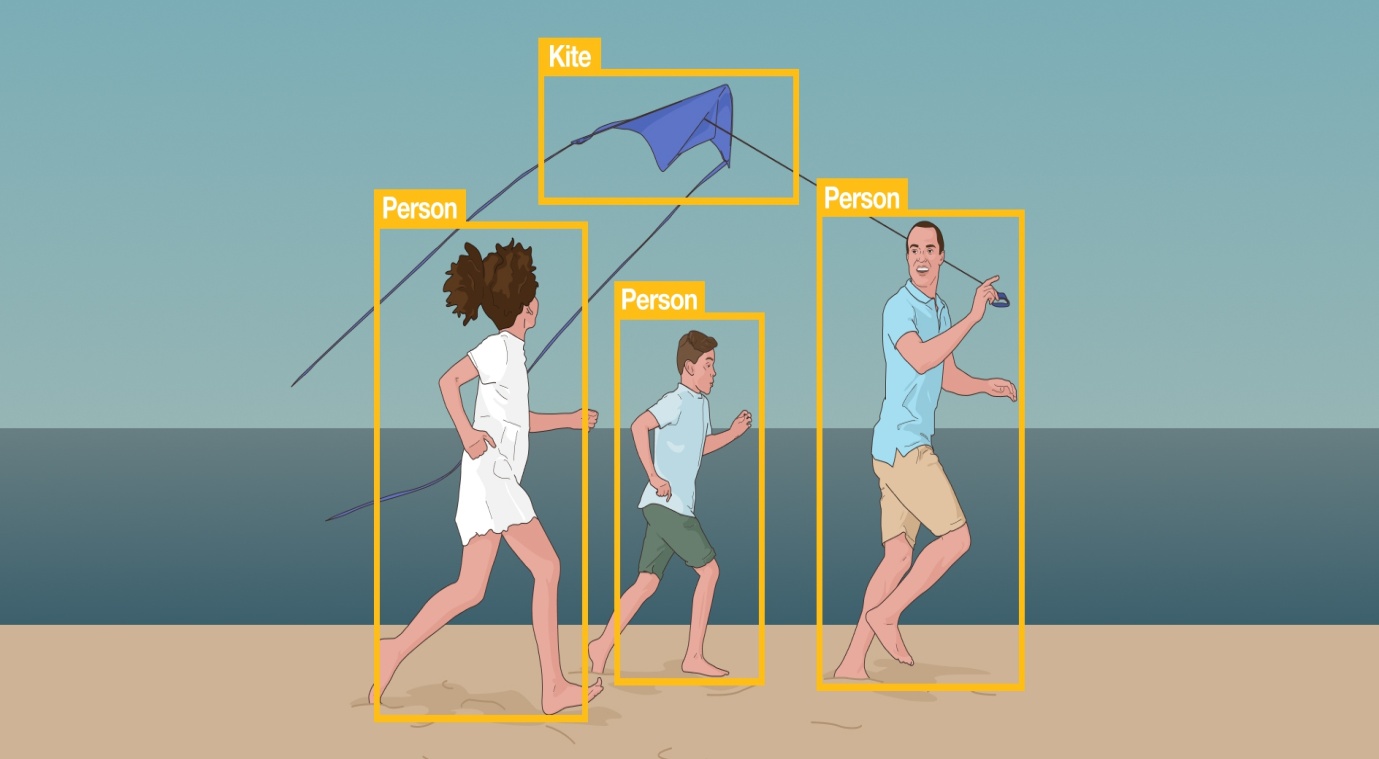
PROJECT REPORT



**IMAGE CLASSIFICATION**

**AND** **OBJECT DETECTION**

SUBMITTED BY-

RIYA GUPTA (A)

***PROBLEM STATEMENT***

* As we are moving forward into modern world, having more precise and detailed object recognition becomes crucial.
* One cares not only about classifying images,but also about precisely estimating the class and location of the objects contained within the images, a problem known as Object detection.
* The more complicated problem (this project), of object detection involves both classification and localization.
* Main Objectives of this project are-

1.Image Classification.

2.Dataset Preprocessing.

3.Object Detection, Segmentation,

Recognization.

4.Object Based image retrieval.

**CODE SNIPPETS AND EXPLANATION**

First, let’s import the required packages as follows:

from tensorflow import keras

import matplotlib.pyplot as plt

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

import numpy as np

import cv2

The dataset can be loaded using the code below:

(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.cifar10.load\_data()

The dataset used in this blog is the CIFAR-10 dataset.

Further, we can obtain the size of the train and test datasets as shown below:

x\_train.shape,x\_test.shape

# Normalization

x\_train = x\_train/255.0

x\_test = x\_test/255.0

The required code for the train set is as follows:

#sklearn expects i/p to be 2d array-model.fit(x\_train,y\_train)=>reshape to 2d array

nsamples, nx, ny, nrgb = x\_train.shape

x\_train2 = x\_train.reshape((nsamples,nx\*ny\*nrgb))

The required code for the test set is given below:

#so,eventually,model.predict() should also be a 2d input

nsamples, nx, ny, nrgb = x\_test.shape

x\_test2 = x\_test.reshape((nsamples,nx\*ny\*nrgb))

from sklearn.ensemble import RandomForestClassifier

Then I Create an instance of the RandomForestClassifier class:

knn=KNeighborsClassifier(n\_neighbors=7)

knn.fit(x\_train2,y\_train)

y\_pred\_knn=knn.predict(x\_test2)

y\_pred\_knn

Now evaluate the model with the test images by obtaining its classification report, confusion matrix, and accuracy score.

accuracy\_score(y\_pred\_knn,y\_test)

print(classification\_report(y\_pred\_knn,y\_test))

confusion\_matrix(y\_pred\_knn,y\_test)

from sklearn.tree import DecisionTreeClassifier

and then instantiating it to create a KNN model:

dtc=DecisionTreeClassifier()

dtc.fit(x\_train2,y\_train)

y\_pred\_dtc=dtc.predict(x\_test2)

y\_pred\_dtc

Now, we proceed to evaluate the KNN model just the way we evaluated our previous model.

accuracy\_score(y\_pred\_dtc,y\_test)

print(classification\_report(y\_pred\_dtc,y\_test))

confusion\_matrix(y\_pred\_dtc,y\_test)

img\_path='/content/bird.jpg'

img\_arr=cv2.imread(img\_path)

img\_arr=cv2.resize(img\_arr,(32,32))

#so,eventually,model.predict() should also be a 2d input

nx, ny, nrgb = img\_arr.shape

img\_arr2 = img\_arr.reshape(1,(nx\*ny\*nrgb))

classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]

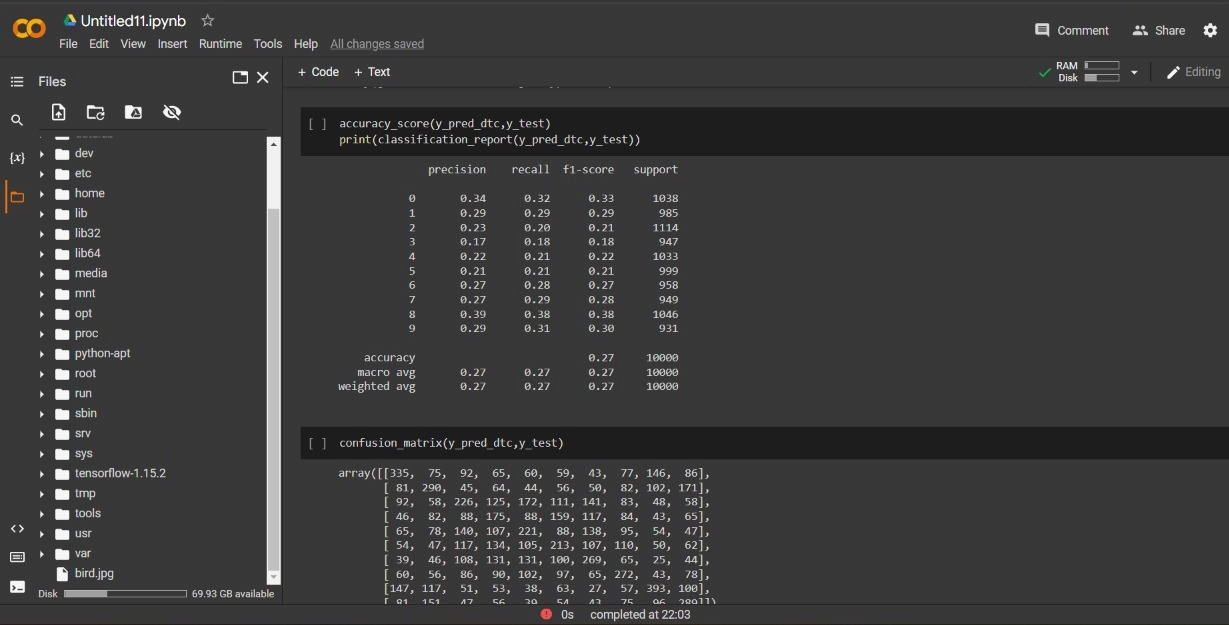
For Final output-

ans=model.predict(img\_arr2)

print(classes[ans[0]])

#RandomForestClassifier

**OUTPUT SNAPSHOTS**

****

