College of Applied Business

Tribhuvan University

Institute of Science and Technology

Object Detection using Tensorflow

A Project Report

Submitted to

Department of Computer Science and Information Technology

College of Applied Business

In partial fulfillment of the requirements for the Bachelor's Degree in Computer Science and Information Technology

Submitted by

Kashyap Raj Karki

Kripa Pokhrel

Kushal Ghimire

SUPERVISOR'S RECOMMENDATION

This is to certify that the report entitled "Object Detection with Tensorflow" submitted by Kashyap Raj Karki, Kripa Pokhrel and Kushal Ghimire is prepared under my supervision in partial fulfillment of the requirements for the degree of Bachelor's in Computer Science and Information Technology be processed for the evaluation.

SUPERVISOR: **Date:**

(Tekendranath Yogi)

DECLARATION

We, Kashyap Raj Karki, Kripa Pokhrel, and Kushal Ghimire of 'College of Applied Business, Gangahiti, Kathmandu', of BSc. CSIT [Semester III], officially announce that our project, titled 'Object Detection,' has been successfully finished. To the best of our knowledge, the content supplied was written by us and on our own initiative, and no part has been plagiarized without citations. This report's findings have not been presented to any other institutions or colleges.

Thank You,

Kashyap Raj Karki

Kripa Pokhrel

Kushal Ghimire

ACKNOWLEDGEMENT

Without the combined efforts of numerous people, this project would not have been feasible. We would like to convey our heartfelt gratitude to everyone who has contributed to this initiative. We would like to express our gratitude to our supervisor, Tekendra Nath Yogi sir, for his consistent monitoring in assisting us in moving forward with our project.

ABSTRACT

The computer is capable of a lot of things, and it's only becoming better at them. Robots, as well as self-driving cars, are now probable. And "Computer Vision" is required for it to be capable of such action. Machine learning and artificial intelligence have come a long way and continue to progress at a rapid pace. With the help of machine learning, a variety of tasks are now possible. If we brainstorm for a while, we can come up with a lot of project ideas just using the concept of "Object Detection." For instance, a sign language processor, a self-navigating drone, and facial recognition are some of its application. We can't deny the utility and constraint of "Object Detection" because it has such a broad range of applications.

The goal of this project was to use 'Tensorflow Object dection' API for machine learning to create a workable object detection model capable of recognize its makers and some sign and hand gestures.

Table of Contents

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Computer vision is a very popular topic. People are very much interested on its applications such as a self-driving car and much more. And there have been many successful attempts on the Machine learning approach to computer vision say Tesla's auto driving for example. Our project "Object Detection" barely scratches the surface of the Computer vision but it does allow us a starting point into the files of computer vision. The Machine is trained by showing it series of images and selecting the object we want the machine to recognize.

1.2 Background

In the 1960s, colleges began working on computer vision in earnest, seeing it as a stepping stone to artificial intelligence. Early researchers were tremendously hopeful about the future of these domains and pushed artificial intelligence as a technology with the potential to change the world. Some believed that within a generation, a machine as intelligent as a human would be constructed. Researchers received millions of dollars in governmental and private funding as a result of the publicity. Around the world, research institutes sprang up. However, a failure to meet lofty expectations stifled the international effort to build artificial intelligence.

1.3 Problem Statement

The application of machine learning approaches to computer vision applications such as picture registration, 3D reconstruction, segmentation, and classification, motion tracking, and object recognition has seen great growth in recent years. ML algorithms based on training data are the best at solving difficult computational data analytics challenges. The current state of computer vision allows for the unrestricted recognition and tracking of single object types (faces, people, and automobiles). It enables smart cameras to recognize smiling people, pedestrian identification, surveillance applications, and image-based online searches, among other things. In my opinion, the computer vision has made the term AI more broader and applicable and has solved many problems of modern day AL's framework.

1.4 Objective

The major objectives of the project are listed below:

- To make useful decision based on sensed images.
- To construct 3D structure form 2D images.

1.5 Related Areas of Object Detection

The major areas of object detection are listed below:

- Image processing
- Artificial Intelligence
- Optical Character Reader
- Unmanned Autonomous Vehicle(UAV)
- Inspecting Products

1.6 Report Organization

This report consists of six chapters where each chapter gives information about different related topics. The first chapter is the introduction where the entire project is introduced and defined. This chapter has several parts such as introduction, background, problem statement, objective, report organization. The second chapter deals with the development of modules and the fouth chapter contains feasibility analysis. The fifth chapter deals with implementation and testing process. The sixth chapter concludes the project.

CHAPTER 2

OBJECT DETECTION MODULES

2. Development of Object Detection Modules

The object detection conglomerates classification and localization. Current object detectors may be classified into two types: networks that separate the tasks of finding the location of objects and their categorization, such as the Faster R-CNN, and networks that predict bounding boxes and class scores simultaneously, such as the YOLO and SSD networks.

Overfeat [2.1] was the first deep neural network for object detection. They used CNNs to create a multi-scale sliding window technique and demonstrated that object detection enhanced image categorization as well. They were quickly followed by R-CNN: CNN features in regions [2.2]. The authors suggested a methodology that combined related pixels into regions using selective-search to generate region ideas. Each region was fed into a CNN, which resulted in a feature vector with a high dimensionality. As illustrated in Figure 2.1, this vector was subsequently employed for the final classification and bounding box regression.

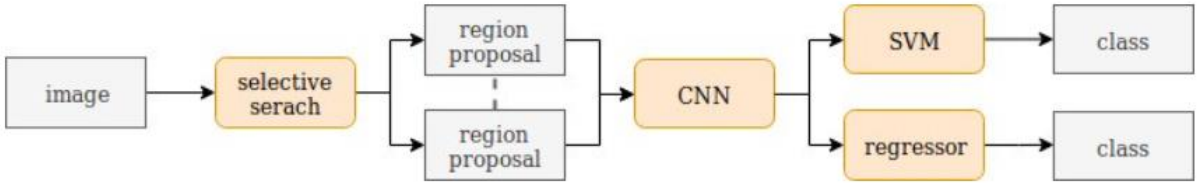

Figure 2.1 Architecture of R-CNN Network

It outperformed the Overfeat network by a huge margin, but it was also highly slow, due to the time-consuming nature of proposal generation utilizing selective-search, as well as the requirement to feed each proposal through a CNN. Fast R-CNN [2.3], a more complex approach, created region proposals with selective-search but passed the entire image through a CNN. ROI pooling was used to pool the region proposals directly on the feature map. Figure 2.2 shows how the pooled feature vectors were input into a fully connected network for classification and regression. Fast R-CNN, like R-CNN, used selective-search to create region ideas.

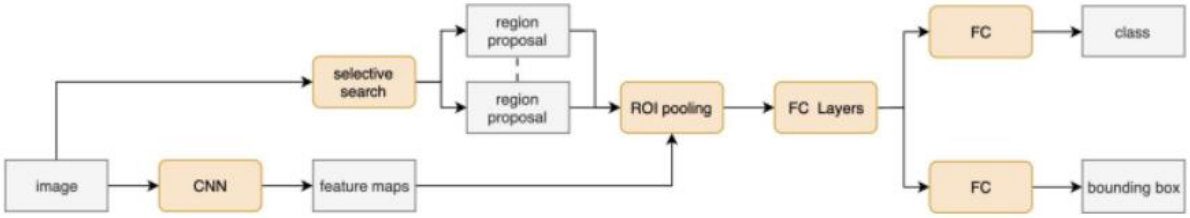

Figure 2.2 Architecture of Fast R-CNN Network

Faster R-CNN [2.3] proposed a revolutionary region proposal network, which was fused with the Fast R-CNN architecture to substantially speed up the process. R-FCN , a region fully

convolutional network that used position-sensitive score maps instead of a pre-region sub network, was another method for detecting objects in images.

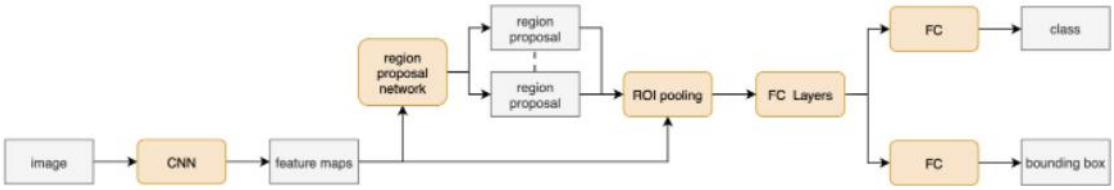

Figure 2.3 Architecture of Faster R-CNN Network

The YOLO network altered the design of object detection networks. It takes a completely different strategy than the previous models and can predict both class scores and bounding boxes at the same time. The proposed model divided the image into a grid, with each cell predicting a confidence score for the presence of an object together with the bounding box coordinates. This enables YOLO to make predictions in real time. The authors also produced two more versions, YOLO9000 and YOLOv2, the former of which could predict over 9000 categories and the latter of which could handle larger photos. The single shot detector, SSD [2.4], is another network that predicts classes and bounding boxes at the same time. It is comparable to YOLO but used multiple aspect ratios per grid cell and more convolutional layers to improve prediction. We are using SSD model for detecting in this project.

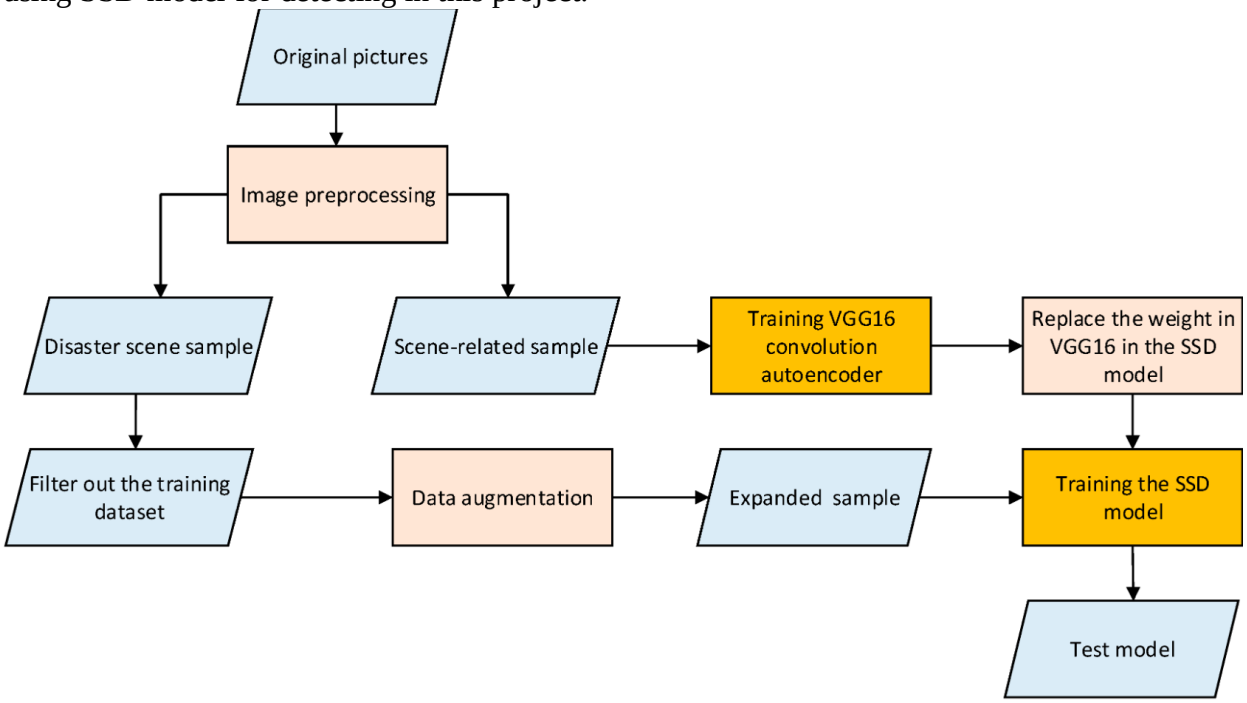

Figure 2.4 Architecture of SSD Network

Chapter 3
FEASIBILITY STUDY

3. Feasibility Study

The feasibility study assisted us in determining the model's strengths and weaknesses. This feasibility study's contents and recommendations aided us in deciding how to continue. The following are some examples of feasibility studies:

3.1 Technical Feasibility

Technically, the model will be developed using Python as core programming language. The object detection model SSD(Single Shot Detector) with 'Tensorflow' will be used to train, label and lastly recognize the image. The technologies that are required for the development of the system are readily available.

3.2 Operational Feasibility

The model is designed to recognize some images and is not capable of recognizing the entire real world object. It may not meet the requirement of every user, but it will be able to recognize generic signs and gestures along with some familiar faces and objects of easy findings such as pen, cup.

2.1.3 Economic Feasibility

This model requires local data set from your webcam so it requires less amount of space to store it. Also the model is trained with Tensorflow's SSD model using CUDA and CUDNN, which requires dedicated NVIDIA graphics to train your image. However, you can choose to use only your CPU which will require more processing time.

CHAPTER 4

SYSTEM DESIGN

4.1 Methodology

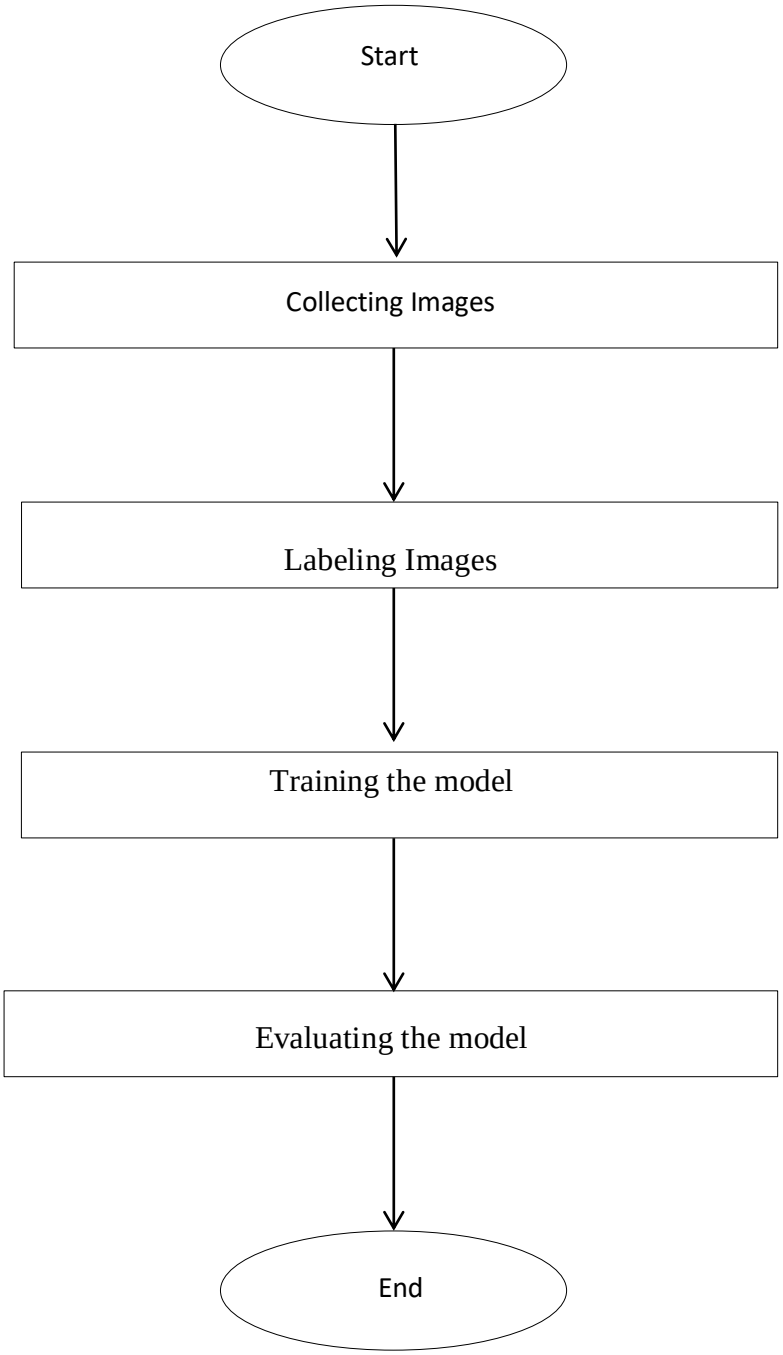

Figure 4.1.1 Flowchart of the entire process

4.1.1 Collecting Images

We used our computer's webcam for collecting data. Those images are our dataset. The dataset consists of 5 images of each person/object/gesturs.

Firstly, we imported opencv for image processing the we imported os for file manipulation.We also imported time for providing pause between each session of image collection.

4.1.2 Labeling Images

The data or the image that we took were labeled using the library 'labelmg' which labeled particular object in the entire image for training purpose. The selected part of image is saves as an xml file.

4.1.3 Training the model

The labeled image is then moved on towards training with the help of Tensorflow's SSD module in which a single label was trained for 2000 times. We chose SSD that had balanced speed and precision for training purpose.

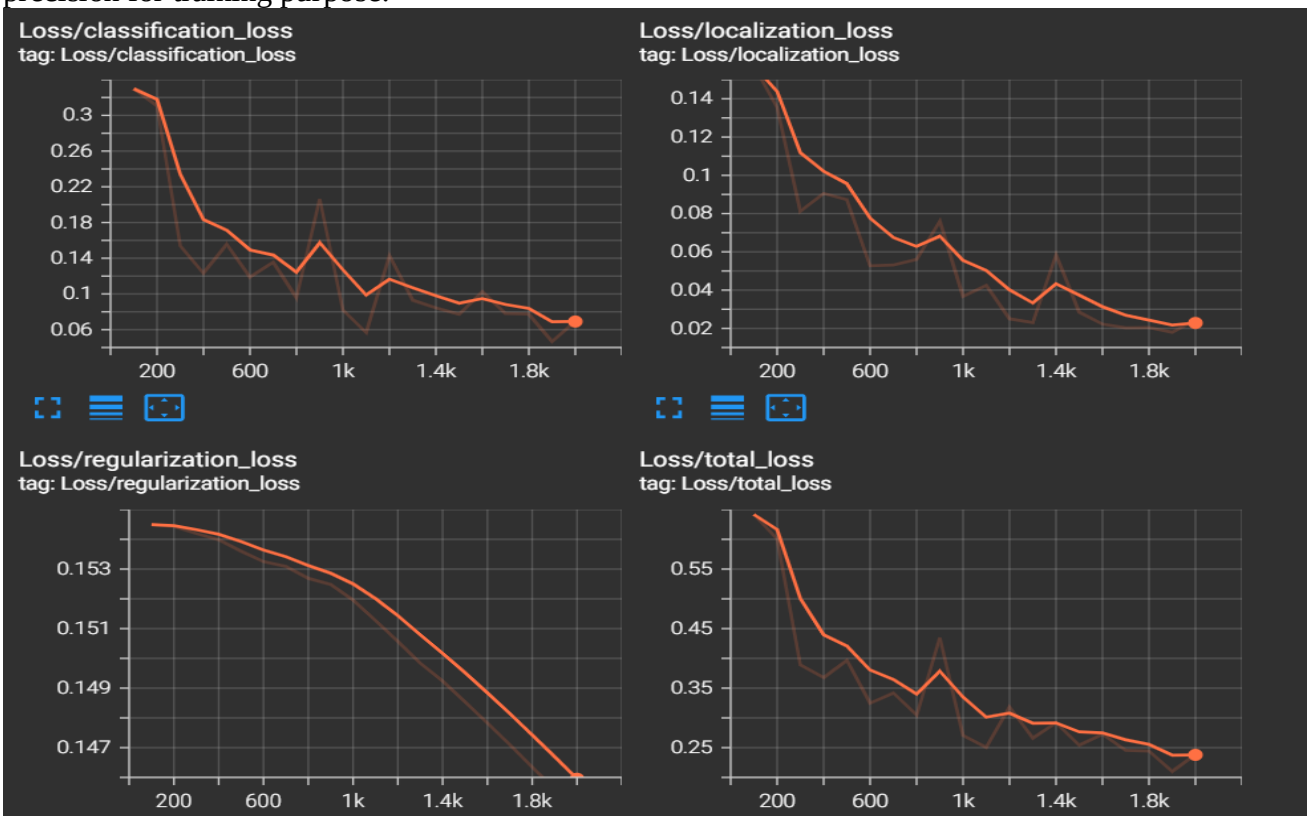

Figure 4.1.3.1 Local Loss

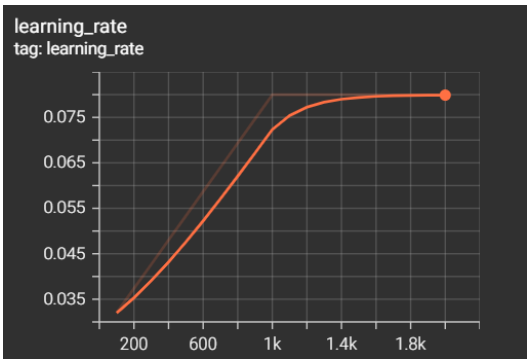

Figure 4.1.3.2 Learning Rate

4.1.4 Evaluating the model

The model was evaluated using matrices such as precision and recall using Tensorboard. Precision and recall are two extremely important model evaluation metrics. While precision refers to the percentage of your results which is relevant, recall refers to the percentage of total relevant results correctly classified by your algorithm.

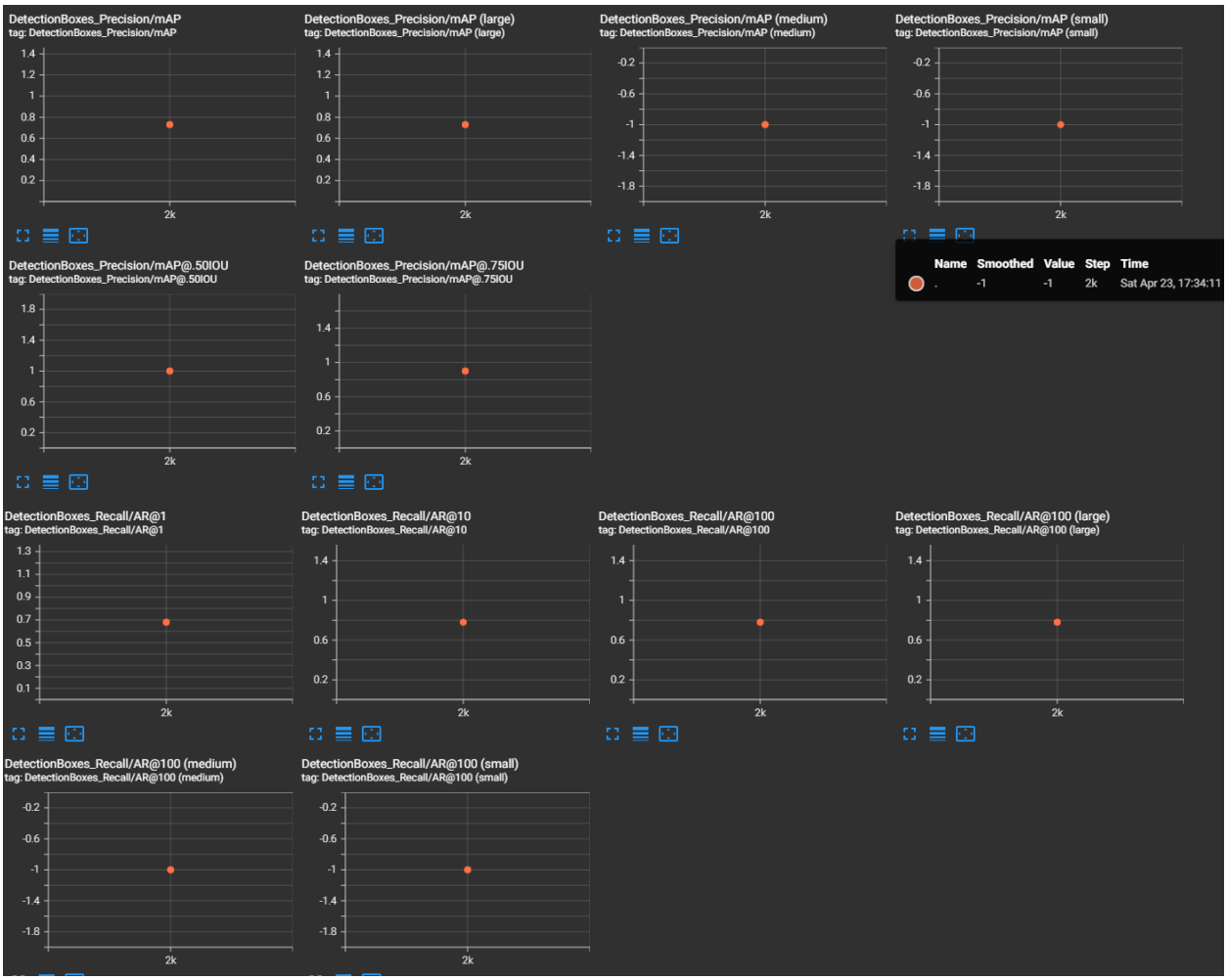

Figure 4.1.4 Precision and Recall matrices

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation

An object detection model was developed which can detect generic object and also capable of recognizing some faces and hand gestures. To develop a working system, implementation was done in a two stages. First stage was image collection and the another stage was training and detection.

5.1.1 Tools Used

1. Python

Python was used as a core programming language to develop the model.

2. OS

OS was imported for file manipulation.

3. Labelimg

Label image was used for labeling the collected image

4. Matplotlib

Matplotlib was used for statistical data visualization.

5. OpenCV

OpenCV was used for image processing.

5.2 Testing

Precision and Recall are taken to validate the performance of the model.

CHAPTER 6

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

The SSD model is one of the fastest and efficient object detection models for multiple categories. And it has also opened new doors in the domain of object detection

6.2 Recommendation

This model can be further enhanced by using more number of images as we have only used 5 images. The more variation in the image the better the accuracy we will get. We also can improve its performance by increasing the number of times we train it. We trained the model for 2000 times. You could certainly increase the number of times it is trained.

References

[1]"Single Shot Detector (SSD) + Architecture of SSD", *OpenGenus IQ: Computing Expertise & Legacy*, 2022. [Online]. Available: https://iq.opengenus.org/single-shot-detector/. [Accessed: 23- Apr- 2022]

[2]*Arxiv.org*, 2022. [Online]. Available: https://arxiv.org/pdf/1512.02325.pdf. [Accessed: 23- Apr- 2022]

[3]"Understanding Object Detection", *Medium*, 2022. [Online]. Available: https://towardsdatascience.com/understanding-object-detection-9ba089154df8. [Accessed: 23- Apr- 2022]

[4]"Precision vs Recall | Precision and Recall Machine Learning", *Analytics Vidhya*, 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/. [Accessed: 23- Apr- 2022

APPENDIX

Code: for image collection

```
!pip install opencv-python
# Import opencv
import cv2
# Import uuid
import uuid
# Import Operating System
import os
# Import time
import time
labels = ['thumbsup','thumbsdown','livelong','thankyou','peace','rock','pen','cup','kashyap','kushal','kripa']
number_imgs = 5
IMAGES_PATH = os.path.join('Tensorflow', 'workspace', 'images', 'collectedimages')
if not os.path.exists(IMAGES_PATH):
    if os.name == 'posix':
        !mkdir -p {IMAGES_PATH}
    if os.name == 'nt':
         !mkdir {IMAGES_PATH}
for label in labels:
    path = os.path.join(IMAGES_PATH, label)
    if not os.path.exists(path):
        !mkdir {path}
for label in labels:
    cap = cv2.VideoCapture(0)
    print('Collecting images for {}'.format(label))
```

```
    time.sleep(5)
    for imgnum in range(number_imgs):
        print('Collecting image {}'.format(imgnum))
        ret, frame = cap.read()
        imgname = os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(imgname, frame)
        cv2.imshow('frame', frame)
        time.sleep(2)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
!pip install --upgrade pyqt5 lxml
LABELIMG_PATH = os.path.join('Tensorflow', 'labelimg')
if not os.path.exists(LABELIMG_PATH):
    !mkdir {LABELIMG_PATH}
    !git clone https://github.com/tzutalin/labelImg {LABELIMG_PATH}
if os.name == 'posix':
    !cd {LABELIMG_PATH} && make qt5py3
if os.name =='nt':
    !cd {LABELIMG_PATH} && pyrcc5 -o libs/resources.py resources.qrc
!cd {LABELIMG_PATH} && python labelImg.py
```

Code: for training and detection

```
import os

CUSTOM_MODEL_NAME = 'my_ssd_mobnet'

PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'

PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x32
0_coco17_tpu-8.tar.gz'

TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'

LABEL_MAP_NAME = 'label_map.pbtxt'

paths = {

    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),

    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),

    'APIMODEL_PATH': os.path.join('Tensorflow','models'),

    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),

    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),

    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),

    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),

    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME),

    'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME,
'export'),

    'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME,
'tfjsexport'),

    'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME,
'tfliteexport'),

    'PROTOC_PATH':os.path.join('Tensorflow','protoc')

 }

files = {

    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MODEL_NAME,
'pipeline.config'),
```

```
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
if os.name=='nt':
    !pip install wget
    import wget
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
# Install Tensorflow Object Detection
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && cp object_detection/packages/tf2/setup.py . && python -m pip install .

if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
```

```
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_detection\\packages\\tf2\\setup.py setup.py && python setup.py build && python setup.py install

    !cd Tensorflow/models/research/slim && pip install -e .

VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model_builder_tf2_test.py')

# Verify Installation

!python {VERIFICATION_SCRIPT}

!pip install tensorflow-gpu –upgrade

!pip install protobuf matplotlib

!pip install pyparsing==2.4.7

!pip install --upgrade witwidget –q

import object_detection

if os.name =='posix':

    !wget {PRETRAINED_MODEL_URL}

    !mv {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}

    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}

if os.name == 'nt':

    wget.download(PRETRAINED_MODEL_URL)

    !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}

    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}

labels = [{'name':'Thumbsup', 'id':1}, {'name':'Thumbsdown', 'id':2}, {'name':'Peace', 'id':3}, {'name':'Livelong', 'id':4}, {'name':'Rock','id':5}, {'name':'Kushal','id':6},{'name':'Kripa','id':7} {'name':'Kashyap','id':8}]


with open(files['LABELMAP'], 'w') as f:

    for label in labels:

        f.write('item { \n')
```

```
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
if not os.path.exists(files['TF_RECORD_SCRIPT']):
    !git clone https://github.com/nicknochnack/GenerateTFRecord {paths['SCRIPTS_PATH']}
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'train')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'train.record')}
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'test')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'test.record')}
if os.name =='posix':
    !cp {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
if os.name == 'nt':
    !copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
```

```
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"

pipeline_config.train_input_reader.label_map_path= files['LABELMAP']

pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] =
[os.path.join(paths['ANNOTATION_PATH'], 'train.record')]

pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']

pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] =
[os.path.join(paths['ANNOTATION_PATH'], 'test.record')]

config_text = text_format.MessageToString(pipeline_config)

with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:

    f.write(config_text)

TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection',
'model_main_tf2.py')

command = "python {} --model_dir={} --pipeline_config_path={} --
num_train_steps=2000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])

print(command)

python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace\
models\my_ssd_mobnet --pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet\pipeline.conf
ig --num_train_steps=2000
!{command}

command = "python {} --model_dir={} --pipeline_config_path={} --
checkpoint_dir={}".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'], paths['CHECKPOINT_PATH'])

print(command)

python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace\
models\my_ssd_mobnet --pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet\pipeline.conf
ig --checkpoint_dir=Tensorflow\workspace\models\my_ssd_mobnet
!{command}
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)
```

```
# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-3')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
MAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'peace.dd727feb-c2b2-11ec-9441-9828a63c9425.j
pg')
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

while cap.isOpened():
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections

    # detection_classes should be ints.
    detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
                image_np_with_detections,
                detections['detection_boxes'],
                detections['detection_classes']+label_id_offset,
                detections['detection_scores'],
                category_index,
                use_normalized_coordinates=True,
                max_boxes_to_draw=5,
                min_score_thresh=.8,
                agnostic_mode=False)

    cv2.imshow('object detection',  cv2.resize(image_np_with_detections, (800, 600)))
```

```
if cv2.waitKey(10) & 0xFF == ord('q'):
    cap.release()
    cv2.destroyAllWindows()
    break
```