COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COCHIN UNIVERSITY COLLEGE OF ENGINEERING

KUTTANADU

APRIL 2021

PROJECT REPORT ON

Multi Label classification

Submitted on partial fulfilment of the requirement for the award of the degree in Master of Computer Applications from COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Submitted by

Pragati(38119225)

Aman sagar(38119207)

Rahul kumar(38119227)

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COCHIN UNIVERSITY COLLEGE OF ENGINEERING

KUTTANADU

CERTIFICATE

This is to certify that this project report entitled "**Multi Label Classification**" is a bona fide record on partial fulfilment for the Degree of the Master of Computer Applications to the COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY through DEPARTMENT OF COMPUTER APPLICATIONS, COCHIN UNIVERSITY COLLEGE OF ENGINEERING KUTTANADU, ALAPPUZHA done by **Pragati *(38119225), Aman sagar (Reg NO:38119207)*** *and* ***Rahul kumar (38119227)***in the year 2021.

Project Guide : Fanny May Joseph

Head of the Department : ***Mr, HARIKRISHNAN D***

Internal Examiner

DECLARATION

We hereby declare that the project entitled “**Multi Label Classification”** submitted to the DEPARTMENT OF COMPUTER APPLICATIONS, COCHIN UNIVERSITY COLLEGE OF ENGINEERING, KUTTANADU in the partial fulfilment of the requirements for the award of Degree in MASTER OF COMPUTER APPLICATIONS is a record of original work done by us under the guidance of ***Mrs. Fanny May joseph***, Assistant Professor in MCA Department during my period of study in COCHIN UNIVERSITY COLLEGE OF ENGINEERING, KUTTANADU.

Place : CUCEK Pragati

Date : Aman sagar

Rahul kumar

ACKNOWLEDGEMENT

We are thankful to god almighty for the blessings in the successful completion of our mini project **“Multi label classification”**. We would like to record my profound gratitude to **Dr. Joseph Kutty Jacob**, Principal and ***Mr, HARIKRISHNAN D***, Head of the Department, MCA, COCHIN UNIVERSITY COLLEGE OF ENGINEERING who has deeply inspired us to do our project.

It’s grateful to express our thanks to **Mrs. Fanny May Joseph** Assistant Professor in MCA Department our Project guide, **COCHIN UNIVERSITY COLLEGE OF ENGINEERING, KUTTANADU**, because her effective guidance, constructive criticism and innovative and useful stream of suggestions that helped us to complete our project.

We are thankful to various resources that provide requirements for our projects, because requirements are backbone of every project.

We are also thankful to our teachers, friends, family members, for their support and prayer for us to complete our project.

SYNOPSIS

Every image in the real world contains objects with multiple labels, there is a need to develop systems that can accurately identify and label multiple attributes within the image. This process, called multi-label classification.

This is the forms the foundation for this project. Classification is a method to extract information from data sets. This is done by dividing the data into categories based on some features.

The idea is to derive a model which can perform the sorting process by training it on data objects where the category, or label, is known.

In this project, system will take input as a image, then it will classify that which type of automobile is this, and what is the colour of this image.

We have trained a model to classify objects based on two attributes: colour and the type of automobile present in the image.

CONTENTS

INTRODUCTION

1.1 ABOUT THE PROJECT

A machine is said to learn from an experience E, if its performance P, for a set of tasks T, improves after the experience E. This project aims to develop a neural network that is capable of identifying the various attributes of the objects in an image, and present a confidence rating for the same.

Classification is a method to extract information from data sets. This is done by dividing the data into categories based on some features. The idea is to derive a model which can perform the sorting process by training it on data objects where the category, or label, is known. The model should then be able to classify unlabeled data with sufficient accuracy. However, since almost every image in the real world contains objects with multiple labels, there is a need to develop systems that can accurately identify and label multiple attributes within the image. This process, called multi-label classification, forms the foundation for this project.

As a proof-of-concept, we have trained a neural network to classify objects based on two attributes: color and the type of automobile present in the image. A simple neural network model, which consists of multiple alternating layers of convolutional and max pooling layers, is implemented in this project, all of which are ultimately combined into a dense layer.

LITERATURE REVIEW

Multi-label Learning is a form of supervised learning where the classification algorithm is required to learn from a set of instances, each instance can belong to multiple classes and so after be able to predict a set of class labels for a new instance. This is a generalized version of most popular multi-class problems where each instances is restricted to have only one class label. There exists a wide range of applications for multi-labelled predictions, such as text categorization, semantic image labeling, gene functionality classification etc. and the scope and interest is increasing with modern applications.

METHODOLOGY

3.1 PROCESS FLOW

Machine learning is a field in computer science aiming to imitate the human learning process. For image recognition problems, convolutional neural networks, or CNNs, are a common choice. A common choice for the structure of a CNN consists of several alternating layers of convolutional and max-pooling layers, which are then combined in a dense layer. A dataset to train the neural network is passed as input to the CNN. Once trained, a model is created, which can be archived for future use. A subset of the training data is also passed to the model, to test the efficiency of the learning process. Normally, 88% of the initial dataset is used to train the model and the remaining 12% is used to test it. Once properly trained and tested, the model can be passed an input that it has never encountered before, and still produce the expected or desired result. A typical neural network is as shown below:

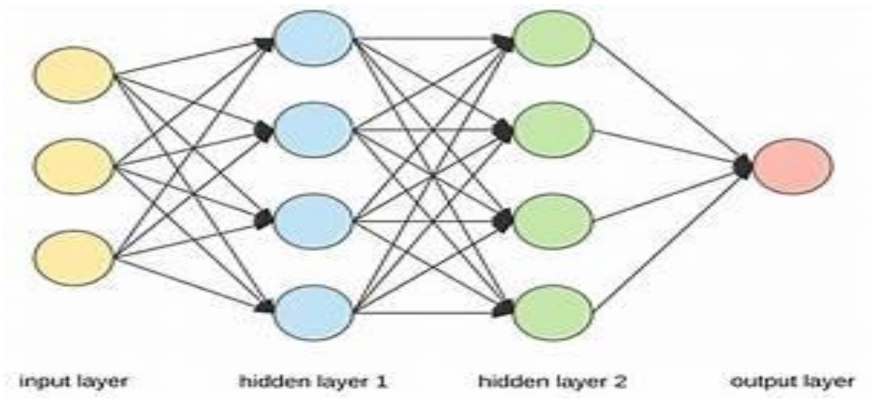


Fig. 1: Simplified representation of a neural network

3.1.1 CONVOLUTIONAL NEURAL NETWORKS

The convolutional layers work as a feature extractor, or a filter, and extract some sort of characteristic from the input data. Usually, one convolutional layer has several filters which are all applied in the same step, but extract different features. The size of the filter, or the kernel, depends on what the size of a specific feature is expected to be. After the convolution is done, some kind of non-linearity is often applied. The most common choices include the sigmoid function and the ReLU function, used in our project. ReLU is defined as Department of Computer Applications 6 Multi-label Image Classification Using CNN f (x)=max(0 , x), thresholding the input at zero. ReLU has become a common choice, as it can help to increase the convergence of the gradient descent optimization method. When a feature has been extracted from the image, we can reduce the spatial size of the image. This will keep the information of which features are present and their relative positions, but in a lower resolution. This process is called subsampling and is done by a pooling layer. In this case we are using a max pooling layer, which is a common choice for convolutional neural networks. This specific king of pooling means that from the pooling kernel, only the pixel with the largest intensity value will be transferred into the subsampled image.

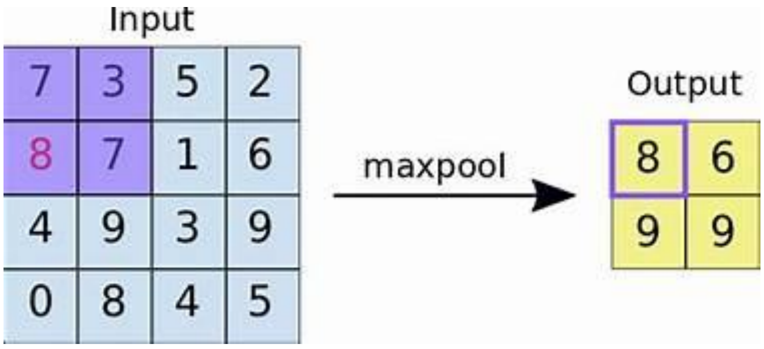


Fig. 2: Max pooling of image data

The CNN architecture used is The CNN architecture used is Smaller VGGNet, a simplified version of VGG Net (Visual Geometry Group). The VGG Net model was first introduced by Simonyan and Zisserman in their 2014 paper, Very Deep Convolutional Networks for Large Scale Image Recognition. Smaller VGGNet, a simplified version of VGG Net (Visual Geometry Group). The VGG Net model was first introduced by Simonyan and Zisserman in their 2014 paper, Very Deep Convolutional Networks for Large Scale Image Recognition.

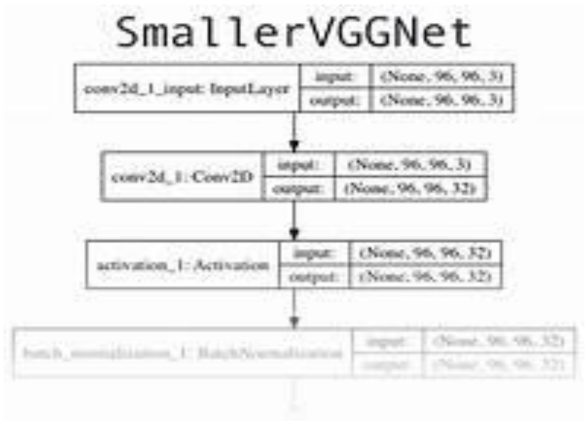

SmallerVGGNet
conv2d_1_input: InputLayer
input: (None, 96, 96, 3)
output: (None, 96, 96, 3)
conv2d_1: Conv2D
input: (None, 96, 96, 3)
output: (None, 96, 96, 32)
activation_1: Activation
input: (None, 96, 96, 32)
output: (None, 96, 96, 32)
batch_normalization_1: BatchNormalization
input: (None, 96, 96, 32)
output: (None, 96, 96, 32)

Fig. 3: Keras network architecture for multi-label classification

3.1.2 TRAINING THE NETWORK

CNNs are trained using back propagation. This is a method of dynamic adjustment based on providing feedback to the network, initiated from a difference between the desired output and the current output. The weights of the interconnected neurons are adjusted depending on the degree they contribute to the error and this process is repeated in cycles until the network achieves a desired accuracy of classification. In order to adjust the weights correctly, we need a sufficiently large set of training data. There is no clear formula for how big this data set should be, but one aspect that is important to consider is variance between classes. If the disparity within a class is big, the number of training data objects should be larger. During the training process, for each step, one data object is run through the model and the weights are adjusted. When all training data has passed through the network once, one epoch is completed. The number of steps and epochs are important parts of the training process, as too few or too many can lead to under- or overfitting. If the model is very well adapted to the training data, but does not perform accurately or reliably in the general case, it is said to be overfitted. Underfitting occurs when the chosen model does not fit well with the data used and causes ‘overgeneralization’ by the model.

3.1.3 LOSS FUNCTION

An important part of the design of the network model, is choosing the loss function. The loss function represents the price paid for inaccuracy in predictions made by the neural network. By minimizing the loss function during the training process the error of the network also will be minimized. For classification problems, one of the most popular choices for loss function is the softmax cross entropy functions, defined as:

$$P(y_i|x_i;W)=e_{f_{y_i}}\sum j e_{f_j}$$

where y is the label and p(y) is the probability of the output being correct.

3.2 SOFTWARE

3.2.1 TENSORFLOW

TensorFlow (TF) is an open source API developed by Google mainly for Machine Learning and Deep Learning, but it is also applicable for other numerical computations. Google uses TensorFlow in some of its commercial products, such as their speech recognition API and Gmail, but it is also used for research. The framework can be used both as a backend, with C++, and as a frontend with Python. One of the advantages of using TensorFlow are the many built in functions. We are using the high-level API Estimators, which can be customized if needed. For this project, we have built our own estimator with a custom model.

3.2.2. KERAS

An open-source neural network library written in Python, Keras is designed to enable fast experimentation with deep neural networks. Keras acts as an interface to the low-level operations provided by TensorFlow, and is not designed to act as a standalone machine learning framework. Keras has the advantages of being user-friendly, extensible and modular in nature.

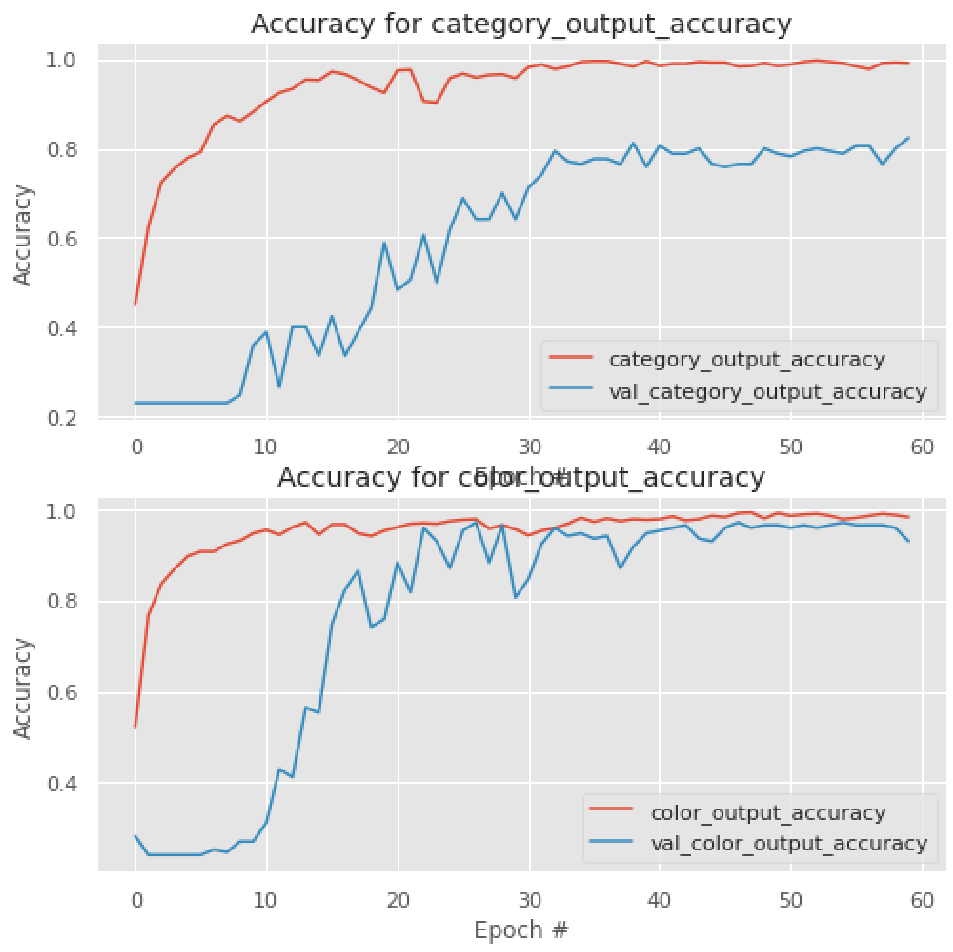
3.2.3 OPENCV

OpenCV is an open-source library of programming functions aimed at the implementation of realtime computer vision. It is well-suited to deep machine learning projects and interfaces reliably with tools like TensorFlow. For this project, the functionality provided by OpenCV is used to process the images that are a part of the data set.

RESULTS AND DISCUSSION

When an image is passed to the model for analysis, the CNN generates a probability score of the labels of the objects in the image. The labels with the highest certainty are superimposed as text on the input image and displayed to the user.

The plot showing epoch and loss/accuracy is presented below:



Based on the dataset used to train and test the data, the model performs reliably with an average accuracy of 98% of color and 80% for category when tested. Further gains can be made with a larger dataset, so that the model keeps evolving and improving. At present, the dataset used is smaller than optimal. With a large enough dataset, more number of classes can be incorporated into the model, so that it can function as a general-purpose object detection/identification software.

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

There exists a wide range of applications for multi-labelled predictions, such as text categorization, semantic image labeling, gene functionality classification etc. and the scope and interest is increasing with modern applications.

2.2 PROPOSED SYSTEM

Objective:

It will classify the image category and color of the image that will feed to the model. that can be used in various application of artificial intelligence.

For example , it can be used in traffic to identify any automobile or in robotics to give the Robot such intelligence to recognise the automobile type and color.

Advantages:

Helpful in artificial intelligence field.

Data mining where images can be categories by its or category.

SYSTEM REQUIREMENT

AND

SPECIFICATIONS

.1 HARDWARE CONFIGURATIONS

The selection of hardware is very important in the existence and proper working of any of the software. When selecting hardware, the size and capacity requirements are also important. This software is able to run with following hardware configuration.

Processor	**: Intel i5 to i7**
Processor Speed	**: 1GHz to 2GHz**
RAM	: 8GB or above
Hard Disk	**: 1GB and above**
Keyboard	**: 108 keys**
Clock speed	: 500 MHz
System bus	: 32 bits

3.2 SOFTWARE CONFIGURATIONS

One of the most difficult tasks is selecting software, once the system requirement is find out then we have to determine whether a particular software package fits for those system requirements. This section summarizes the application requirement.

Operating System	**:** Windows
Language	**:** Python
IDE	**:** kaggle Notebook

3.3 TECHNOLOGY USED

- Python Programming Language
- Dataset for storing trained images
- TensorFlow
- OpenCV
- CNN
- Flask
- HTML

3.4 PLATFORM USED

- Jupyter notebook
- Windows
- VScode editor

SYSTEM DESIGN

4.4 USE CASE DIAGRAM

4.6 ACTIVITY DIAGRAM

4.3 sequence diagram

SYSTEM IMPLEMENTATION,

TESTING

&

MAINTENANCE

5.1 SYSTEM IMPLEMENTATION

We are implementing the system in the Python language. Python is a general-purpose imperative computer language supporting structured programming.

5.2 SYSTEM TESTING

For tesing the accuracy of the model we have made a test dataset of 20% data of our dataset.

We tested the model and it has given 84.32% accuracy on test data.

CONCLUSION:-

The project developed shows how a simple neural network can be trained to identify multiple data points from an input image. This is a stepping stone to full-range computer vision, which can aid artificial intelligence in breaking new grounds. With further development, the convolutional neural network developed can be used to identify dozens of objects, each with multiple attributes that are labelled autonomously by the model.

FUTURE SCOPE:

Features that can be added to the software in the future include hosting it on a web server, so that simultaenous multi-label classification and image analysis can be done remotely on different images uploaded by multiple users.

Code

Uploadpic.py

```
from flask import *

import os

import cv2

from opencv import *

import numpy as np

import matplotlib.pyplot as plt

import time

import sys

import os

import pandas as pd

from collections import Counter

from sklearn.cluster import KMeans

import urllib.request
```

MULTI LABEL CLASSIFICATION

```
from flask import Flask, flash, request, redirect, url_for, render_template

from werkzeug.utils import secure_filename

from modulevechiletetect import myvechiledetection

global result

result=()



app = Flask(__name__,template_folder='templates')

UPLOAD_FOLDER = 'static/uploads/'

app.secret_key = "secret key"

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024

index=["color","color_name","hex","R","G","B"]

csv = pd.read_csv('colors.csv', names=index, header=None)

ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg', 'gif'])
```

```
def allowed_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS



@app.route('/')

def upload_form():

    return render_template('upload.html')



@app.route('/', methods=['POST'])

def upload_image():



    if 'file' not in request.files:

        flash('No file part')

        return redirect(request.url)
```

```
file = request.files['file']

if file.filename == '':

    flash('No image selected for uploading')

    return redirect(request.url)

if file and allowed_file(file.filename):

    filename = secure_filename(file.filename)

    file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

    print('upload_image filename: ' + filename)




    flash('Image successfully uploaded and displayed below')




    return render_template('upload.html', filename=filename)

else:

    flash('Allowed image types are -> png, jpg, jpeg, gif')

    return redirect(request.url)
```

```
print("display")

@app.route('/display/<filename>')

def display_image(filename):



    #print('display_image filename: ' + filename)



    return redirect(url_for('static', filename='uploads/' + filename), code=301)



@app.route('/<filename>')

def result_model(filename):

    print('start')
    path_image=str('static/uploads/' + str(filename))

    result=tuple(myvechiledetection.run(str(path_image)))

    return render_template('result.html',result=result)




if __name__ == '__main__':

    app.run(debug = True)
```

Code

MyVechiledetection.py

```
import cv2

import opencv

import numpy as np

import matplotlib.pyplot as plt

import time

import sys

import os

import pandas as pd

from collections import Counter

from sklearn.cluster import KMeans

global text
```

MULTI LABEL CLASSIFICATION

```
index=["color","color_name","hex","R","G","B"]

csv = pd.read_csv('colors.csv', names=index, header=None)




def getColorName(R, G, B):

    global text

    minimum = 10000

    for i in range(len(csv)):

        d = abs(R - int(csv.loc[i, "R"])) + abs(G - int(csv.loc[i, "G"])) + abs(B - int(csv.loc[i, "B"]))

        if (d <= minimum):

            minimum = d

            cname = csv.loc[i, "color_name"]

    return cname




def run(image_path: str):
    u=0

    str1=[]

    CONFIDENCE = 0.5

    SCORE_THRESHOLD = 0.50

    IOU_THRESHOLD = 0.3



# the neural network configuration

    config_path = "cfg\yolov3.cfg"

# the YOLO net weights file

    weights_path = "cfg\yolov3.weights"

# weights_path = "weights/yolov3-tiny.weights"



# loading all the class labels (objects)

    labels = open("data\coco.names").read().strip().split("\n")
```

```
# generating colors for each object for later plotting

    colors = np.random.randint(0, 255, size=(len(labels), 3), dtype="uint8")




# load the YOLO network

    net = cv2.dnn.readNetFromDarknet(config_path, weights_path)




    image = cv2.imread(image_path)

    print(image_path)

    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    copy_image = image.copy()

    file_name = os.path.basename(image_path)

    filename, ext = file_name.split(".")  # to check later

    h, w = image.shape[:2]

    # create 4D blob
```

```
blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True, crop=False)



# sets the blob as the input of the network

net.setInput(blob)



# get all the layer names

ln = net.getLayerNames()

ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# feed forward (inference) and get the network output

# measure how much it took in seconds

layer_outputs = net.forward(ln)  # PREDICTION HAPPENING IN THIS STEP



font_scale = 1  # FONT PARAMETERS

thickness = 1
```

```
boxes, confidences, class_ids = [], [], []

# loop over each of the layer outputs

# detection = [1:4] : [BoxCenterX, BoxCenterY, Width, Height], [5:] : class confidences

for output in layer_outputs:

    # loop over each of the object detections

    for detection in output:

        # extract the class id (label) and confidence (as a probability) of

        # the current object detection
```

```
scores = detection[5:]

class_id = np.argmax(scores)

confidence = scores[class_id]

# discard out weak predictions by ensuring the detected

# probability is greater than the minimum probability

if confidence > CONFIDENCE:

   # PRESENT IN CONFIGURATIONS

   # scale the bounding box coordinates back relative to the

   # size of the image, keeping in mind that YOLO actually

   # returns the center (x, y)-coordinates of the bounding

   # box followed by the boxes' width and height

   box = detection[:4] * np.array([w, h, w, h])

   (centerX, centerY, width, height) = box.astype("int")

   # use the center (x, y)-coordinates to derive the top and

   # and left corner of the bounding box
```

```
            x = int(centerX - (width / 2))

            y = int(centerY - (height / 2))

            # update our list of bounding box coordinates, confidences,

            # and class IDs

            boxes.append([x, y, int(width), int(height)])

            confidences.append(float(confidence))

            class_ids.append(class_id)



    # FIX INDENTATION OF THIS BLOCK [Till line 77]



    # perform the non maximum suppression given the scores defined before

    idxs = cv2.dnn.NMSBoxes(boxes, confidences, SCORE_THRESHOLD, IOU_THRESHOLD)

    clt = KMeans(n_clusters=4)
```

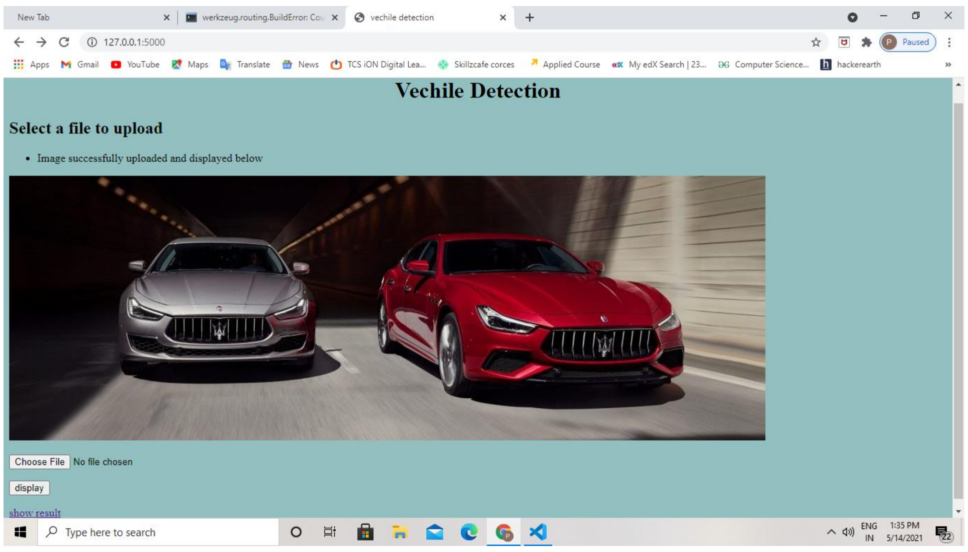
MULTI LABEL CLASSIFICATION

```
vehicles = ['car', 'motorbike', 'bus', 'truck', 'aeroplane']

# plt.imshow(image)

# plt.show()




# ensure at least one detection exists

if len(idxs) > 0:

    # loop over the indexes we are keeping

    for i in idxs.flatten():

        # extract the bounding box coordinates

        x, y = boxes[i][0], boxes[i][1]

        w, h = boxes[i][2], boxes[i][3]

        if i in idxs:

            if labels[class_ids[i]] in vehicles:

                copied_image = image.copy()
                x, y, w, h = boxes[i]

                scaleTop = int(h * 0.30)  # scaling the top with the %

                scaleBottom = int(h * 0.15)  # scaling the bottom with the %

                x1 = x

                y1 = y + scaleTop

                x2 = x + w

                y2 = (y + h) - scaleBottom

                # print("x: {}, y: {}, w: {}, h: {}, scaleTop: {}, scaleBottom: {}".format(x,y, w, h,
scaleTop, scaleBottom))

                # print("scaleTop:y + scaleBottom, x:x + w")

                crop_img = copied_image[y1:y2, x1:x2]

                # print(crop_img.shape)




                crop_img = cv2.cvtColor(crop_img, cv2.COLOR_BGR2RGB)

                crop_img = cv2.fastNlMeansDenoisingColored(crop_img, None, 10, 10, 7, 21)
```

```
            # cv2.imwrite(filename +  str(i) + "_Object." + ext, crop_img)

            pixels = crop_img.reshape((crop_img.shape[0] * crop_img.shape[1], 3))

            labelsinvehicle = clt.fit_predict(pixels)

            label_counts = Counter(labelsinvehicle)

            # subset out most popular centroid

            dominant_color = clt.cluster_centers_[label_counts.most_common(1)[0][0]]

            r, g, b = dominant_color

            color_present = getColorName(b, g, r)

        # draw a bounding box rectangle and label on the image

        color = [int(c) for c in colors[class_ids[i]]]

        cv2.rectangle(image, (x, y), (x + w, y + h), color=color, thickness=thickness)

        if labels[class_ids[i]] in vehicles:

            global text
            text = f"{color_present} {labels[class_ids[i]]}: {confidences[i]:.2f} "

            #str1.append(str("Object - {}: {} - {}".format(i + 1, dominant_color, text)))

            str1.append(text)

          else:

            text = f"{labels[class_ids[i]]}: {confidences[i]:.2f}"

            str1.append(text)

          # calculate text width & height to draw the transparent boxes as background of the
text

          (text_width, text_height) = \

          cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, fontScale=font_scale,
thickness=thickness)[0]

          text_offset_x = x

          text_offset_y = y - 5

          box_coords = ((text_offset_x, text_offset_y), (text_offset_x + text_width + 2,
text_offset_y - text_height))

          overlay = image.copy()
```

```
        # overlay = image = cv2.cvtColor(overlay, cv2.COLOR_BGR2RGB)

        cv2.rectangle(overlay, box_coords[0], box_coords[1], color=color,
thickness=cv2.FILLED)

        # add opacity (transparency to the box)




        image = cv2.addWeighted(overlay, 0.6, image, 0.4, 0)




        # now put the text (label: confidence %)

        cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX,
fontScale=font_scale, color=(0, 0, 0),

            thickness=thickness)

return str1
```

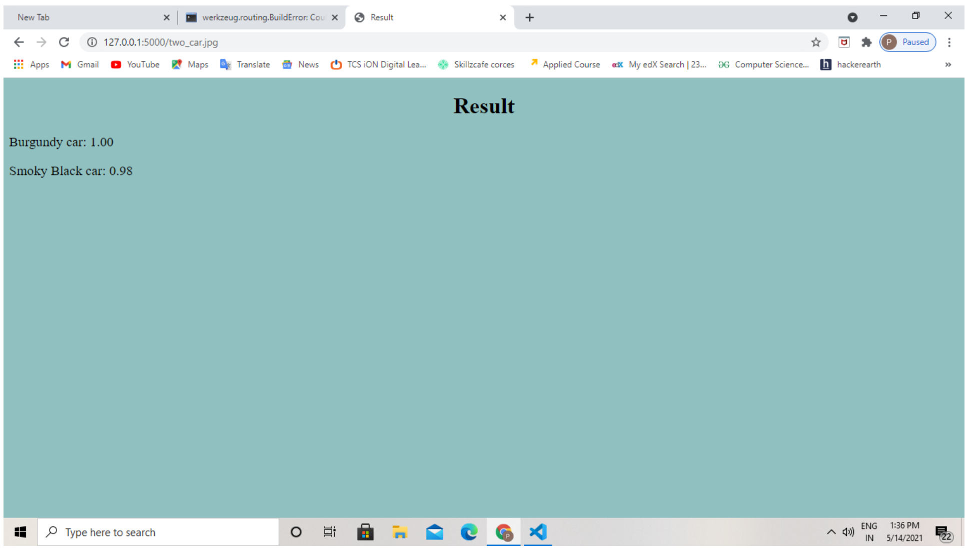
Output



Vechile Detection

Select a file to upload

- Image successfully uploaded and displayed below



Result

Burgundy car: 1.00

Smoky Black car: 0.98