 0.3375 - accuracy: 0.8840 - val_loss: 1.3130 - val_accuracy: 0.6
573
Epoch 29/40
26/26 [==============================] - 83s 3s/step - loss: 0.3178 - accuracy: 0.8856 - val_loss: 1.1468 - val_accuracy: 0.6
746
Epoch 30/40
26/26 [==============================] - 80s 3s/step - loss: 0.3256 - accuracy: 0.8896 - val_loss: 0.9976 - val_accuracy: 0.7
457
Epoch 31/40
26/26 [==============================] - 79s 3s/step - loss: 0.2053 - accuracy: 0.9314 - val_loss: 1.0972 - val_accuracy: 0.7
306
Epoch 36/40
26/26 [==============================] - 78s 3s/step - loss: 0.2413 - accuracy: 0.9159 - val_loss: 1.2258 - val_accuracy: 0.7
091
Epoch 37/40
26/26 [==============================] - 82s 3s/step - loss: 0.2164 - accuracy: 0.9302 - val_loss: 1.1342 - val_accuracy: 0.7
155
Epoch 38/40
26/26 [==============================] - 81s 3s/step - loss: 0.2174 - accuracy: 0.9262 - val_loss: 1.1446 - val_accuracy: 0.7
435
Epoch 39/40
26/26 [==============================] - 82s 3s/step - loss: 0.2362 - accuracy: 0.9199 - val_loss: 1.2590 - val_accuracy: 0.7
047
Epoch 40/40
26/26 [==============================] - 81s 3s/step - loss: 0.1949 - accuracy: 0.9386 - val_loss: 1.1133 - val_accuracy: 0.7
435
```

Out[17]:
```
<keras.callbacks.History at 0x1e769998fa0>
```

Testing

In [20]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model(r"C:\Users\Dell\Downloads\garbage collection\training\garbage.h5")
```

In [22]:
```
#patch of image you  want to predict
img=image.load_img(r'C:\Users\Dell\Downloads\garbage collection\dataset\Testing\glass\glass422.jpg',target_size=(128,128))
x=image.img_to_array(img)#img to array
x.shape
```

Out[22]:
```
(128, 128, 3)
```

In [23]:
```
import numpy as np
```

In [24]:
```
x=np.expand_dims(x,axis=0)#used for adding one more dimension
x.shape
```

Out[24]:
```
(1, 128, 128, 3)
```

In [25]:
```
prediction=model.predict(x)#instead of predict_classes(x) we can use predict(X)
prediction
```

```
1/1 [==============================] - 1s 534ms/step
```

Out[25]:
```
array([[0., 0., 0., 0., 0., 1.]], dtype=float32)
```

In [26]:
```
prediction = np.argmax(prediction)
print(prediction)
```

```
5
```

In [27]:
```
index=["cardbord","glass","metal","paper","plastic","trash"]
```

In [27]:
```
index=["cardbord","glass","metal","paper","plastic","trash"]
```

In [28]:
```
result=str(index[prediction])
result
```

Out[28]:
```
'trash'
```

5. Project Flow

- User interacts with User interface to upload the image
- The uploaded image is analysed by the model which is integrated
- Once the model analyses the uploaded image, the prediction is showcased on the UI

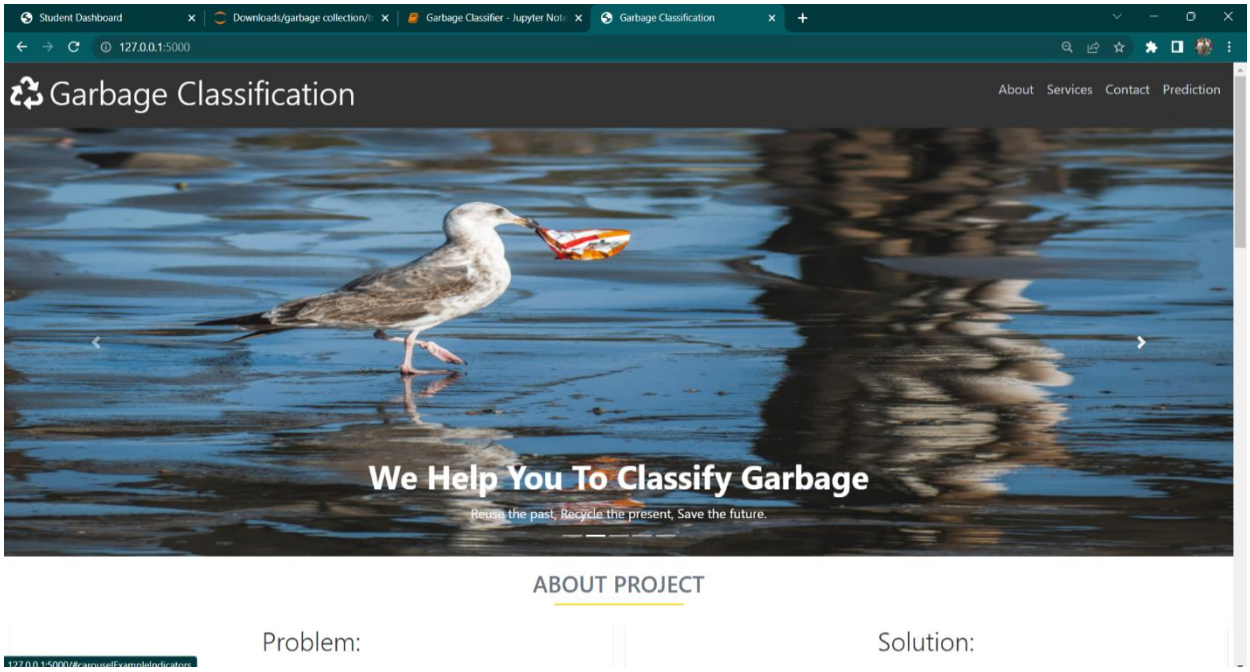
To accomplish this, we have to complete all the activities and tasks listed below

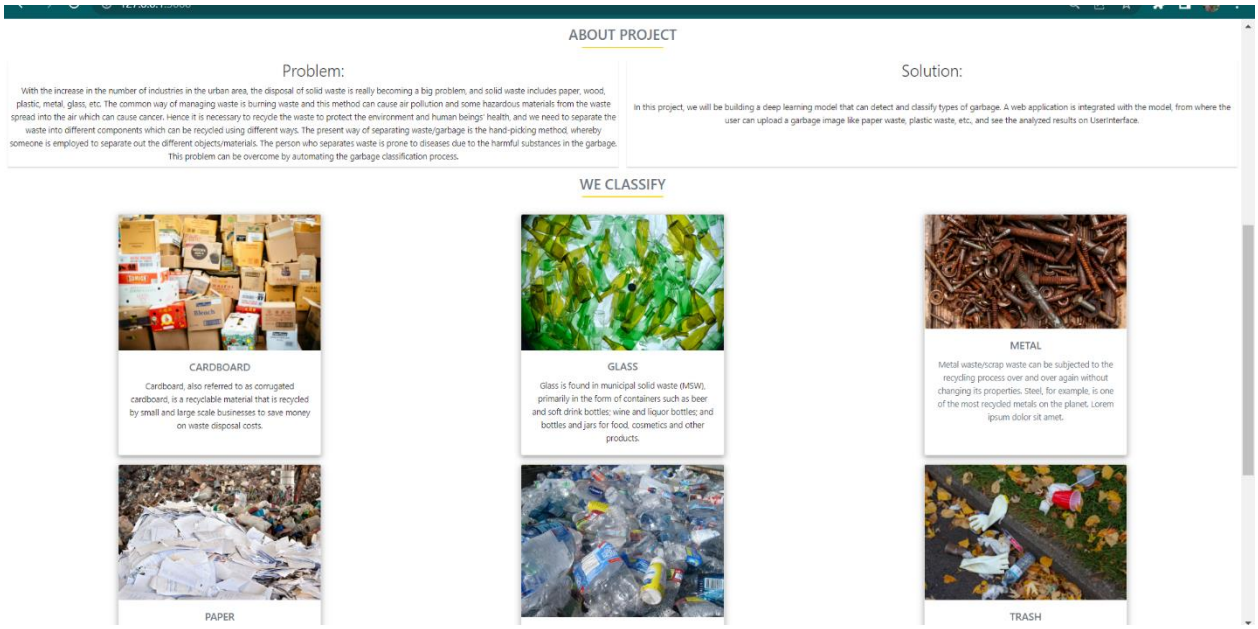
- Data Collection.
  - Collect the dataset or Create the dataset
- Data Preprocessing.
  - Import the ImageDataGenerator library
  - Configure ImageDataGenerator class
  - Apply ImageDataGenerator functionality to Trainset and Testset
- Model Building
  - Import the model building Libraries
  - Initializing the model
  - Adding Input Layer
  - Adding Hidden Layer
  - Adding Output Layer
  - Configure the Learning Process
  - Training and testing the model
  - Optimize the Model
  - Save the Model
- Application Building
  - Create an HTML file
  - Build Python Code

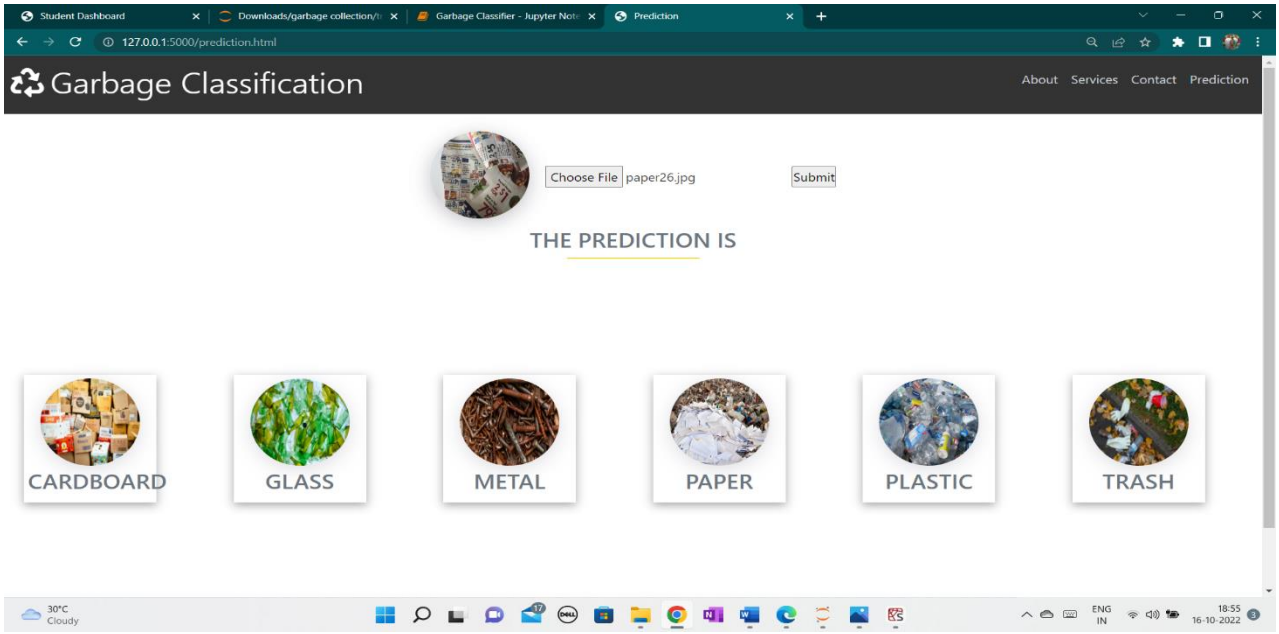
6. Result

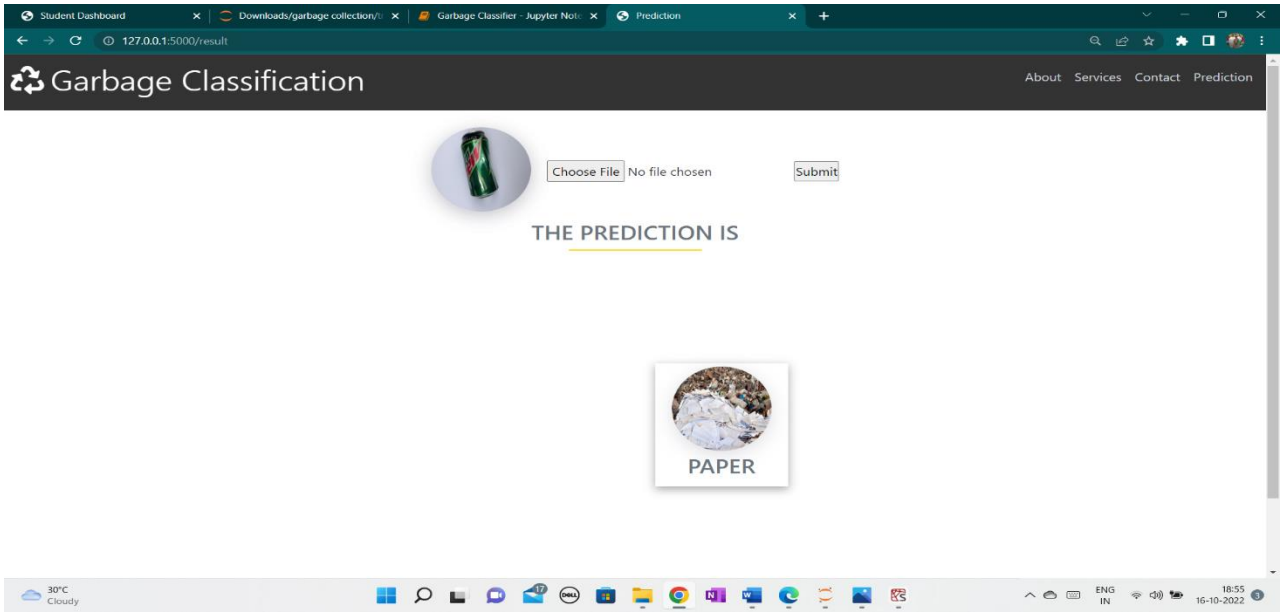
A series of images were used to implement and test the suggested approach. A set of 2508 training images are implemented, while a set of464 images are used for the testing database as soon as the waste is classified on the screen, an equivalent type of waste is displayed.

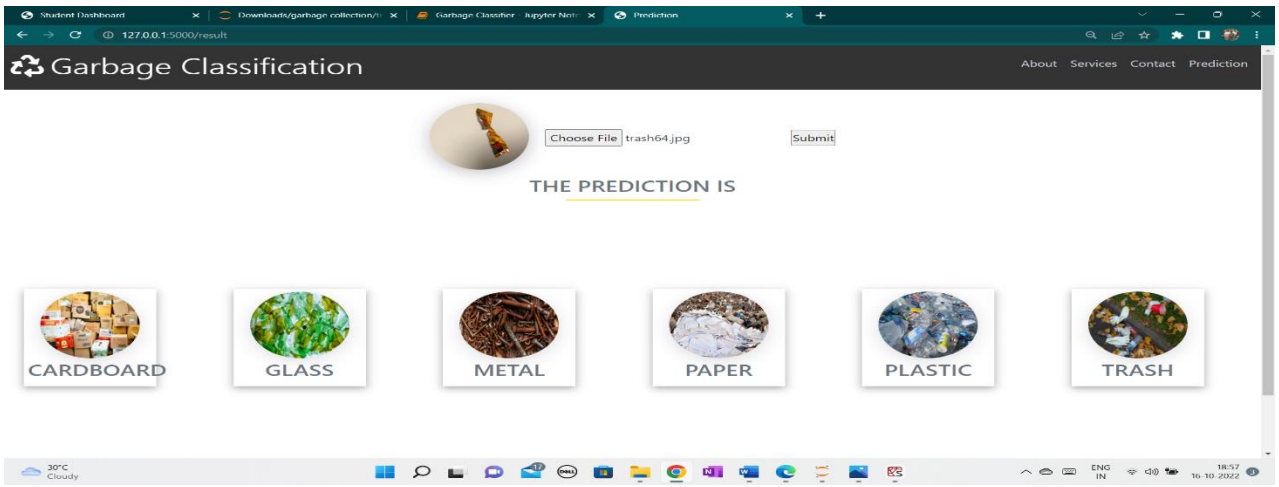
Below are some examples of the output images:

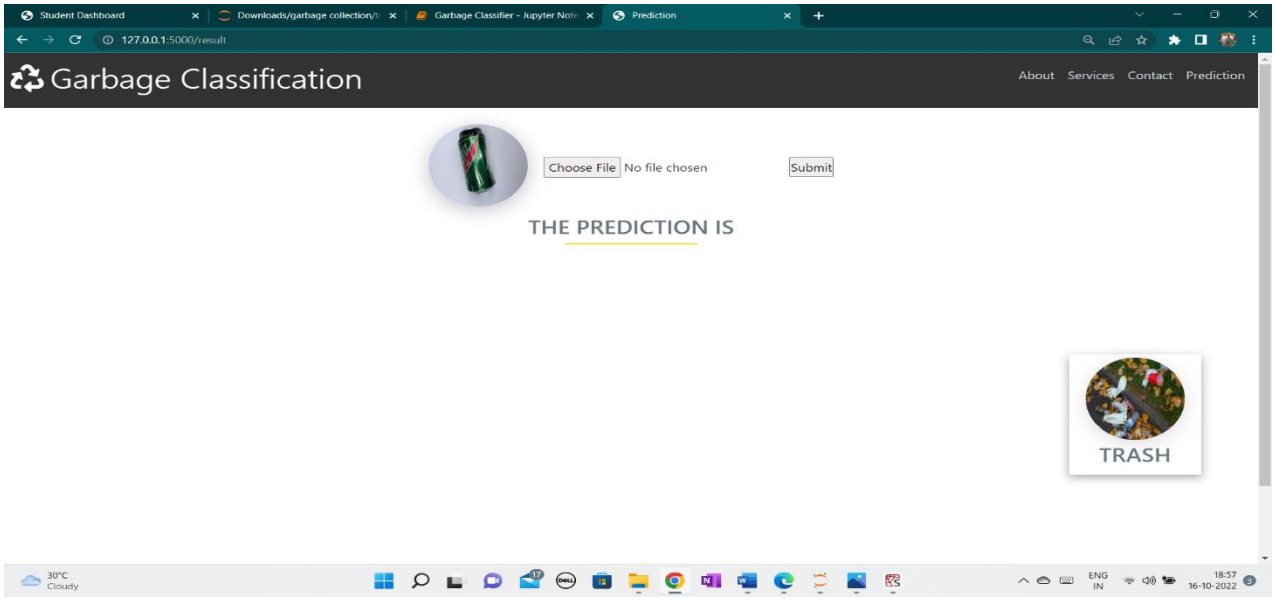

Student Dashboard
Downloads/garbage collection/t
Garbage Classifier - Jupyter Not
Garbage Classification
127.0.0.1:5000
Garbage Classification
About
Services
Contact
Prediction
We Help You To Classify Garbage
Reuse the past, Recycle the present, Save the future.
ABOUT PROJECT
Problem:
Solution:

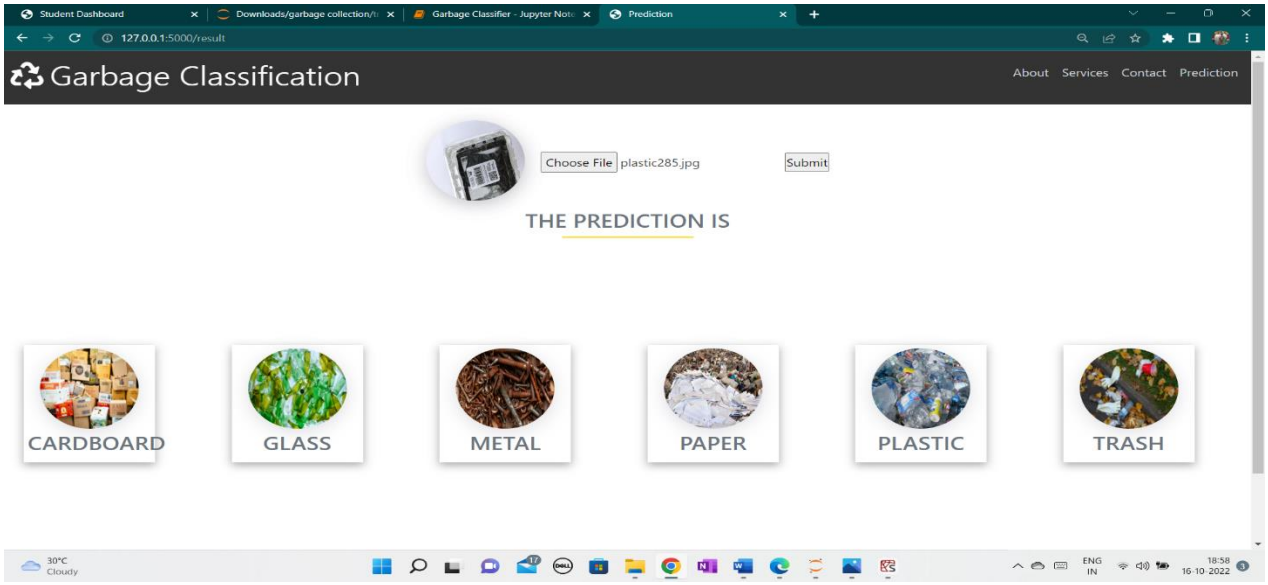

ABOUT PROJECT
Problem:
With the increase in the number of industries in the urban area, the disposal of solid waste is really becoming a big problem, and solid waste includes paper, wood, plastic, metal, glass, etc. The common way of managing waste is burning waste and this method can cause air pollution 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. The present way of separating waste/garbage is the hand-picking method, whereby someone is employed to separate out the different objects/materials. The person who separates waste is prone to diseases due to the harmful substances in the garbage. This problem can be overcome by automating the garbage classification process.
Solution:
In this project, we will be building a deep learning model that can detect and classify types of garbage. A web application is integrated with the model, from where the user can upload a garbage image like paper waste, plastic waste, etc., and see the analyzed results on Userinterface.
WE CLASSIFY
CARDBOARD
Cardboard, also referred to as corrugated cardboard, is a recyclable material that is recycled by small and large scale businesses to save money on waste disposal costs.
GLASS
Glass is found in municipal solid waste (MSW), primarily in the form of containers such as beer and soft drink bottles; wine and liquor bottles; and bottles and jars for food, cosmetics and other products.
METAL
Metal waste/scrap waste can be subjected to the recycling process over and over again without changing its properties. Steel, for example, is one of the most recycled metals on the planet. Lorem ipsum dolor sit amet.
PAPER
TRASH

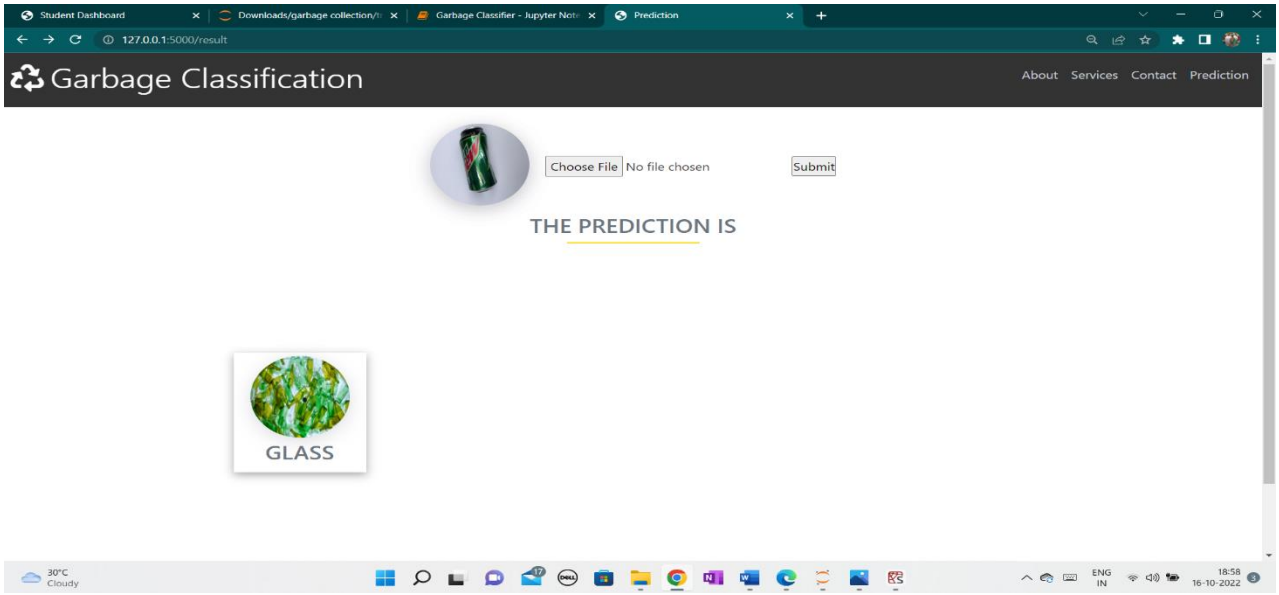

Student Dashboard
Downloads/garbage collection/t
Garbage Classifier - Jupyter Not
Prediction
127.0.0.1:5000/prediction.html
Garbage Classification
About Services Contact Prediction
Choose File paper26.jpg
Submit
THE PREDICTION IS
CARDBOARD
GLASS
METAL
PAPER
PLASTIC
TRASH
30°C Cloudy
ENG IN
18:55 16-10-2022


Student Dashboard
Downloads/garbage collection/t
Garbage Classifier - Jupyter Not
Prediction
127.0.0.1:5000/result
Garbage Classification
About Services Contact Prediction
Choose File No file chosen
Submit
THE PREDICTION IS
PAPER
30°C Cloudy
ENG IN
18:55 16-10-2022


Student Dashboard
Downloads/garbage collection/t
Garbage Classifier - Jupyter Not
Prediction
127.0.0.1:5000/result
Garbage Classification
About Services Contact Prediction
Choose File trash64.jpg
Submit
THE PREDICTION IS
CARDBOARD
GLASS
METAL
PAPER
PLASTIC
TRASH
30°C Cloudy
ENG IN
18:57 16-10-2022


Student Dashboard
Downloads/garbage collection/t
Garbage Classifier - Jupyter Note
Prediction
127.0.0.1:5000/result
Garbage Classification
About
Services
Contact
Prediction
Choose File
No file chosen
Submit
THE PREDICTION IS
TRASH


Student Dashboard
Downloads/garbage collection/t
Garbage Classifier - Jupyter Note
Prediction
127.0.0.1:5000/result
Garbage Classification
About
Services
Contact
Prediction
Choose File
plastic285.jpg
Submit
THE PREDICTION IS
CARDBOARD
GLASS
METAL
PAPER
PLASTIC
TRASH


Student Dashboard
Downloads/garbage collection/t
Garbage Classifier - Jupyter Note
Prediction
127.0.0.1:5000/result
Garbage Classification
About
Services
Contact
Prediction
Choose File
No file chosen
Submit
THE PREDICTION IS
GLASS

7. Advantages and Disadvantages

Advantages:

1. The classification can reduce the amount of garbage disposal and treatment equipment, reduce the treatment cost, reduce the consumption of land resources and have ecological benefits.
2. Segregating your waste allows your business to recycle more items, preventing them from ending up in landfills. This in turn, reduces your overall impact on environment.

Disadvantages:

1. The present model is able to predict only the five types of waste.
2. The quality of the dataset used is not efficient, has a great impact on theaccuracy.
3. Waste management can cause more problems.

8.Applications

1. It will have major role in classification of waste with specially in different types of waste.
2. By using many machine learning and deep learning we can classify waste with different algorithms.

9.Conclusion

In this, we proposed a waste classification system that is able to separate different components of waste using the Machine Learning tools. This system can be used to automatically classify waste and help in reducing human intervention and preventing infection and pollution. The accuracy is around 80%. If more image is added and we should add numbers of cnn layers to the dataset, the system accuracy can be improved.

10.Future Scope

In the future, we will tend to improve our system to be able to categorize more waste items and improve the accuracy by turning some of the parameters used in project.

11. Bibilography

Dataset: https://www.kaggle.com/datasets/arfathbaigs/garbageclassification-final

CNN using Tensorflow: https://www.youtube.com/watch?v=umGJ30-15_A

1. Flask: https://www.youtube.com/watch?v=lj4I_CvBnt0
2. IBM Cloud Account Creation: https://www.youtube.com/watch?v=x6i43M7BAqE
3. CNN Deployment and Download through IBM Cloud: https://www.youtube.com/watch?v=BzouqMGJ41k
4. For information regarding CNN Layers refer to the link
   Link: https://victorzhou.com/blog/intro-to-cnns-part-1/

12. Appendix

Training and Testing the Model

Importing libraries

n [2]:
```
from keras.models import Sequential
from keras.layers import Convolution2D,Flatten,Dense,MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
```

Loading Images

n [3]:
```
train_datagen=ImageDataGenerator(horizontal_flip=True,rescale=1./255,zoom_range=0.2)
#rescale=1./255 means transform every pixel value from range [0,255] -> [0,1].
```

n [6]:
```
test_datagen=ImageDataGenerator(rescale=1./255)
```

[41]:
```
pip install botocore
```

```
Requirement already satisfied: botocore in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.21.41)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from botocore) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from botocore) (1.26.7)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from botocore) (0.10.0)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

[43]:
```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_f6579e33ac5c4a12a6008d664142329b = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='qFVZ2nHK2oxdvBnHw-X31J5aqB0hHJUvx1BkL9UV-60M',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

streaming_body_1 = client_f6579e33ac5c4a12a6008d664142329b.get_object(Bucket='garbageclassification-donotdelete-pr-tkwrf0ckoycis9', Key='dataset.zip')['Body'
```

In [44]:
```
# Unzip the Dataset Zip File
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()), 'r')
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

In [45]:
```
%%bash
ls dataset
```

```
Testing
Training
```

In [46]:
```
X_train=train_datagen.flow_from_directory(r'/home/wsuser/work/dataset/Training',target_size=(128,128),class_mode='categorical',batch_size=100)
```

```
Found 2508 images belonging to 6 classes.
```

In [47]:
```
X_test=test_datagen.flow_from_directory(r'/home/wsuser/work/dataset/Testing',target_size=(128,128),class_mode='categorical',batch_size=100)
```

```
Found 464 images belonging to 6 classes.
```

In [48]:
```
X_train.class_indices
```

Out[48]:
```
{'cardboard': 0, 'glass': 1, 'metal': 2, 'paper': 3, 'plastic': 4, 'trash': 5}
```

ModelBuilding

In [49]:
```
model=Sequential()
```

In [50]:
```
#1)convolution layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu',padding='same'))
```

In [51]:
```
#1)maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [52]:
```
#2)convolution layer
model.add(Convolution2D(32,(3,3),activation='relu',padding='same'))
```

In [53]:
```
#2)maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
```



```
 conv2d_2 (Conv2D)           (None, 128, 128, 32)      896

 max_pooling2d_2 (MaxPooling  (None, 64, 64, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 64, 64, 32)        9248

 max_pooling2d_3 (MaxPooling  (None, 32, 32, 32)       0
 2D)

 flatten_1 (Flatten)         (None, 32768)             0

=================================================================
Total params: 10,144
Trainable params: 10,144
Non-trainable params: 0
_________________________________________________________________
```

In [56]:
```
model.add(Dense(300,activation='relu'))#hidden layer
model.add(Dense(150,activation='relu'))#hidden layer
model.add(Dense(6,activation='softmax'))#output layer
```

In [57]:
```
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=['accuracy'])
```

In [58]:
```
#training
#Note :increse epochs number
model.fit_generator(X_train,validation_data=X_test,epochs=30)
```

```
/tmp/wsuser/ipykernel_164/710134095.py:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(X_train,validation_data=X_test,epochs=30)
Epoch 1/30
26/26 [==============================] - 44s 2s/step - loss: 1.8770 - accuracy: 0.1918 - val_loss: 1.7745 - val_accuracy: 0.2134
Epoch 2/30
26/26 [==============================] - 43s 2s/step - loss: 1.6972 - accuracy: 0.2727 - val_loss: 1.5557 - val_accuracy: 0.3815
Epoch 3/30
26/26 [==============================] - 44s 2s/step - loss: 1.5403 - accuracy: 0.3589 - val_loss: 1.4681 - val_accuracy: 0.4375
Epoch 4/30
26/26 [==============================] - 43s 2s/step - loss: 1.4634 - accuracy: 0.4258 - val_loss: 1.5116 - val_accuracy: 0.4310
Epoch 5/30
26/26 [==============================] - 44s 2s/step - loss: 1.4065 - accuracy: 0.4422 - val_loss: 1.4690 - val_accuracy: 0.4353
Epoch 6/30
26/26 [==============================] - 43s 2s/step - loss: 1.3127 - accuracy: 0.4976 - val_loss: 1.3216 - val_accuracy: 0.4871
Epoch 7/30
```

```
Epoch 20/30
26/26 [==============================] - 44s 2s/step - loss: 0.5956 - accuracy: 0.7927 - val_loss: 1.0546 - val_accuracy: 0.6638
Epoch 21/30
26/26 [==============================] - 44s 2s/step - loss: 0.6017 - accuracy: 0.7895 - val_loss: 1.0662 - val_accuracy: 0.6616
Epoch 22/30
26/26 [==============================] - 44s 2s/step - loss: 0.5812 - accuracy: 0.7899 - val_loss: 1.1376 - val_accuracy: 0.6358
Epoch 23/30
26/26 [==============================] - 44s 2s/step - loss: 0.5647 - accuracy: 0.7931 - val_loss: 1.0813 - val_accuracy: 0.6595
Epoch 24/30
26/26 [==============================] - 44s 2s/step - loss: 0.5177 - accuracy: 0.8134 - val_loss: 1.0607 - val_accuracy: 0.6638
Epoch 25/30
26/26 [==============================] - 44s 2s/step - loss: 0.4547 - accuracy: 0.8405 - val_loss: 1.1465 - val_accuracy: 0.6681
Epoch 26/30
26/26 [==============================] - 44s 2s/step - loss: 0.5173 - accuracy: 0.8154 - val_loss: 1.0151 - val_accuracy: 0.6810
Epoch 27/30
26/26 [==============================] - 44s 2s/step - loss: 0.4096 - accuracy: 0.8561 - val_loss: 1.0614 - val_accuracy: 0.6918
Epoch 28/30
26/26 [==============================] - 44s 2s/step - loss: 0.3825 - accuracy: 0.8593 - val_loss: 1.1918 - val_accuracy: 0.6509
Epoch 29/30
26/26 [==============================] - 44s 2s/step - loss: 0.3502 - accuracy: 0.8832 - val_loss: 1.0228 - val_accuracy: 0.7155
Epoch 30/30
26/26 [==============================] - 44s 2s/step - loss: 0.3176 - accuracy: 0.8880 - val_loss: 1.0528 - val_accuracy: 0.7263
```

Out[58]:
```
<keras.callbacks.History at 0x7fbc0792bdf0>
```

In [59]:
```
model.save('garbage2.h5')
```

In [62]:
```
# Convert the Saved Model to a Tar Compressed Format
!tar -zcvf IBM_TrainedModel.tgz garbage2.h5
```

```
garbage2.h5
```

In [60]:
```
import numpy as np
from keras.preprocessing import image
```

In [63]:
```
ls -1
```

```
dataset/
garbage1.h5
garbage2.h5
IBM_TrainedModel.tgz
```

Testing

[64]:
```
!pip install watson-machine-learning-client --upgrade
```

```
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
     |████████████████████████████████| 538 kB 22.8 MB/s eta 0:00:01
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.18.21)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2022.6.15)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (0.10.0)
Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (1.21.41)
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (0.5.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from botocore<1.22.0,>=1.21.21->boto3->watson-machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.22.0,>=1.21.21->boto3->watson-machine-learning-client) (1.15.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client) (2.0.4)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (1.20.3)
Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391
```

[65]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "z44AWaH210GSLg3U_HWGXRyLhvUZXHZ_Z20LF47Hojlk"
}

client = APIClient(wml_credentials)
```

[66]:
```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return (next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

In [66]:
```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return (next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

In [67]:
```
space_uid = guid_from_space_name(client, 'garbagee')
print("Space UID : ", space_uid)
```

```
Space UID :  526f9115-5524-4d40-a9ec-104503f3f081
```

In [68]:
```
client.set.default_space(space_uid)
```

Out[68]:
```
'SUCCESS'
```

In [69]:
```
client.software_specifications.list()
```

```
------------------------------  ------------------------------------  ----
NAME                            ASSET_ID                              TYPE
default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12       020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9       0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6           0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                      0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod    1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6               10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl       111e41b3-de2d-5422-a4d6-bf776828c4b7  base
runtime-22.1-py3.9              12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6         154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                    1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6          1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
pytorch-onnx_rt22.1-py3.9-edt   1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6            1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
```

In [70]:
```
software_spec_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
software_spec_uid
```

Out[70]:
```
'acd9c798-6974-5d2f-a657-ce06e986df4d'
```

In [ ]:
```
model_details = client.repository.store_model(model='IBM_TrainedModel.tgz', meta_props={
    client.repository.ModelMetaNames.NAME: "CNN",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid,
    client.repository.ModelMetaNames.TYPE: "tensorflow_2.7"})
model_id = client.repository.get_model_uid(model_details)
```

In [114…
```
model_id
```

Out[114]:
```
'4568980b-9ec4-42f3-b274-770ee65a5772'
```

In [75]:
```
client.repository.download(model_id,'my_model.tar.gz')
```

```
Successfully saved model content to file: 'my_model.tar.gz'
```

Out[75]:
```
'/home/wsuser/work/my_model.tar.gz'
```

In [78]:
```
streaming_body_2 = client_f6579e33ac5c4a12a6008d664142329b.get_object(Bucket='garbageclassification-donotdelete-pr-tkwrf0ckoycis9', Key='glass401.jpg')['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

In [87]:
```
pwd
```

Out[87]:
```
'/home/wsuser/work'
```

In [104…
```
img=image.load_img(r'/home/wsuser/work/dataset/Testing/glass/glass401.jpg',target_size=(128,128))
```

In [105…
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [106…
```
model=load_model("garbage2.h5")
```

Downloading trained model from IBM Cloud

Downloading From IBM

Connecting to IBM Cloud Storage to Get Model from Deployment

In [1]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "z44AWaH210GSLg3U_HWGXRyLhvUZXHZ_Z20LF47HojlK"
}

client = APIClient(wml_credentials)
```

In [2]:
```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return (next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

In [3]:
```
space_uid = guid_from_space_name(client, 'garbagee')
print("Space UID : ", space_uid)
```

```
Space UID :  526f9115-5524-4d40-a9ec-104503f3f081
```

In [4]:
```
client.set.default_space(space_uid)
```

Out[4]:
```
'SUCCESS'
```

In [5]:
```
client.repository.download("4568980b-9ec4-42f3-b274-770ee65a5772","IBM_Modell_Download.tar3.gz")
```

```
Successfully saved model content to file: 'IBM_Modell_Download.tar3.gz'
```

Out[5]:
```
'C:\\Users\\Dell\\Downloads\\garbage collection\\IBM_deployment/IBM_Modell_Download.tar3.gz'
```