!pip list | grep tensorflow

!git clone --depth 1 https://github.com/tensorflow/models

%%bash

apt-get install -y protobuf-compiler

%%bash

cd models/research

protoc object\_detection/protos/\*.proto --python\_out=.

cp object\_detection/packages/tf2/setup.py .

python -m pip install .

!pip list | grep tensorflow

!pip install tensorflow==2.10.0

import os

import pathlib

import cv2

import logging

logging.disable(logging.WARNING)

import matplotlib

import matplotlib.pyplot as plt

import numpy as np

from six import BytesIO

from PIL import Image

from six.moves.urllib.request import urlopen

import tensorflow as tf

import tensorflow\_hub as hub

tf.get\_logger().setLevel('ERROR')

from object\_detection.utils import label\_map\_util

from object\_detection.utils import visualization\_utils as viz\_utils

from object\_detection.utils import ops as utils\_ops

%matplotlib inline

!wget https://storage.googleapis.com/tf\_model\_garden/vision/waste\_identification\_ml/material\_model.zip

!wget https://storage.googleapis.com/tf\_model\_garden/vision/waste\_identification\_ml/material\_form\_model.zip

#!wget https://storage.googleapis.com/tf\_model\_garden/vision/waste\_identification\_ml/plastic\_types\_model.zip

%%bash

mkdir material material\_form

unzip material\_model.zip -d material/

unzip material\_form\_model.zip -d material\_form/

#unzip plastic\_types\_model.zip -d plastic\_type/

ALL\_MODELS ={

'material\_model' : 'material/saved\_model/saved\_model',

'material\_form\_model': 'material\_form/saved\_model/saved\_model/',

#'plastic\_model':'plastic\_type/saved\_model/saved\_model/'

}

IMAGES\_FOR\_TEST ={

#'Image1':'models/official/projects/waste\_identification\_ml/pre\_processing/config/sample\_images/image2\_.png''

'Image1':''

}

def normalize\_image(image,

offset=(0.485, 0.456, 0.406),

scale=(0.229, 0.224, 0.225)):

with tf.name\_scope('normalize\_image'):

image=tf.image.convert\_image\_dtype(image, dtype=tf.float32)

offset = tf.constant(offset)

offset= tf.expand\_dims(offset, axis=0)

offset= tf.expand\_dims(offset, axis=0)

image-=offset

scale= tf.constant(scale)

scale=tf.expand\_dims(scale, axis=0)

scale=tf.expand\_dims(scale, axis=0)

image/=scale

return image

def load\_image\_into\_numpy\_array(path):

image=None

if(path.startswith('http')):

response= urlopen(path)

image\_data =response.read()

image\_data=BytesIO(image\_data)

image =Image.open(image\_data)

else:

image\_data = tf.io.gfile.GFile(path, 'rb').read()

image=Image.open(BytesIO(image\_data))

(im\_width, im\_height) =image.size

return np.array(image.getdata()).reshape((1, im\_height, im\_width, 3)).astype(np.uint8)

def build\_inputs\_for\_segmentation(image):

image= normalize\_image(image)

return image

# @title Model Selection { display-mode: "form", run: "auto" }

model\_display\_name='material\_model' #param ['material\_model", model\_handle= ALL\_MODELS[model\_display\_name]

print('Selected model: '+ model\_display\_name)

model\_handle = ALL\_MODELS[model\_display\_name]

print('Model Handle at TensorFlow Hub: {}'.format(model\_handle))

if model\_display\_name=='material\_model':

PATH\_TO\_LABELS ='./models/official/projects/waste\_identification\_ml/pre\_processing/config/data/material\_labels.pbtxt'

elif model\_display\_name == 'material\_form\_model':

PATH\_TO\_LABELS='./models/official/projects/waste\_identification\_ml/pre\_processing/config/data/material\_form\_labels.pbtxt'

#elif model\_display\_name =='plastic\_model':

#PATH\_TO\_LABELS ='./models/official/projects/waste\_identification\_ml/pre\_processing/config/data/plastic\_type\_labels.pbtxt'

print('Labels selected for',model\_display\_name)

print('\n')

category\_index =label\_map\_util.create\_category\_index\_from\_labelmap(PATH\_TO\_LABELS, use\_display\_name=True)

category\_index

labels={1: {'id': 1, 'name': 'Soil'},

2: {'id': 2, 'name': 'clothes'},

3: {'id': 3,

'name': 'Rubber and leather products'},

4: {'id': 4,

'name': 'Wood products '},

5: {'id': 5,

'name': ' food waste'},

6: {'id': 6,

'name': 'Plastic '},

7: {'id': 7,

'name': 'bio waste'},

8: {'id': 8,

'name': 'Cardboard products'},

9: {'id': 9,

'name': 'Glass products '},

10: {'id': 10,

'name': 'metal waste'}}

# Print the updated label dictionary

print(labels)

category\_index.update(labels)

print(category\_index)

import tensorflow\_hub as hub

print('loading model...')

hub\_model = hub.load(model\_handle)

print('model loaded!')

#@title Image Selection

selected\_image = 'Image1'

image\_path=IMAGES\_FOR\_TEST[selected\_image]

image\_np= load\_image\_into\_numpy\_array(image\_path)

print('min:', np.min(image\_np[0]), 'max:', np.max(image\_np[0]))

plt.figure(figsize=(10,10))

plt.imshow(image\_np[0])

plt.show()

hub\_model\_fn = hub\_model.signatures["serving\_default"]

height = hub\_model\_fn.structured\_input\_signature[1]['inputs'].shape[1]

width = hub\_model\_fn.structured\_input\_signature[1]['inputs'].shape[2]

input\_size=(height,width)

print(input\_size)

!pip install tensorflow-addons

image\_np\_cp = cv2.resize(image\_np[0],input\_size[::-1])

image\_np=build\_inputs\_for\_segmentation(image\_np\_cp)

image\_np = tf.expand\_dims(image\_np,axis=0)

image\_np.get\_shape()

plt.figure(figsize=(10,10))

plt.imshow(image\_np[0])

plt.show()

results = hub\_model\_fn(image\_np)

result = {keys:value.numpy() for keys,value in results.items()}

print(result.keys())

label\_id\_offset =0

min\_score\_thresh =0.6

use\_normalized\_coordinates=True

if use\_normalized\_coordinates:

result['detection\_boxes'][0][:,[0,2]]/=height

result['detection\_boxes'][0][:,[1,3]]/=width

if 'detection\_masks' in result:

detection\_masks = tf.convert\_to\_tensor(results['detection\_masks'][0])

detection\_boxes = tf.convert\_to\_tensor(results['detection\_boxes'][0])

detection\_masks\_reframed = utils\_ops.reframe\_box\_masks\_to\_image\_masks(

detection\_masks,detection\_boxes,

image\_np.shape[1],image\_np.shape[2])

detection\_masks\_reframed=tf.cast(detection\_masks\_reframed>0.5,np.uint8)

result['detection\_mask\_reframed']=detection\_masks\_reframed.numpy()

viz\_utils.visualize\_boxes\_and\_labels\_on\_image\_array(

image\_np\_cp,

result['detection\_boxes'][0],

(result['detection\_classes'][0]+label\_id\_offset).astype(int),

result['detection\_scores'][0],

category\_index=category\_index,

use\_normalized\_coordinates=use\_normalized\_coordinates,

max\_boxes\_to\_draw=200,

min\_score\_thresh = min\_score\_thresh,

agnostic\_mode =False,

instance\_masks = result.get('detection\_masks\_reframed',None),

line\_thickness=5

)

plt.figure(figsize=(10,10))

plt.imshow(image\_np\_cp)

plt.show()

recycling\_methods = {

1: {'id': 1, 'name': 'Soil', 'method': """Soil recycling involves various techniques such as composting, mulching, and soil stabilization.

Composting converts organic materials into nutrient-rich soil amendments,

while mulching helps to retain moisture and suppress weeds.

Soil stabilization methods, such as using lime or cement, improve the structural integrity of soil for construction purposes."""},

2: {'id': 2, 'name': 'clothes', 'method': """Clothes recycling can involve several approaches.

Textiles can be sorted and cleaned for reuse in second-hand stores or donated to those in need.

Additionally, fabrics can be shredded and recycled into new materials such as insulation, carpet padding, or industrial rags.

Some innovative techniques even allow for the breakdown of certain fabrics to create new fibers for textiles."""},

3: {'id': 3, 'name': 'Rubber and leather products', 'method': """Recycling rubber and leather products typically involves shredding or grinding the materials into small pieces,

which can then be used for various applications.

Ground rubber can be used in asphalt paving, playground surfaces,or molded into new rubber products.

Leather scraps can be processed into composite materials or used for smaller goods such as wallets and belts."""},

4: {'id': 4, 'name': 'Wood products ', 'method': """Wood recycling encompasses processes like chipping, grinding, and shredding.

Recovered wood can be used for various purposes,

including mulch production, biomass energy generation,

or manufacturing composite wood products such as particleboard and engineered lumber.

Additionally, some wood can be reclaimed for architectural salvage or crafted into new furniture and decorative items."""},

5: {'id': 5, 'name': 'food waste', 'method': """Food waste recycling involves composting organic materials to create nutrient-rich soil amendments.

Through composting, food scraps, yard waste, and other organic matter break down into humus,

which can enrich soil and support plant growth.

Industrial-scale composting facilities often process large volumes of food waste,

producing compost for agricultural and landscaping use."""},

6: {'id': 6, 'name': 'Plastic ', 'method': """Plastic recycling typically involves sorting, cleaning, shredding, melting, and pelletizing.

Sorted plastic waste is cleaned to remove contaminants before being shredded into small pieces or melted down.

The melted plastic can then be formed into pellets for manufacturing new plastic products.

Advanced recycling technologies, such as chemical or enzymatic recycling,

are also emerging to break down plastics into their molecular components for reuse."""},

7: {'id': 7, 'name': 'bio waste', 'method': """Bio waste, also known as organic waste, can be recycled through composting or anaerobic digestion.

Composting involves the decomposition of organic materials in the presence of oxygen,

producing compost that can be used as a soil amendment.

Anaerobic digestion breaks down organic waste in the absence of oxygen,

generating biogas and nutrient-rich digestate.

Biogas can be used for energy production, while the digestate can be used as fertilizer."""},

8: {'id': 8, 'name': 'Cardboard products', 'method': """Cardboard recycling typically involves collection, sorting, and processing.

Once collected, cardboard is sorted based on its grade and quality.

The sorted cardboard is then shredded, pulped, and mixed with water to create a slurry.

This slurry is then formed into new sheets of cardboard through a process of pressing and drying.

Recycled cardboard can be used to manufacture packaging materials, paperboard, or even new cardboard boxes."""},

9: {'id': 9, 'name': 'Glass products ', 'method': """Glass recycling involves collecting, sorting, cleaning, and melting glass containers and other glass items.

After collection, glass is sorted by color and processed to remove any contaminants.

Clean glass is then crushed and melted in a furnace to produce molten glass, which can be molded or blown into new glass products.

Recycled glass can be used to make new containers, fiberglass insulation, or decorative glassware."""},

10: {'id': 10, 'name': 'metal waste', 'method': """Metal recycling typically involves collection, sorting, processing, and melting.

Scrap metal is collected from various sources, including household appliances, vehicles, and industrial equipment.

After collection, metal is sorted by type and quality before being processed.

Processing may involve shredding, shearing, or compacting the metal to prepare it for melting.

Once melted, the metal can be formed into ingots or cast into new products.

Recycling metal reduces the need for virgin ore extraction and conserves energy."""}

}

#print((result['detection\_classes'][0]+label\_id\_offset).astype(int))

# Assuming result['detection\_classes'] contains the detected class IDs as a NumPy array

detected\_class\_id = int(result['detection\_classes'][0][0]) # Take the first element of the array and convert to integer

category\_name = category\_index[detected\_class\_id] # Look up category name

print("Detected category:", category\_name)

str = recycling\_methods[category\_name['id']]['method']

print(str)

# Assuming result['detection\_classes'][0] contains the detected class IDs as a NumPy array

detected\_class\_ids = result['detection\_classes'][0]

# Convert the detected class IDs to integers

detected\_class\_ids = detected\_class\_ids.astype(int)

# Assuming 'label\_id\_offset' is already defined earlier in your code

label\_id\_offset = 0

# Add the label ID offset if necessary (this depends on how your model is trained)

# Note: You may need to adjust this if your model's label IDs start from 1 instead of 0

detected\_class\_ids += label\_id\_offset

# Now you have the detected class IDs for the test image

print("Detected class IDs:", detected\_class\_ids)

# Assuming result['detection\_classes'] contains the predicted class IDs as a NumPy array

predicted\_classes = result['detection\_classes'][0].astype(int) # Extract and convert to integers

# Model predictions (class IDs) for test images

# Replace the example predictions with your actual model predictions

model\_predictions = set(predicted\_classes.tolist()) # Assuming you stored the predictions in `predicted\_classes`

print( model\_predictions)

# Convert image path to numpy array

image\_np = load\_image\_into\_numpy\_array(image\_path)

# Preprocess the image for segmentation

image\_np\_cp = cv2.resize(image\_np[0], input\_size[::-1])

image\_np = build\_inputs\_for\_segmentation(image\_np\_cp)

image\_np = tf.expand\_dims(image\_np, axis=0)

# Get model predictions

results = hub\_model\_fn(image\_np)

result = {keys: value.numpy() for keys, value in results.items()}

# Define the ground truth label

ground\_truth\_label = 1 # Assuming the image contains waste

# Define threshold for detection score

min\_score\_thresh = 0.8

# Compute model predictions

detection\_score = result['detection\_scores'][0][0]

predicted\_label = 1 if detection\_score >= min\_score\_thresh else 0

# Compute evaluation metrics

if ground\_truth\_label == 1:

if predicted\_label == 1:

true\_positive = 1

false\_negative = 0

else:

true\_positive = 0

false\_negative = 1

else:

if predicted\_label == 1:

true\_positive = 0

false\_negative = 0

else:

true\_positive = 1

false\_negative = 0

# Compute precision, recall, and F1 score

precision = true\_positive / (true\_positive + (1e-9)) # Avoid division by zero

recall = true\_positive / (true\_positive + false\_negative)

f1\_score = 2 \* (precision \* recall) / (precision + recall + (1e-9)) # Avoid division by zero

print("Precision:", precision)

print("Recall:", recall)

print("F1 Score:", f1\_score)