**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

2019-2022

**COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**COCHIN UNIVERSITY COLLEGE OF ENGINEERING**

**KUTTANADU**



**CERTIFICATE**

This is to certify that this project report entitled **“vehicle detection and toll tax calculation ”** 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 **“vehicle detection and toll tax calculation ”** 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 Departmentduring 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 **“vehicle detection and toll tax calculation ”**. 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**

Our project is divided into two major components: vehicle detection and toll tax calculation. the projects is basically a vehicle detection system that takes an image as input and recognise every vehicle of image. after recognising the vehicles it will calculate the toll tax of each vehicle .it will also show the total toll tax amount .

It can be able to detect four types of vehicle such as truck , bus ,motorbike and car.

The image will be recognise by using a machine learning algorithm which is yolov3.

We use HTML ,CSS, JavaScript for the frontend of the projects and in the backend , flask will be used.

**CONTENTS**

**No: Title Page**

1. **INTRODUCTION....................................................................................1**
   1. ABOUT THE PROJECT
   2. OBJECTIVE & SCOPE OF THE PROJECT
   3. DEFINITION OF PROBLEM
2. **SYSTEM ANALYSIS..............................................................................4**
   1. EXISTING SYSTEM
   2. PROPOSED SYSTEM

2.2.1 ADVANTAGES

* 1. FEASIBILITY STUDY

1. **SYSTEM REQUIREMENTS AND SPECIFICATIONS....................9**
   1. HARDWARE CONFIGURATIONS
   2. SOFTWARE CONFIGURATIONS
   3. TECHNOLOGY USED
   4. PLATFORM USED
2. **SYSTEM DESIGN................................................................................12**
   1. USE CASE DIAGRAM
   2. SEQUENCE DIAGRAM
   3. ACTIVITY DIAGRAM
3. **SYSTEM IMPLEMENTATION AND TESTING.............................20**

5.1 SYSTEM IMPLEMENTATION

5.2 SYSTEM TESTING

5.3 SYSTEM MAINTENANCE

1. **CONCLUSION.......................................................................................26**

6.1 CONCLUSION

6.2 FUTURE SCOPE

1. **SAMPLE CODE.....................................................................................28**
2. **SCREENSHOTS....................................................................................33**

**INTRODUCTION**

* 1. **ABOUT THE PROJECT**

LITERATURE REVIEW

METHODOLOGY

3.1 PROCESS FLOW

Yolo ke bare me yhan likh dena

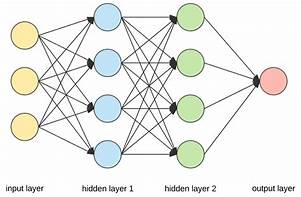


Fig. 1: Simplified representation of a neural network

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.

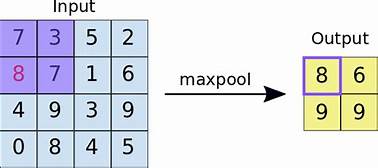


Fig. 2: Max pooling of image data

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.

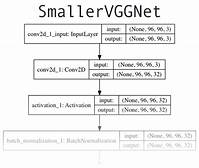


Fig. 3: Keras network architecture for multi-label classification

**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(yi|xi;W)=efyi∑jefj

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.

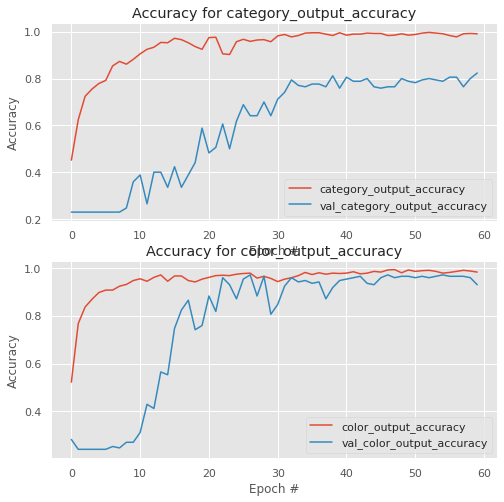
3.2.3 OPENCV

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

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

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)

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

