ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

Deep Learning Approaches in Computer Vision

Ravi Kumar Vadana
Department of Computer Science & Engineering, Sarada Institute of Science, Technology and Management, Srikakulam, Andhra Pradesh 532404, India.

Mr.K.Srinivasarao
Department of Computer Science & Engineering, Sarada Institute of Science, Technology and Management, Srikakulam, Andhra Pradesh 532404, India.

ABSTRACT

Efficient and correct object detection has been an important topic in the advancement of computer vision systems. With the arrival of deep learning techniques, the accuracy for object detection has increased drastically. The project aims to include progressive technique for object detection with the goal of achieving high accuracy with real time performance.

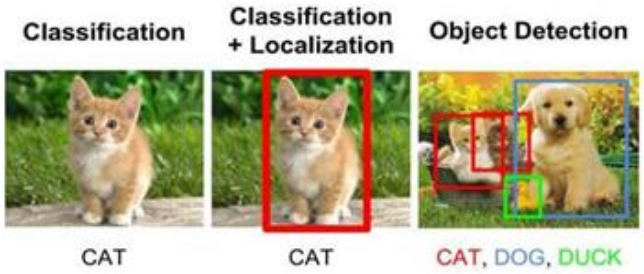
A significant challenge in many of the object detection systems is the dependency on other computer vision techniques for helping the deep learning based approach, which leads to slow and non-optimal performance.

In this project, we used a completely deep learning based approach to solve the problem of object detection in an end-to-end fashion. The network is trained on the most challenging publicly available dataset (PASCAL VOC), on which an object detection challenge is conducted annually. The resulting system is fast and accurate, thus aiding those applications which require object detection.

INTRODUCTION

Problem Statement

Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problems was that of image classification, which is defined as predicting the class of the image. A slightly complicated problem is that of image localization [1-5], where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem (this project), of object detection involves both classification and localization.



Computer Vision Task

CHALLENGES

The major challenge in this problem is that of the variable dimension of the output which is caused due to the variable number of objects that can be present in any given input image. Any general machine learning task requires a fixed dimension of input and output for the model to be trained. Another important obstacle for widespread adoption of object detection systems is the requirement of real-time while being accurate in detection. The more complex the model is, the more time it requires for inference; and the less complex the model is, the less is the accuracy. This trade-off between accuracy and performance needs to be chosen as per the application. The problem involves classification as well as regression, leading the model

Cite this article as: Ravi Kumar Vadana & Mr.K.Srinivasarao, "Deep Learning Approaches in Computer Vision", International Journal of Research in Advanced Computer Science Engineering, Volume 5 Issue 4, 2019, Page 18-28.

ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

to be learnt simultaneously. This adds to the complexity of the problem [2], [11-19].

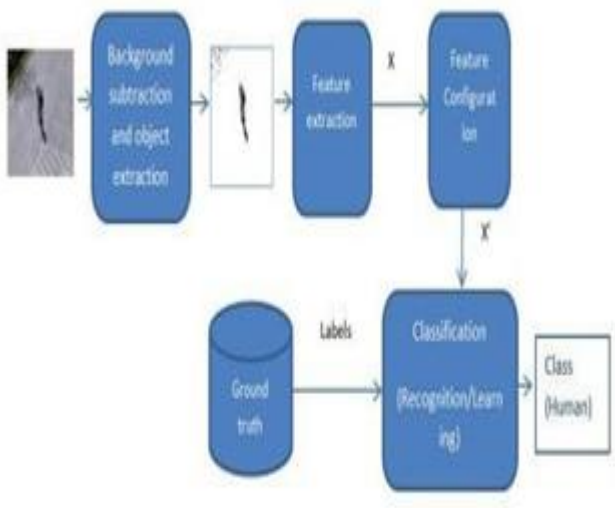
REQUIREMENT ANALYSIS

Software Requirements Specification (SRS) is a complete description of the behavior of the system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software.

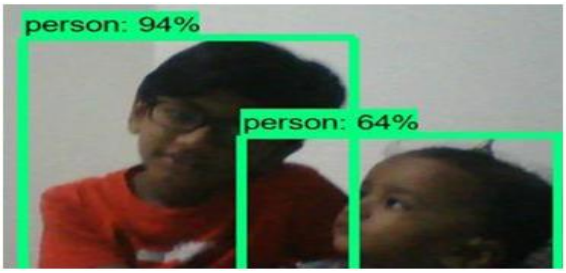
FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability) [6].

Detect object from image and labeling



Detect Accuracy of the face object



Detect Face emotions like Sad, Angry, Happy etc and labeling

NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals" and "quality of service requirements, ―and "non-behavioral requirements."[1] Qualities, that is, non- functional requirements, can be divided into two main categories:

1. Execution qualities, such as security and usability, which are observable at run time.
2. Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL) [8], especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

1. Central Processing Unit (CPU) — Intel Core i5 6th Generation processor or higher. An AMD equivalent processor will also be optimal.
2. RAM — 8 GB minimum, 16 GB or higher is recommended.
3. Graphics Processing Unit (GPU) — NVIDIA GeForce GTX 960 or higher. AMD GPUs are not able to perform deep learning regardless. For more information on NVIDIA GPUs for deep learning please visit https://developer.nvidia.com/cuda- gpus.
4. Operating System — Ubuntu or Microsoft Windows 10. I recommend updating Windows 10 to the latest version before proceeding forward.
5. Keyboard: Standard Keyboard
6. Mouse: Standard Mouse

Please note that based on the dataset size, we need higher configuration systems like GPUs (Graphical processing Units) instead of CPUs.

SOFTWARE REQUIREMENTS

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

1. **Operating System:** Any Operating System Windows, Mac or Linux (Preferably 64 Bit Computer)
2. **Platform:** Anakonda3
3. **Language:** Python (Preferably Version >=3.0)
4. **Python Modules:** TensorFlow,ObjectCV, NLTK (Natural Language Tool Kit) Keras, TFLearn
5. **Integrated Development Environment:** Any IDE (Integrated Development Environment) Spyder or Jupiter Notebook or VS Code or Sublim Text or Atom or PyCharm

SYSTEM DESIGN

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.

Methods for computer system design [citation needed]. The UML has become the standard language used in Object-oriented analysis and design [citation needed]. It is widely used for modeling software

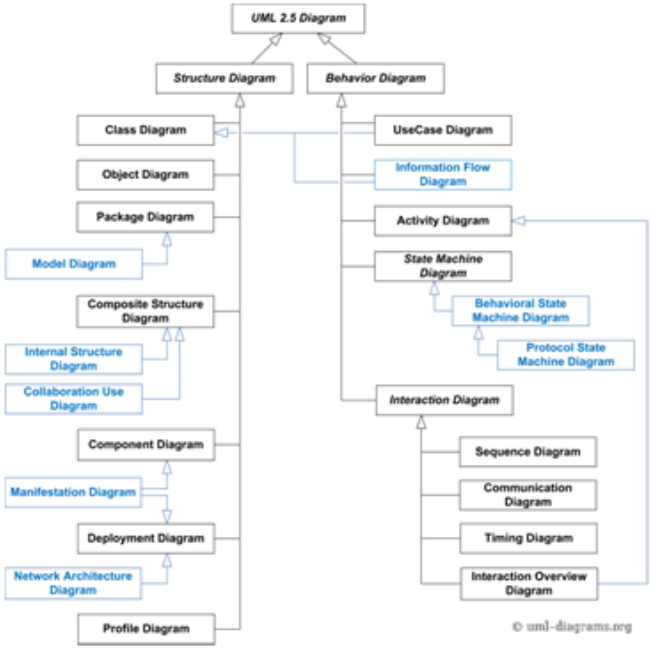
Systems and is increasingly used for high designing non-software systems and organizations.

Unified Modeling Language (UML) :

The Unified Modeling Language (UML) [10] is a general-purpose, developmental, modeling language in the field of that is intended to provide a standard way to visualize the design of a system.

The Unified Modeling Language (UML) offers a way to visualize a system's architectural blueprints in a diagram (see image), including elements such as:

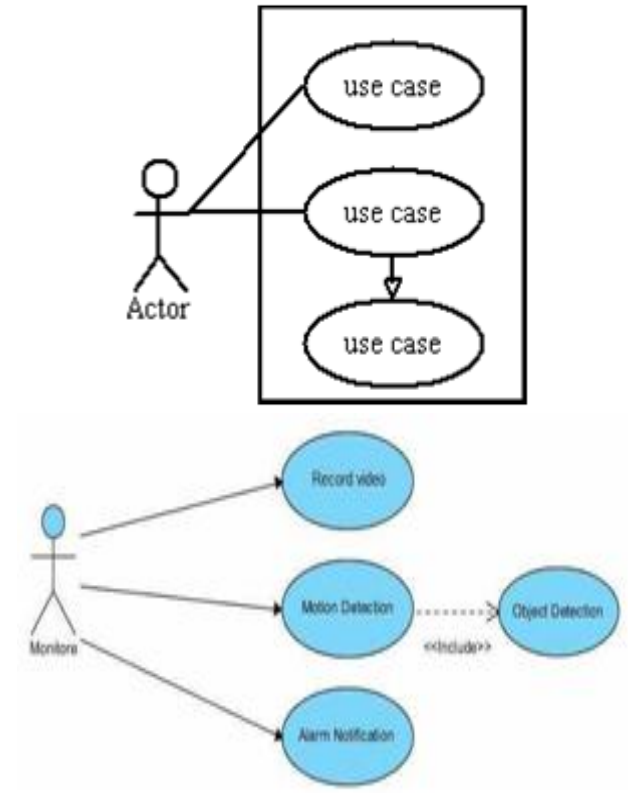
- Any activity
- Individual component of the system
- And how they can interact with the other components
- How the system will run
- How entities interact with others (components and interfaces)
- External user interface



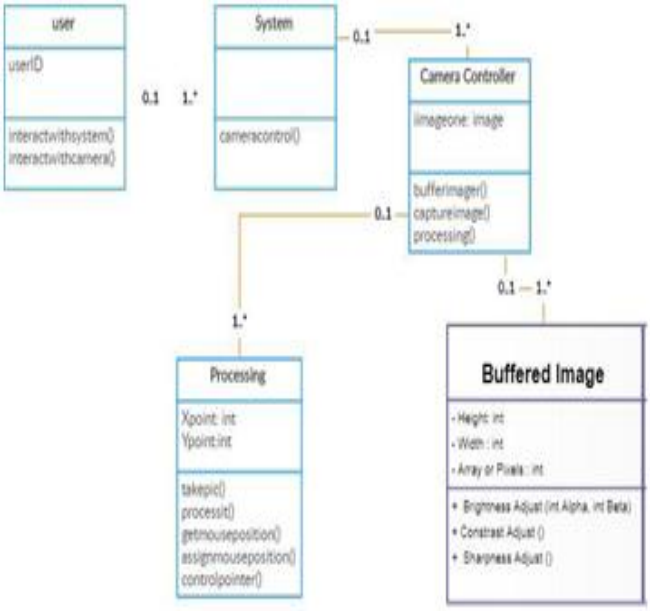
ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

Classification of UML Diagrams

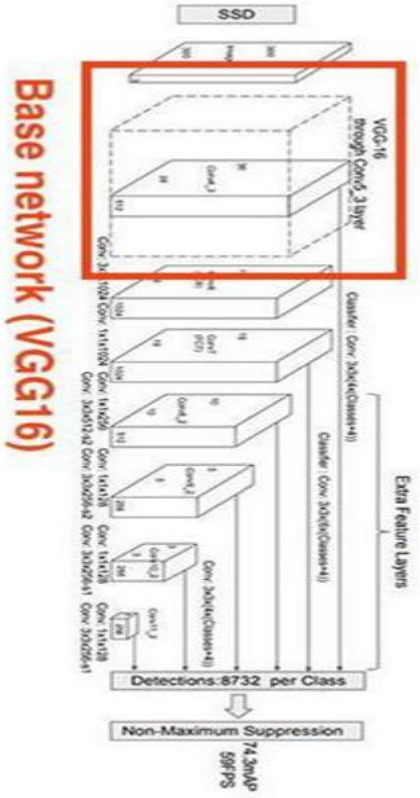
Use case Diagram



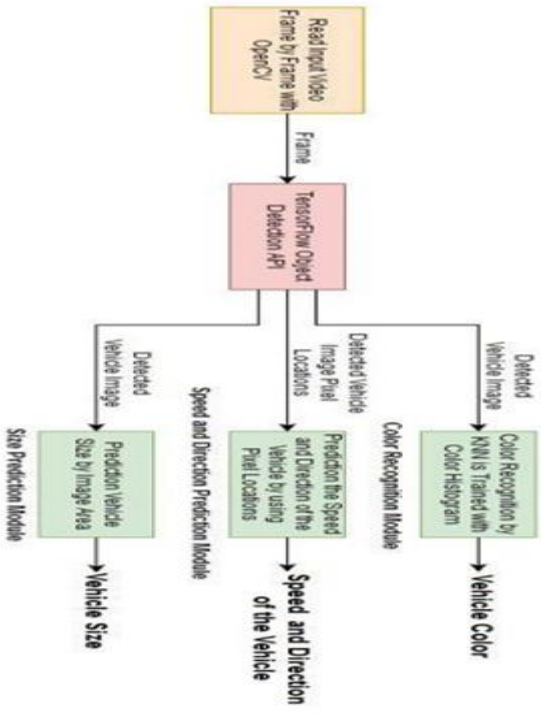
Use Case Diagram of Object detection and notification on Object Behavior



Class Diagram of Web Camera Capturing System

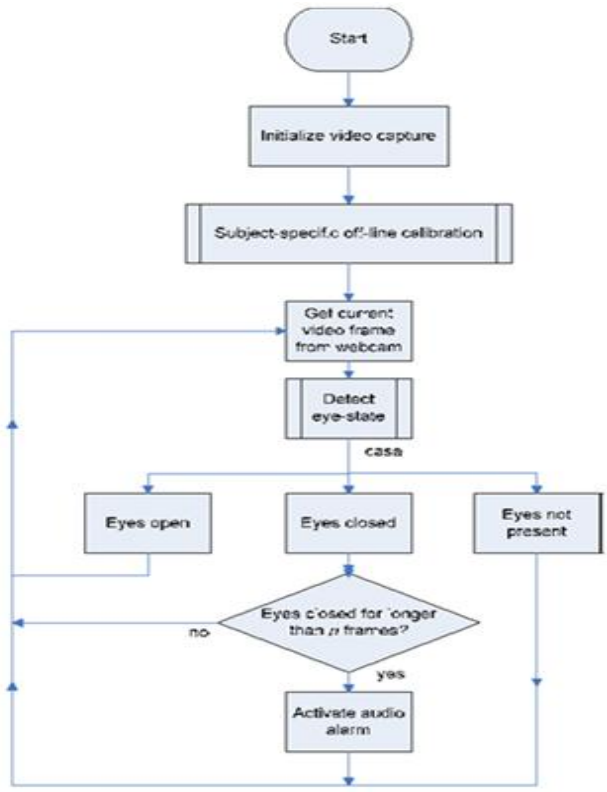


Activity Diagram of Object Detection

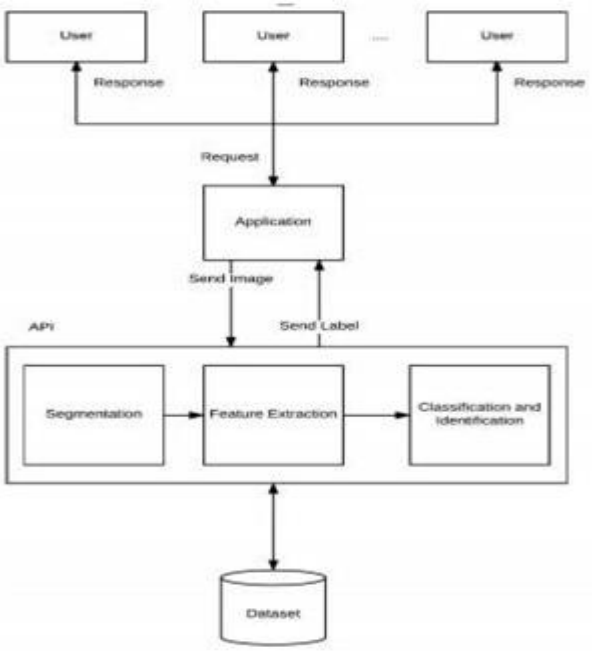


ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

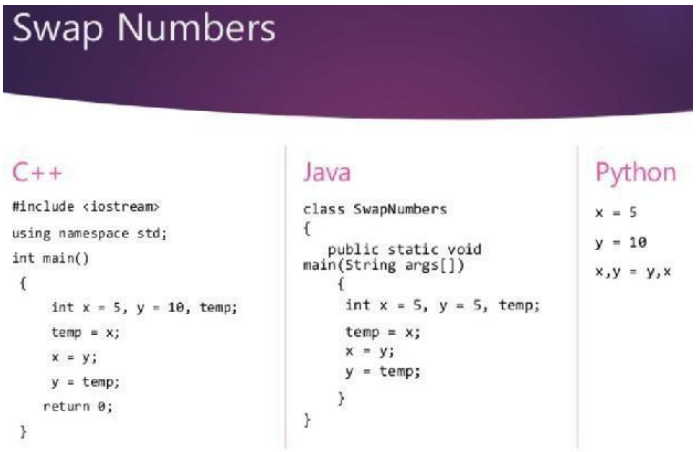
Building Blocks Diagram



State Chart Diagram of Object Eye



System Architecture



Python Language Comparison with C++ and Java

FEATURES OF PYTHON:

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial SQL and No SQL databases.

ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

 GUI Programming − Python supports GUI applications that can be created and
ported to many system calls, libraries and windows systems, such as Windows
MFC, Macintosh, and the X Window system of Unix.
 Scalable − Python provides a better structure and support for large programs than
shell scripting.
Apart from the above-mentioned features, Python has a big list of good features,
few are listed below –
1) It supports functional and structured programming methods as well as OOP.
2) It can be used as a scripting language or can be compiled to byte-code for
building large applications.
3) It provides very high-level dynamic data types and supports dynamic type
checking.
4) It supports automatic garbage collection.
5) It can be easily integrated with C, C++, COM, ActiveX, CORBA, Java and
.NET.

SYSTEM IMPLEMENTATION

Sample Code

Object_image.py:

```
# coding: utf-8
# # Object Detection Demo
# Welcome to the object detection inference walkthrough! This notebook will
walk you step by step through the process of using a pre-trained model to detect
objects in an image. Make sure to follow the [installation
instructions](https://github.com/tensorflow/models/blob/master/object_detection/
g3doc/installation.md) before you start.
# # Imports
# In[ ]:
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
44
from PIL import Image
from utils import label_map_util
from utils import visualization_utils as vis_util
# # Model preparation
# Any model exported using the `export_inference_graph.py` tool can be loaded
here simply by changing `PATH_TO_CKPT` to point to a new .pb file.
#
# By default we use an "SSD with Mobilenet" model here. See the [detection
model
zoo](https://github.com/tensorflow/models/blob/master/object_detection/g3doc/d
etection_model_zoo.md) for a list of other models that can be run out-of-the-box
with varying speeds and accuracies.
# What model to download.
MODEL_NAME = 'ssd_mobilenet_v1_coco_11_06_2017'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE =
'http://download.tensorflow.org/models/object_detection/'
# Path to frozen detection graph. This is the actual model that is used for
the object detection.
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')
```

ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

45

```
NUM_CLASSES = 90
# ## Download Model
if not os.path.exists(MODEL_NAME + '/frozen_inference_graph.pb'):
print ('Downloading the model')
opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
tar_file = tarfile.open(MODEL_FILE)
for file in tar_file.getmembers():
file_name = os.path.basename(file.name)
if 'frozen_inference_graph.pb' in file_name:
tar_file.extract(file, os.getcwd())
print ('Download complete')
else:
print ('Model already exists')
# ## Load a (frozen) Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
od_graph_def = tf.GraphDef()
with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
serialized_graph = fid.read()
od_graph_def.ParseFromString(serialized_graph)
tf.import_graph_def(od_graph_def, name='')
# ## Loading label map
```

46

```
# Label maps map indices to category names, so that when our convolution
network predicts `5`, we know that this corresponds to `airplane`. Here we use
internal utility functions, but anything that returns a dictionary mapping integers
to appropriate string labels would be fine
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES,
use_display_name=True)
category_index = label_map_util.create_category_index(categories)
# ## Helper code
defload_image_into_numpy_array(image):
(im_width, im_height) = image.size
return np.array(image.getdata()).reshape(
(im_height, im_width, 3)).astype(np.uint8)
# # Detection
# For the sake of simplicity we will use only 2 images:
# image1.jpg
# image2.jpg
# If you want to test the code with your images, just add path to the images to the
TEST_IMAGE_PATHS.
PATH_TO_TEST_IMAGES_DIR = 'test_images'
# Running the tensorflowsessionTEST_IMAGE_PATHS = [
os.path.join(PATH_TO_TEST_IMAGES_DIR, 'image{}.jpg'.format(i)) for i in range(1,
4) ]
# Size, in inches, of the output images.
```

47

```
IMAGE_SIZE = (12, 8)
with detection_graph.as_default():
with tf.Session(graph=detection_graph) as sess:
for image_path in TEST_IMAGE_PATHS:
image = Image.open(image_path)
# the array based representation of the image will be used later in order to
prepare the
# result image with boxes and labels on it.
image_np = load_image_into_numpy_array(image)
# Expand dimensions since the model expects images to have shape: [1,
None, None, 3]
image_np_expanded = np.expand_dims(image_np, axis=0)
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
# Each box represents a part of the image where a particular object was
```

ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

detected.

```
boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
# Each score represent how level of confidence for each of the objects.
# Score is shown on the result image, together with the class label.
scores = detection_graph.get_tensor_by_name('detection_scores:0')
classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = detection_graph.get_tensor_by_name('num_detections:0')
# Actual detection.
(boxes, scores, classes, num_detections) = sess.run(
[boxes, scores, classes, num_detections],
feed_dict={image_tensor: image_np_expanded})
# Visualization of the results of a detection.
vis_util.visualize_boxes_and_labels_on_image_array(
image_np,
np.squeeze(boxes),
48
np.squeeze(classes).astype(np.int32),
np.squeeze(scores),
category_index,
use_normalized_coordinates=True,
line_thickness=8)
plt.figure(figsize=IMAGE_SIZE)
plt.imshow(image_np)
```

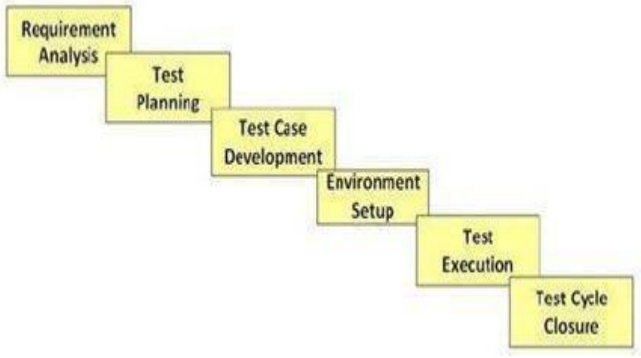
TESTING

Software testing can also be stated as the process of validating and verifying that a software program/application/product:

1. meets the business and technical requirements that guided its design and development;
2. Works as expected; and
3. Can be implemented with the same characteristics.

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development methodology adopted.

Software Test Life Cycle



Test Planning

This phase is also called Test Strategy phase. Typically, in this stage, a Senior QA
manager will determine effort and cost estimates for the project and would prepare and finalize the Test Plan

Activities:

- Preparation of test plan/strategy document for various types of Testing

Test tool selection.

- Testing Test tool selection

Test effort estimation

- Resource planning and determining roles and responsibilities.
- Training requirement

Test Environment Setup

Test environment decides the software and hardware conditions under which a work product is tested. Test environment set-up is one of the critical aspects of testing process and can be done in parallel with Test Case Development Stage. Test team may not be involved in this activity if the customer team provides

ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

the test environment in which case the test team is required to do a readiness check (smoke testing) of the givenenvironment.

Activities:

- Understand the required architecture, environment set-up and hardware and software requirement list for the Test Environment.
- Setup test Environment and test data
- Perform smoke test on the build

Test Execution

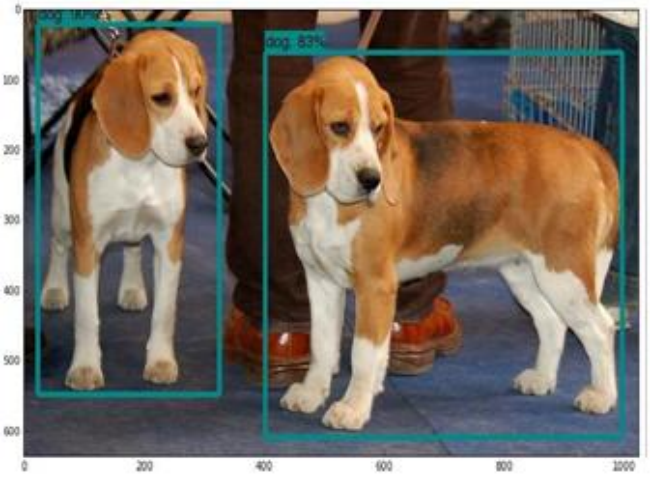
During this phase test team will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed.

Activities:

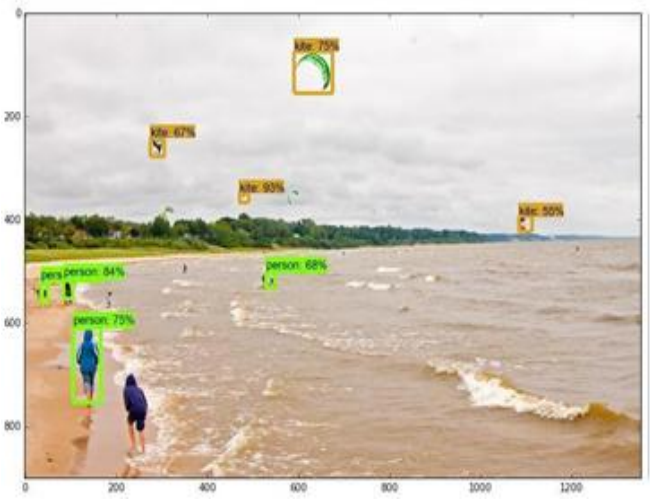
- Execute tests as per plan
- Document test results, and log defects for failed cases Map defects to test cases in RTM
- Retest the defect fixes
- Track the defects to closure

RESULTS

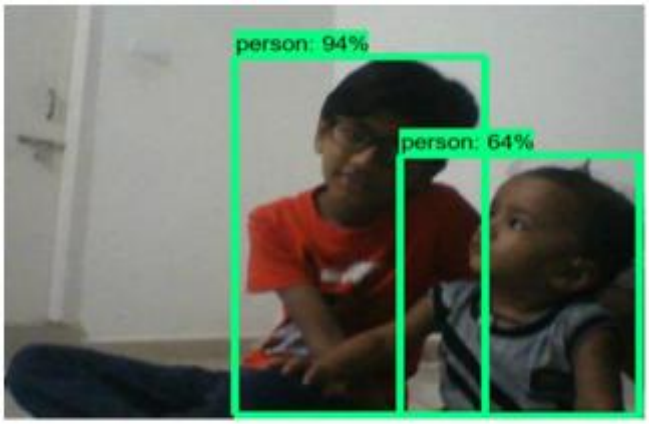
Object Detect in Image



Animal Detection from Image



Multiple Objects Detection in Image



Object Detection with Accuracy from Webcam Video

Object Diction from Webcam Video

ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

Multiple Object Diction from Webcam Video

CONCLUSION

This paper expresses the importance of deep learning technology applicationsand the impact of dataset for deep learning through the use of the faster r-cnn on newdatasets. In recent years, the technology of deep learning in image classification, objectdetection and face identification and many other computer vision tasks have achievedgreat success. Experimental data shows that the technology of deep learning is aneffective tool to pass the man-made feature relying on the drive of experience to thelearning relying on the drive of data. Large data is the base of the success of deeplearning, large data just as fuel to the rocket for deep learning. More and moreapplications are continually accumulating increasingly rich application data, which is critical to the further development and application of deep learning. However, the qualityof the data affects the deep learning in deed, of course, in addition to these real data,maybe we can also consider some of synthetic data to increase the amount of data in thefurther.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS,2012.

[2] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In ECCV,

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR, 2016. 71

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ―Rich feature hierarchies for accurate object detection and semantic segmentation,‖ in CVPR, 2014.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. ImageNet: A large-scale hierarchical image database. In CVPR, 2009.

[7] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. FeiFei. ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012).

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. IJCV, 2010.

[10] Lin, Tsung Yi, et al. Microsoft COCO: Common Objects in Context. Computer Vision – ECCV 2014. Springer International Publishing, 2014:740-755.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.

[12] R. Girshick. Fast R-CNN. arXiv:1504.08083, 2015.

[13] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NIPS, 2015.

ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[15] Ross Girshick. Fast R-CNN. In International Conference on Computer Vision (ICCV), 2015.

[16] ShaoqingRen, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards realtime object detection with region proposal networks. In Advances in Neural Information Processing Systems (NIPS), 2015.

[17] Joseph Redmon, SantoshDivvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[18] Wei Liu, DragomirAnguelov, DumitruErhan, Christian Szegedy, Scott Reed, ChengYang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In ECCV, 2016.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for largescale image recognition. arXiv preprint arXiv:1409.1556, 2014.