**MID SEMESTER REPORT**

**On**

**Object Detection & Image Recognition using Python**

**Submitted by**

Paras Sharma (Enrol. No. R114216017)

**Under the guidance of**

Ms. Aradhana Kumari Singh

Assistant Professor, Department of Informatics

School Of Computer Science



**SCHOOL OF COMPUTER SCIENCE**

UNIVERSITY OF PETROLEUM & ENERGY STUDIES

Bidholi Campus, Energy Acres, Dehradun – 248007.

**Year – 2019**

 **School of Computer Science**

**University of Petroleum & Energy Studies, Dehradun**

**Project Proposal Approval Form (2019)**

**Major I**

**PROJECT TITLE:** Object Detection & Image Recognition using Python

**ABSTRACT**

With the tremendous increase in the amount of images that are being produced daily, there was a need for the development of a robust and efficient object detection system. Object recognition is the process by which objects are detected within images and videos.

Object detection can be used for various purposes including retrieval and surveillance. In this study, various basic concepts used in object detection are described while making use of OpenCV library of python 2.7, improving the efficiency and accuracy of object detection are presented.

**Keywords:​** *Object Detection, Python, OpenCV, NumPy, Haar Cascade*

**INTRODUCTION**

The modern world is encircled with heavy masses of digital visual information, may it be images, videos and so on. It is a need of hour to analyse and organise these plundering ocean of visual information and for these techniques are required. Particularly, it will be more useful to analyse semantic information of the images or videos. One imperative part of image content is the objects in the image.

So there is a need for object detection and image recognition techniques.

Object detection is a vital, yet difficult vision task. It is a basic part in numerous applications, for example, image search, image auto-annotation and scene understanding; be that as it may it is as yet an open issue because of the intricacy of object classes and images. It is being widely used in industries to ease user, save time and to achieve parallelism. Object detection is a part of computer vision which aims at having a human like vision, which can locate and differentiate various objects such as, numbers, location, size, position etc. The common object detection method is the color-based approach, detecting objects based on their color values.

Object detection is one of the most challenging applications of the image processing. It is a branch of computer vision and artificial intelligence. The aim of this project is to identify and locate the objects in the images or videos and also naming the specific objects detected.

OpenCv (Python) is used for the implementation of the project. NumPy library is also required for the same. Also, an OpenCv algorithm i.e. Haar Cascade is used and Haar like features are emphasized.

**PROBLEM STATEMENT**

It is not possible for machine to identify an object and classify them according to different classes.

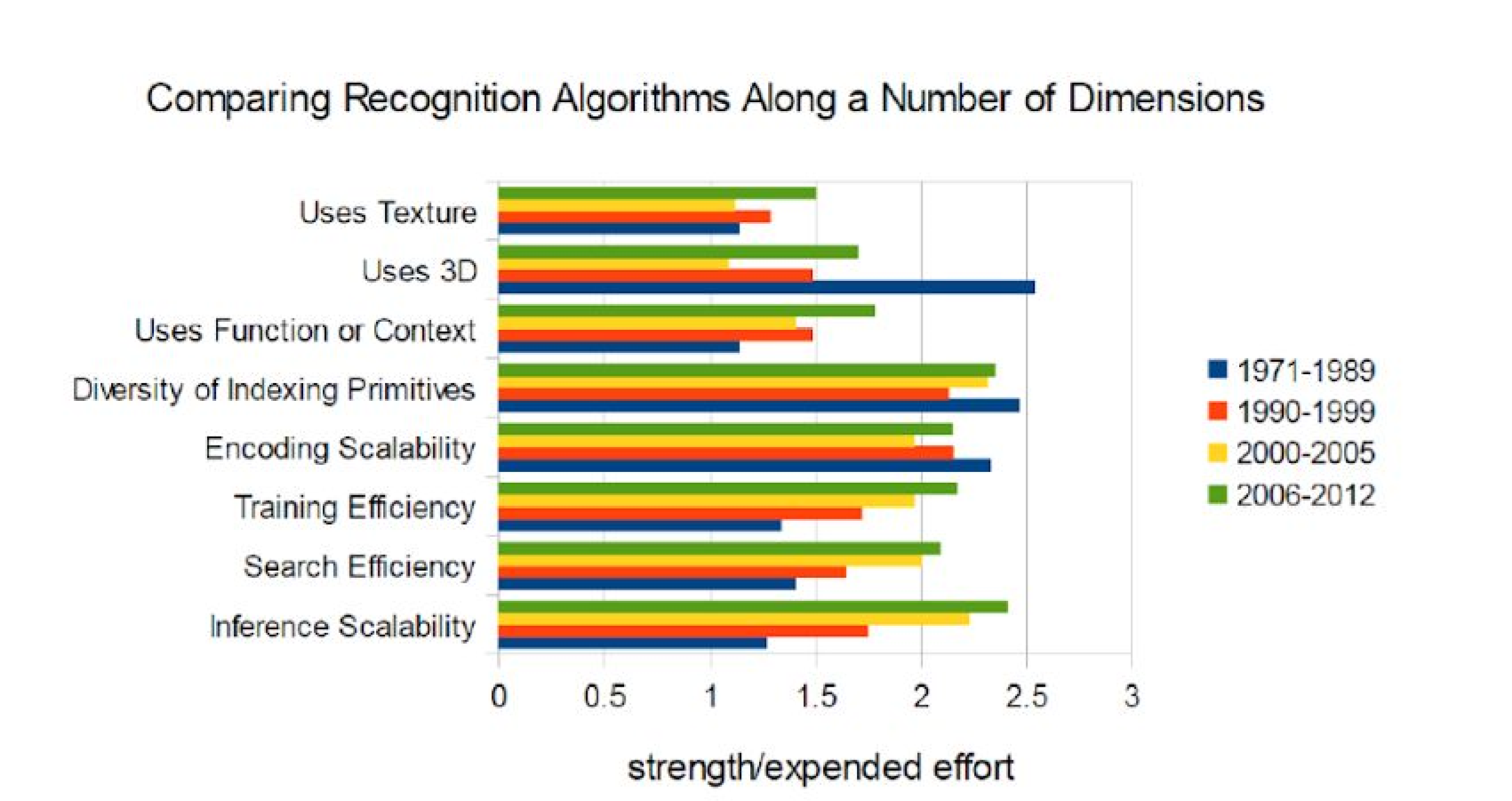
Hence, the aim of this project is to identify and locate the objects in the images or videos and also naming the specific objects detected.

**LITERATURE REVIEW**

Modern computer vision approach has its starting point in early 1960s. First and foremost application was pattern recognition systems used in offices. While performing a task, some practical difficulties such as scene complexities increased as some illumination variability increased. Also time, cost and sensor noise constraints became for common.

1964 involved the automation of the wire-bonding process of transistors, with the ultimate goal of replacing human workers. However, it attained 95% accuracy in test labs but regarded too low to replace human workers. By 1973, fully automated assembly machines had been made, resulting in the world’s 1st image-based machine for the automatic assembly of semiconductor devices. It is much more evolved in these 50 years and achieved almost human accuracy.

As the experiments went on and on making the systems more efficient and reducing the need of human workers, computer vision has now become an effective and efficient technology. Working very much like human vision. It is now being used in various fields like face recognition, autonomous cars, and robots and of course object detection and recognition.



# **Computer Vision**

Humans use their eyes and brain to see and spot the objects around. Computer vision is the science giving similar functionality and capability to a machine or a computer. It aims at enabling computers to see, identify and process the images in the same way as human vision does, and then provide appropriate output. It resembles giving human intelligence and instincts to a computer.

**OBJECT DETECTION**

It is the technology in the field of computer vision for finding and identifying objects in an image or video sequence. Humans recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may vary somewhat in different viewpoints, in many different sizes and scales or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for computer vision systems. Many approaches to the task have been implemented over multiple decades.

**APPLICATIONS**

Object recognition technology has matured to a point at which exciting applications are becoming possible. Indeed, industry has created a variety of computer vision products and services from the traditional area of machine inspection to more recent applications such as video surveillance, or face recognition.

**Face detection**

Popular applications include face detection and people counting. Have you ever noticed how Facebook detects your face when you upload a photo? This is a simple application of object detection that we see in our daily life.

**People Counting**

Object detection can be also used for people counting, it is used for analysing store performance or crowd statistics during festivals. These tend to be more difficult as people move out of the frame quickly (also because people are non-rigid objects).

**Vehicle detection**

Similarly when the object is a vehicle such as a bicycle or car, object detection with tracking can prove effective in estimating the speed of the object. The type of ship entering a port can be determined by object detection (depending on shape, size etc.). This system for detecting ships are currently in development in some European countries.

**Manufacturing Industry**

Object detection is also used in industrial processes to identify products. Say you want your machine to only detect circular objects. Hough circle detection transform can be used for detection.

**Online images**

Apart from these object detection can be used for classifying images found online. Obscene images are usually filtered out using object detection.

**Security**

In the future we might be able to use object detection to identify anomalies in a scene such as bombs or explosives (by making use of a quadcopter).

**CONTENT BASED IMAGE RETRIEVAL (CBIR)**

Many retrieval methods that accept query images as input from the user represent images as vectors in the feature space and search for images based on their features and feature representations. When the user presents a sample query image, region of interest (ROI), or pattern to the system, it performs various visual query mechanisms, such as the query-by example (QBE) paradigm, and finally outputs the relevant images. In CBIR, image content is frequently represented using image features. CBIR finds applications in internet, advertising, medicine, crime detection, entertainment, and digital libraries. High retrieval efficiency and less computational complexity are the desired characteristics of CBIR system and they are the key objectives in the design of a CBIR system.

**Convolutional Neural Network (CNN)**

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. **Object Detection comes under Convolutional Neural Network (CNN).**

**OBJECTIVES**

* The aim of this project is to identify and locate the objects in the images or videos and also naming the specific objects detected.

**METHODOLOGY**

**IMPLEMENTATION OF OBJECT DETECTION**

In Object Detection, a particular object will be detected from whole image. For example, from an image, consisting of several animals, we want to find particularly image of a cat, then a set of images of cat will be stored and image detector will be trained. It is a part of computer vision.

**Selecting a Model**

The default model in the notebook is the simplest (and fastest) pre-trained model offered by Tensor Flow. Looking at the table below, you can see there are many other models available. mAP stands for mean average precision, which indicates how well the model performed on the COCO dataset. Generally models that take longer to compute perform better.

To get a rough approximation for performance just try each model out on a few sample images. If the item you are trying to detect is not one of the 90 COCO classes, find a similar item (if you are trying to classify a squirrel, use images of small cats) and test each model’s performance on that.

**COCO DATASET**

COCO is a large image dataset designed for object detection, segmentation, person key points detection, stuff segmentation, and caption generation. COCO dataset contains photos of 91 objects types that would be easily recognizable by a 4 year old. With a total of 2.5 million labelled instances in 328k images, the creation of 25 our dataset drew upon extensive crowd worker involvement via novel user interfaces for category detection, instance spotting and instance segmentation

**IMPLEMENTATION OF CBIR USING PYTHON**

In CBIR, we had created a retrieval system that used the content of image. The user had to provide a query image, color histogram of the query image would be calculated and the closest images from the database would be retrieved.

**The Four Steps of Any CBIR System**

Any Content Based Image Retrieval System has four basic steps –

**1. Defining the Image Descriptor**: We need to decide what aspect of the image we want to describe namely - the colour of the image, the shape or the texture of the image.

**2. Indexing the Dataset :** Now that we have the image descriptor defined, we need to apply this image descriptor to each image in the dataset, extract 22 features from these images, and write the features to storage (ex. CSV file, RDBMS, Redis, etc.) so that they can be later compared for similarity.

**3. Defining the similarity metric**: After indexing the dataset, we now have a bunch of feature vectors. We need to compare these feature vectors. Popular choices include the Euclidean distance, Cosine distance, and chi-squared distance, but the actual choice is highly dependent on (1) the dataset and (2) the types of features that we have extracted.

**4. Searching:** The final step is to perform an actual search. A user will submit a query image and we have to (1) extract features from this query image and then (2) apply the similarity function to compare the query features to the features already indexed. Then, the most relevant results are returned according to the similarity function.

**COLLECTING DATASET FOR OBJECT DETECTION**

* First of all, we have downloaded the example dataset, which includes a zip folder of 2,000 pictures of cats and dogs and we extracted it locally in /tmp. The dataset which we are using is being taken from Kaggle, which contains 25,000 images.



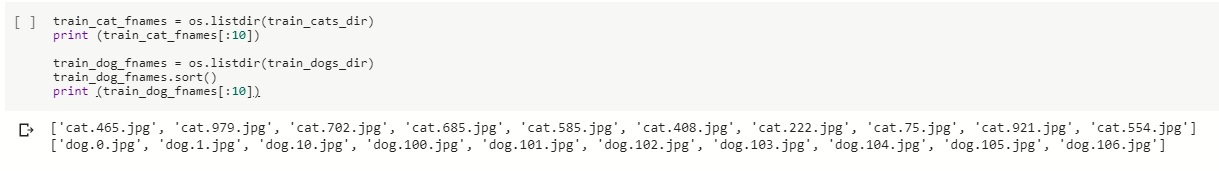
* Then, we have extracted the contents of zip file to the base directory /tmp/cats\_and\_dogs\_filtered, which contains train and validation subdirectories for the training and validation datasets which in turn each contains cats and dogs subdirectories.



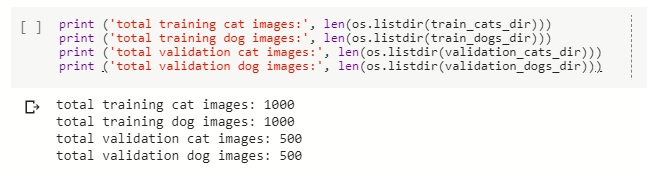
* The contents of the .zip are extracted to the base directory /tmp/cats\_and\_dogs\_filtered, which contains train and validation subdirectories for the training and validation datasets which in turn each contain cats and dogs subdirectories. Let's define each of these directories:



* Now, let's see what the filenames look like in the cats and dogs train directories (file naming conventions are the same in the validation directory):

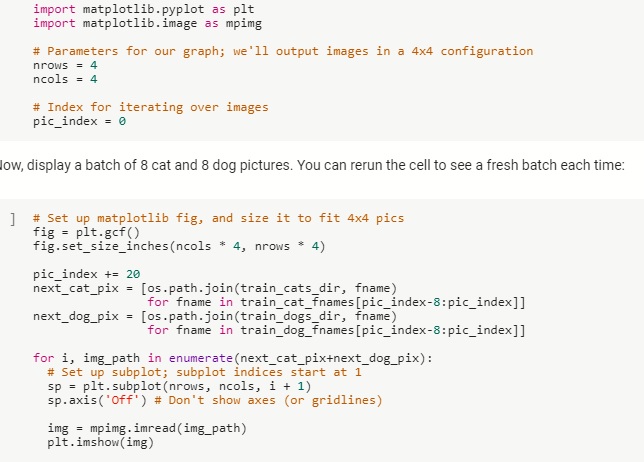


* Let's find out the total number of cat and dog images in the `train` and `validation` directories:



* For both cats and dogs, we have 1,000 training images and 500 test images.

Now let's take a look at a few pictures to get a better sense of what the cat and dog datasets look like. First, configure the matplot parameters:



**SYSTEM REQUIREMENTS**

* **Operating System:** Windows 10
* **Minimum Hardware:** 2 GB RAM, 1GB Free Hard Disk Storage, 32/64 –Bit Processor (Intel i5)
* **Environment :** Google Colaboratory (For Python)
* **Programming Language to be used:** Python 3.6
* **Libraries:** CV2, NumPy
* **Dataset to be used:** COCO Dataset , Images of Cats & Dogs used from Kaggle

**CODE & DEMO (COLLECTING DATASET)**

import os

import zipfile

local\_zip = '/tmp/cats\_and\_dogs\_filtered.zip'

zip\_ref = zipfile.ZipFile(local\_zip, 'r')

zip\_ref.extractall('/tmp')

zip\_ref.close()

The above code is used to extract the contents of zip file to the base directory, which contains train and validation subdirectories for the training and validation datasets which in turn each contain cats and dogs subdirectories. The code given below will help us to define each of these directories:

base\_dir = '/tmp/cats\_and\_dogs\_filtered'

train\_dir = os.path.join(base\_dir, 'train')

validation\_dir = os.path.join(base\_dir, 'validation')

# Directory with our training cat pictures

train\_cats\_dir = os.path.join(train\_dir, 'cats')

# Directory with our training dog pictures

train\_dogs\_dir = os.path.join(train\_dir, 'dogs')

# Directory with our validation cat pictures

validation\_cats\_dir = os.path.join(validation\_dir, 'cats')

# Directory with our validation dog pictures

validation\_dogs\_dir = os.path.join(validation\_dir, 'dogs')

Now the given code below will help us to find what the filenames look like in the cats and dogs train directories:

train\_cat\_fnames = os.listdir(train\_cats\_dir)

print (train\_cat\_fnames[:10])

train\_dog\_fnames = os.listdir(train\_dogs\_dir)

train\_dog\_fnames.sort()

print (train\_dog\_fnames[:10])

Now we will find out the number of cat and dog images in the train and validation directories:

print ('total training cat images:’ len(os.listdir(train\_cats\_dir)))

print ('total training dog images:’ len(os.listdir(train\_dogs\_dir)))

print ('total validation cat images:’ len(os.listdir(validation\_cats\_dir)))

print ('total validation dog images:’ len(os.listdir(validation\_dogs\_dir)))

For both cats and dogs, we have 1,000 training images and 500 test images.

Now let's take a look at a few pictures to get a better sense of what the cat and dog datasets look like. First, configure the matplot parameters:

%matplotlib inline

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

# Parameters for our graph; we'll output images in a 4x4 configuration

nrows = 4

ncols = 4

# Index for iterating over images

pic\_index = 0

Now, we will display a batch of 8 cat and 8 dog pictures which is given in the below code:

# Set up matplotlib fig, and size it to fit 4x4 pics

fig = plt.gcf()

fig.set\_size\_inches(ncols \* 4, nrows \* 4)

pic\_index += 20

next\_cat\_pix = [os.path.join(train\_cats\_dir, fname)

for fname in train\_cat\_fnames[pic\_index-8:pic\_index]]

next\_dog\_pix = [os.path.join(train\_dogs\_dir, fname)

for fname in train\_dog\_fnames[pic\_index-8:pic\_index]]

for i, img\_path in enumerate(next\_cat\_pix+next\_dog\_pix):

# Set up subplot; subplot indices start at 1

sp = plt.subplot(nrows, ncols, i + 1)

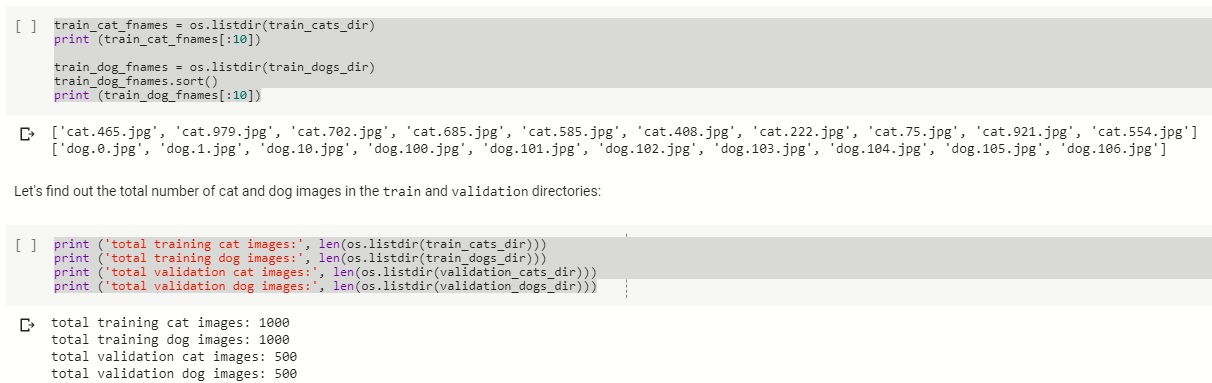
sp.axis('Off') # Don't show axes (or gridlines)

img = mpimg.imread(img\_path)

plt.imshow(img)

plt.show()

**OUTPUT**



The snippet given above is the no of cat and dog images in jpg format and given below it is the no of training images and validation images of cats and dogs which are to be trained and validated further by CNN which we would be going to work upon.



These are 8 cat images which we have extracted from the zip folder and we have actually obtained these images with the help of matplotlib. Given below are the images of dogs which we have extracted along with these images as well.



**SCHEDULE**

**Study Period**

**Duration 1 weeks**

**Requirement gathering**

**Duration 1 week**

**Pseudo code**

**Duration 1 week**

**Design**

**Duration 2 week**

**Coding and implementation**

**Duration 4 weeks**

**Debugging**

**Duration 1 week**

**Testing**

**Duration 2 week**

**Publish Report**

**Duration 1 week**

**REFERENCES**

*[1]R. Priyatharshini and S. Chitrakala, "Association based image retrieval: A survey, Springer-Verlag Berlin Heidelberge, pp. 17-26, 2013.*

*[2]Y. Rui and T. S. Huang, “Image retrieval: Current techniques, promising directions and open issues," Journal of Visual Communication and Image Representation, vol. 10, pp. 39{62, 1999.*

*[3] A. Ponomarev et al., "Content-Based Image Retrieval Using Color, Texture and Shape Features", Key Engineering Materials, Vol. 685, pp. 872-876, 2016*

*[4, 1]Yushi Jing, David Liu , Dmitry Kislyuk , Andrew Zhai , Jiajing Xu , Jeff Donahue, Sarah Tavel “Visual Search at Pinterest “ 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015*

*[5] U.S.N. Raju, Shibin George, V. Sairam Praneeth, Ranjeet Deo,Priyanka Jain Department of Computer Science and Engineering, National Institute of Technology, Warangal, India -“Content Based Image Retrieval on Hadoop Framework”*

*[6] Nima Razavi, Juergen Gall and Luc Van Gool, “Scalable Multi-class Object Detection”, IEEE Conference on Computer Vision and Pattern Recognition, pp. 1505- 1512, 2011.*

*[7]Swati V. Sakhare & Vrushali G. Nasre,” Design of Feature Extraction in Content Based Image Retrieval (CBIR) using Color and Texture.*

**Synopsis Draft verified by**

**Project Guide HOD**

**(Name & Sign) (Dept. of Informatics)**