PROJECT REPORT

on

"OBJECT DETECTION USING TENSORFLOW"

I- Master of Computer Applications

For the academic year

2020-21

Submitted by

2020178011 Dennis Churchill J

2020178014 Ebenezer Isaac Veeraraju

2020178034 Mahmood Raj Mohamed

2020178056 Suvetha E

INTRODUCTION

Object detection is a general term for computer vision techniques for locating and labeling objects. Object detection techniques can be applied both to static images or moving images. Computer vision techniques are already in wide use in every field today, as they can offer valuable insight about movement of individual object or whole items to training, help in tracking and help visualizing progression during object movement [1].

Open source libraries such as OpenCV's DNN library and TensorFlow Object Detection API offer easy-to-use, open source frameworks where pre-trained models for object detection (such as ones downloadable from the TensorFlow model zoo) reach high accuracy in detecting various object from humans to tv monitors [2]Object detection is a term related to computer vision and image processing techniques for locating and annotating all the potential objects in the images. The images can be either static pictures or moving frames.

The task of object detection is closely related to two other problems: image classification and object localization. Image classification also referred to as image recognition, is about classifying an image to a particular class out of all the possible classes.

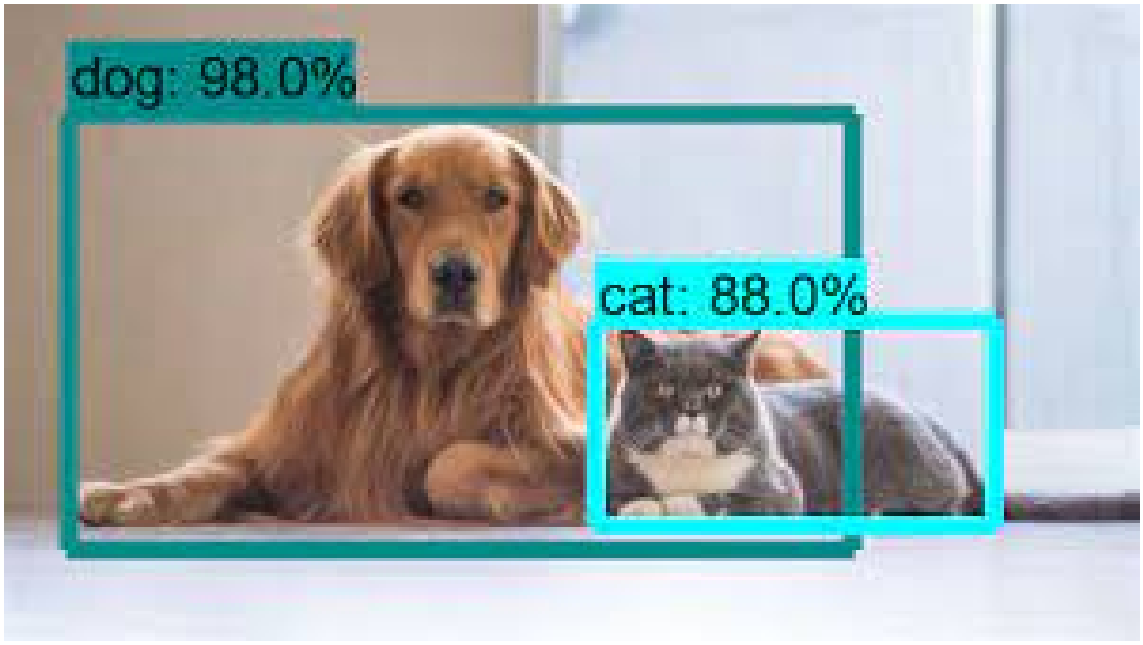
Object localization is more advanced than image recognition, and it demands to localize the labeled object in an image. Object detection is the most complex of all—it involves labeling and localizing multiple objectsin an image. Object detection techniques are extensively used for applications related to face detection, locating pedestrian on streets or players on football grounds, and detecting vehicles on roads.

A major constraint of a deep CNN is that it requires a large amount of annotated datasets for training them. To overcome the problem of enormous training datasets, several public datasets such as the ImageNet database, the Common Objects in Context (COCO) dataset, and the Open Images dataset are readily available. Moreover, the development of Graphical Processing Units (GPUs) helps in training the computationally expensive deep neural networks. Tensorflow Object Detection API,1 OpenCV's DNN library,2 and Microsoft Cognitive Toolkit3 provide open-source frameworks to construct, train and deploy object detection models.

We are using Tensorflow because of its popularity and ease of use. Tensorflow Object Detection API also presents pre-trained models for object detection. The pre-trained models are trained on some of the public datasets such as COCO, Kitti, Open

Images, and Atomic Visual Actions (AVA) v2.1.

The performance of the pre-trained models is sufficient for recognizing objects, such as a person, television and cars, but not good enough for detecting the food classes relevant for our purposes. Moreover, the pre- trained models do not even detect some of the food classes expected in our application. Therefore, it is essential to fine-tune the existing models for our particular datasets and application requirements.



Artificial Neural Network (ANN)

Artificial Neural Network, loosely inspired by the neural structure of our brain, consists of a network of connected nodes known as neurons. Neurons are the central processing unit of the neural networks, and they are connected just like neurons and synapses in a biological brain. The weighted neural connections assist in transmitting signals (features) from one neuron to another.

Artificial neural networks learn by examples rather than programming them with task-specific rules. For instance, in visual pattern recognition, they might learn with training examples of handwritten digits from 0 to 9 and use this knowledge to identify digits in other unseen samples.

Artificial neural networks (ANN) mostly consist of three types of layers, an input layer, an output layer, and one or more hidden layers. The first hidden layer learns simple, and basic features and passes them to the second hidden layer. Based on the inputs from theprevious layer, the second layer learns features which are more complex and at a more abstract level than the first one. Similarly, layers farther in the network can learn more complicated features and so on. In this way, a multi-layered network can be used effectively to solve sophisticated problems like image classification, object detection, and others.

Deep Learning:

Deep Learning, a subfield of machine learning and artificial intelligence, is related to the training of computational models that are composed of multi-layered artificial neural networks. The multi-layered ANN is known as a deep neural network (DNN). Deep in deep neural networks corresponds to the depth of the network. Deep networks usually have more than two layers of hidden neurons between the input and output layers.

The deep neural networks have enhanced the state-of-the-art accuracy in image classification, object detection, speech recognition, language translation, and other areas considerably. Deep learning methods are based on learning representations (features) from data, such as text, images, or videos, rather than implementing task-specific algorithms. Learning can either be unsupervised or supervised. However, most of the practical systems deploy supervised learning to leverage the benefits of deep learning. Supervised learning, in essence, means learning from labeled data.

Convolutional Neural network (CNN):

Convolutional Neural Network (CNN) is the most common class of architectures used for deep learning. CNN works similarly as artificial neural network except it has a series of convolutional layers at the beginning. CNN is widely used for visual recognition tasks such as image classification, object detection, and speech recognition.

Data collection and Preprocessing

Typically one of the significant challenging aspects of a deep CNN is to collect labeled examples to train image classifiers. In general, supervised learning demands thousands of annotated training examples. To meet the requirement of large training data and benchmark evaluation, the number of publicly available datasets are proliferating.

In the context of computer vision, the available datasets contain large sets of images with additional information such as annotations, segmentation masks or other contextual data. Since there are several such datasets available for training and validation, it is essential to select the one which serves our purposes best. We selected the user defined database because it contains our own images for each class along with bounding box data for some images. Moreover, the user database has been used successfully for various applications of image classification which is an integral.

Dataset Creation

After completing the annotation process, we divided each class dataset into two sets: training dataset, and test dataset. Then, the corresponding datasets for all the classes are merged—three training datasets for three classes. We then combined the three training datasets into one training dataset with the training examples from all the classes. Similarly, we combined the validation and test datasets as well.

TF Record Creation

To fine-tune a pre-trained Tensorflow object detection model, the training and validation datasets are converted into a TFRecord format. TFRecord is a standard format, supported by Tensorflow for training object detection models. Tensorflow models are pre-trained for multiple datasets such as COCO, Kitti, and Open Images.

Object Detection with SSD:

The pre-trained 'ssd_inception_v2_coco_2017_11_17' is fine-tuned with the training and validation datasets created earlier using the provided configuration file as the basis of our fine-tuning. The configuration file requires a checkpoint file to initiate the finetuning process which is already provided by Tensorflow for 'ssd_inception_v2_coco_2017_11_17'. The configuration file also requires training record and test record file locations. Training record is a TFRecord file created for training dataset, and a test record is a file created for validation dataset. Another requisite of the configuration file is the location of an object label map file. The structure of an object label map file is similar, and ends with '.pbtxt' extension. The training record and test record file locations are added along with the object label map file. Other important parameters mentioned in the configuration file include:

• number of layers: 6

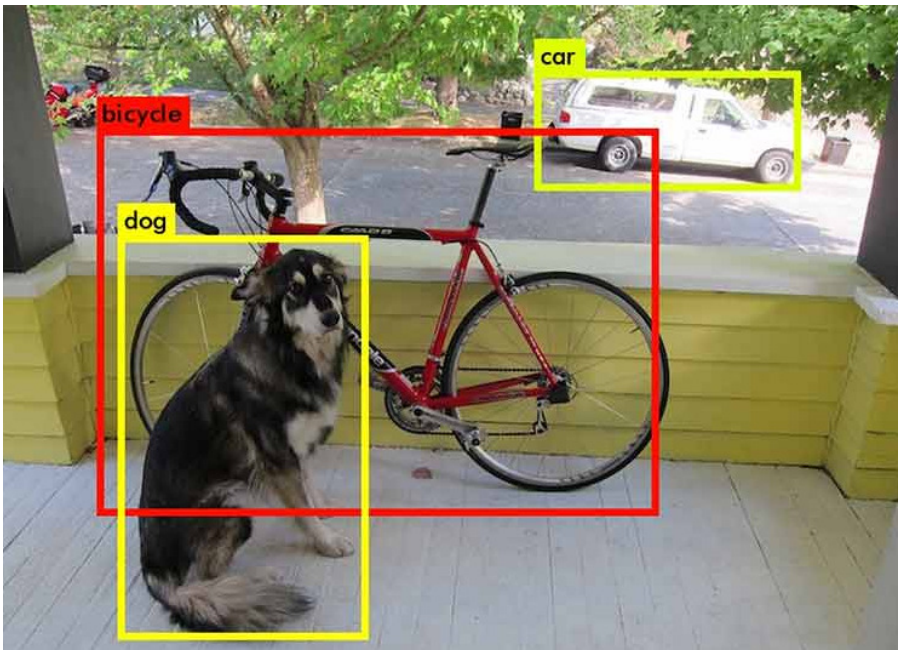
• aspect ratios: 0.33, 0.5, 1.0, 2.0, and 3.0

• learning rate: initial value is 0.004 with a decay factor of 0.95 after 800,720

steps (iterations) of training

• data augmentation options: random horizontal flip and ssd random crop

• batch size: 24



NEURAL NETWORKS:

Object detection using complex machine learning techniques is possible due to the recent advances in image classification using convolutional neural networks and improvements in hardware. Artificial Neural Networks A neural network is a group of neurons or nodes connected to each other. Each node can receive a signal from its input nodes, process it, and pass it on to its output nodes.In an Artificial Neural Network (ANN), the nodes are structured in layers, where there is one input layer, one output layer, and one or more hidden layers in between. When a node receives a signal, it processes it using an activation function, which, given the input, defines the output of the node. The goal is to get the nodes to work together such that the network as a whole gives the correct output.

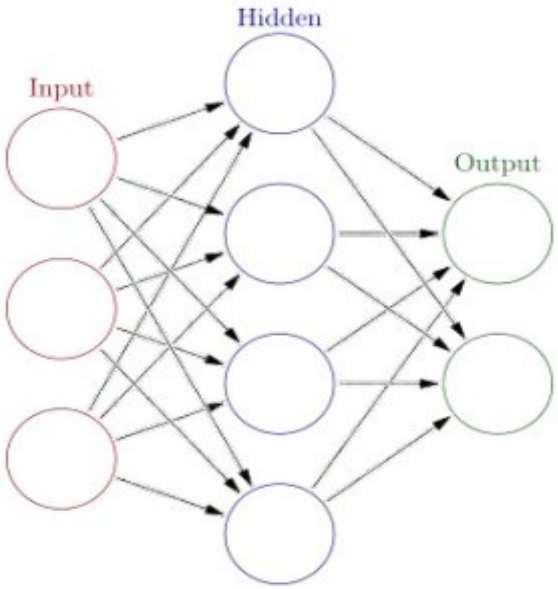


Figure.1 Artificial Neural Network

Training of Neural Networks

The process of training neural networks is to tune the parameters of the activation function of each individual node such that the final output of the network is correct. The input nodes passes on the values of the object features to the hidden layer, which processes the values using the activation function and passes on the value to the output nodes. The output nodes would be the object type, e.g.Mobile and person. Ideally, if the input is in fact a Mobile, we want the value ofthe mobile output node to get the value 1 and the person output node to get the value 0.

Then, by using gradient descent, we can change the parameters in all the activation functions in a way that decreases the loss. This is done using backpropagation, which is an algorithm that calculates how the parameters in each activation function should be changed in order to reduce the loss. The idea is that doing this many times makes the network able to classify objects correctly, also on new data.

Convolutional Neural Networks

A convolutional neural networks is a type of artificial neural networks, and consist of nodes with activation functions structured into layers. The main difference is that it is assumed that the input is an image, and use this assumption to do operations that are tailored for images, and improve both runtime and accuracy for image classification. If we want to process an image through an artificial neural network, we would transform the image into a vector, where each row of the image now is in one large row. That would mean that color images, which has three channels (RGB) of size 256 x 256 would have 256 x 256 x 3 = 196,608 neurons in the first hidden layer. Convolutional neural networks reduces the number of neurons by using convolutional layers and pooling layers.

Convolutional Layers

A convolutional layer consists of a set of filters. When a convolutional layer receives an image input, it slides, or convolves, filters over all the pixels in the input image and output the dot product of the filter and the image at the filters position. This will create an activation map, where we can see the response of the filter at different positions. During training the different filters will change in order to detect different features in the input image. Imagine we want to classify images of handwritten ones and

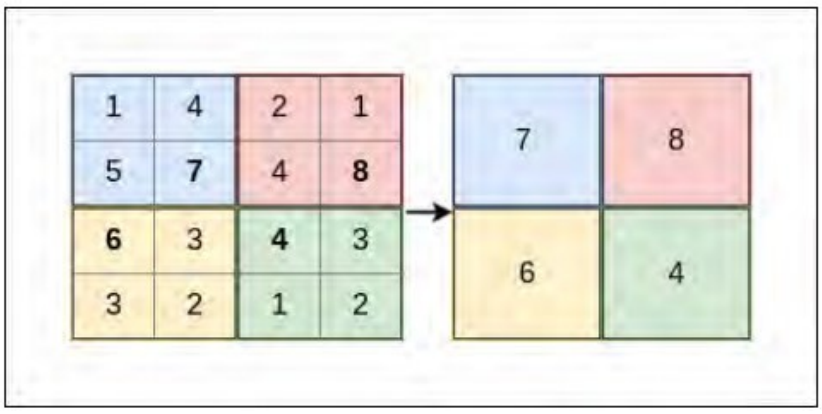
zeros. Some filters may aim to detect the rounded shape of the zeros, and will therefore output a large number if it detects a rounded shape. The dot product, or the activation of the two filters at this position in the image is $(30\cdot1)+(10\cdot1)+(10\cdot1)+(10\cdot1) = 60$ for filter 1, and $(30\cdot1)+(30\cdot1)+(30\cdot1)+(30\cdot1)+(30\cdot1) = 150$ for filter 2. This means that the activation map for filter 2 is much higher than for filter 1 at this position. For a different subregion the result would be different. When we have several different filters they will activate at different positions and thus have different activation maps. Combining the different activation maps for the different filters tells us something about the image as a whole. During training of a convolutional neural network, the network is trained to draw conclusions of what is in the image based on which filters activate at which positions.

Pooling Layers

The pooling layers are downsampling layers. They work by sliding boxes of e.g. 2x2 over the pixels of the image, and process the values to output a single value for those 4 pixels. There are different ways to process the pixels, where max pooling is a commonly used method.

Max pooling transforms the pixels inside the sliding box into one pixel with the same value as the highest pixel value in the original box. By downsampling the image the number of parameters is reduced, which in turn reduces the computational cost of the algorithm.

The pooling layers usually follow a convolutional layer, and can therefore help the next convolutional layer pick up features the previous layer could not. Pooling layers can also help preventing overfitting, which is when a classification algorithm performs well on training data, but badly on test data.



Pooling Layers

Tensorflow Object Detection API

Google's Tensorflow Object Detection API is an open source framework for object detection; it is based on Tensorflow and allows a user to define, train and utilize the models for object detection. The Tensorflow open source software library, developed by Google Brain Team, utilizes data flow graphs for numerical computations. Data flow graphs consist of two important components, nodes, and edges. Nodes represent the mathematical computations (operations), and edges represent tensors (multi-dimensional arrays) that flow between the nodes. Tensorflow is widely used to develop applications, with deep learning.

Another interesting feature of Tensorflow is that it supports a tool known as Tensorboard for in-depth visualization of models during the training process. Tensorboard provides a web interface for visualization of computational graphs and to understand how the parameters and performance change while training a model. As mentioned earlier, Tensorflow Object Detection API provides multiple pretrained models such as SSD model with MobileNet, Faster R-CNN model with ResNet, and R- FCN model with ResNet on different datasets.

SOFTWARE:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until July 2018.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python features a comprehensive standard library, and is referred to as "batteries included".

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model. Python and CPython are managed by the non-profit Python Software Foundation.

Features

Python is a multi-paradigm programming language. Object-oriented programming

and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

Applications

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

PYTHON CODING:

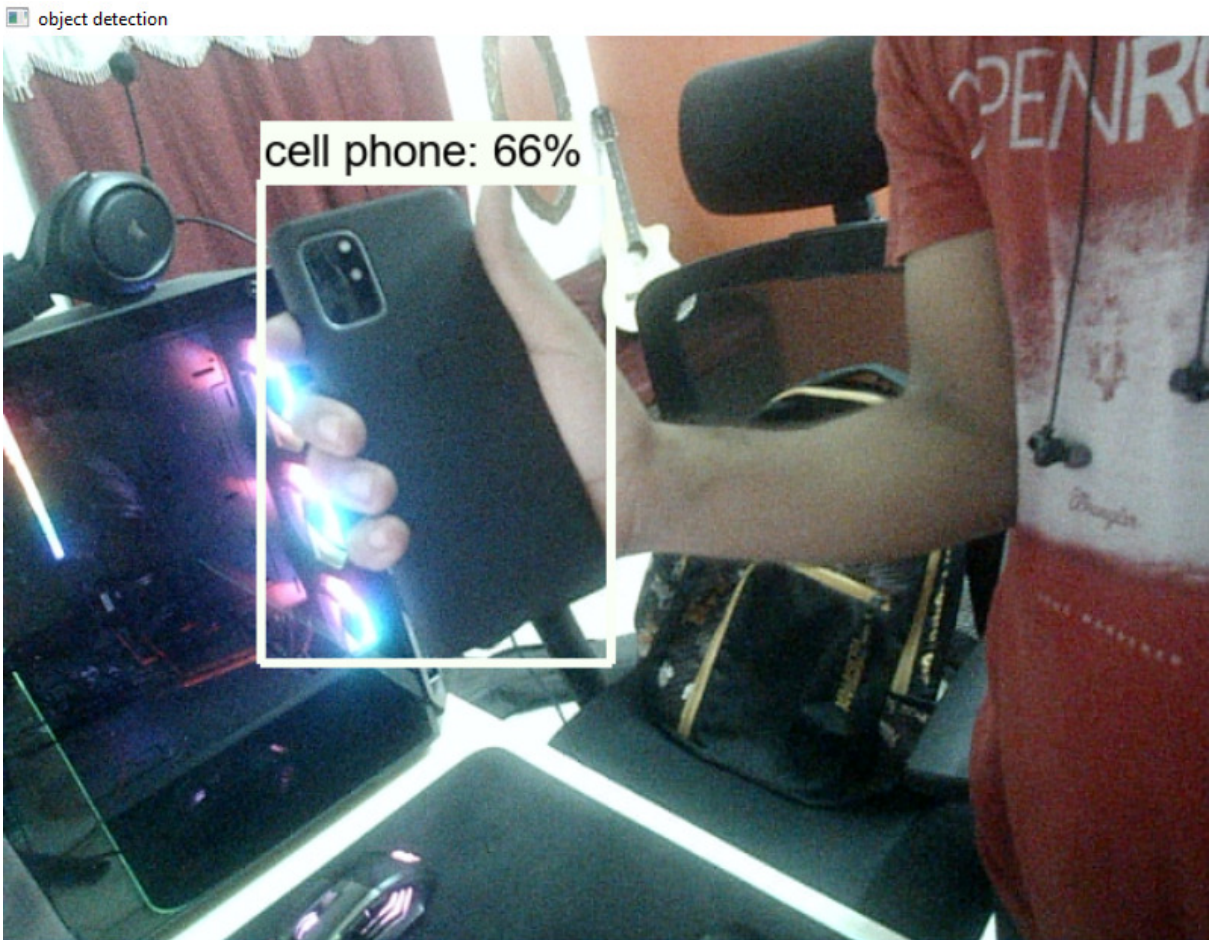
OBJECT DETECTION.PY:

```
import numpy as np

import os, requests, base64, sys, tarfile, zipfile, pathlib, cv2

import six.moves.urllib as urllib

import tensorflow as tf

from collections import defaultdict

from io import StringIO

from matplotlib import pyplot as plt

from PIL import Image

from IPython.display import display

from object_detection.utils import ops as utils_ops

from object_detection.utils import label_map_util

from object_detection.utils import visualization_utils as vis_util


def load_model(model_name):

base_url = 'http://download.tensorflow.org/models/object_detection/'

model_file = model_name + '.tar.gz'

model_dir = tf.keras.utils.get_file(fname=model_name, origin=base_url + model_file,untar=True)

model_dir = pathlib.Path(model_dir)/"saved_model"

model = tf.saved_model.load(str(model_dir))

return model


def run_inference_for_single_image(model, image):

image = np.asarray(image)

input_tensor = tf.convert_to_tensor(image)

input_tensor = input_tensor[tf.newaxis,...]

model_fn = model.signatures['serving_default']

output_dict = model_fn(input_tensor)

num_detections = int(output_dict.pop('num_detections'))
```

```
output_dict = {key:value[0, :num_detections].numpy() for key,value in output_dict.items()}

output_dict['num_detections'] = num_detections

output_dict['detection_classes'] = output_dict['detection_classes'].astype(np.int64)

if 'detection_masks' in output_dict:

detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(output_dict['detection_masks'],
output_dict['detection_boxes'],image.shape[0], image.shape[1])

detection_masks_reframed = tf.cast(detection_masks_reframed > 0.5, tf.uint8)

output_dict['detection_masks_reframed'] = detection_masks_reframed.numpy()

return output_dict

def show_inference(model, frame):

image_np = np.array(frame)

output_dict = run_inference_for_single_image(model, image_np)

vis_util.visualize_boxes_and_labels_on_image_array(

image_np,

output_dict['detection_boxes'],

output_dict['detection_classes'],

output_dict['detection_scores'],

category_index,

instance_masks=output_dict.get('detection_masks_reframed', None),

use_normalized_coordinates=True,

line_thickness=5
)

return(image_np)

category_index=label_map_util.create_category_index_from_labelmap('mscoco_label_map.pbtxt',
use_display_name=True)

model_name = 'ssd_inception_v2_coco_2017_11_17'

detection_model = load_model(model_name)


video_capture = cv2.VideoCapture(0)

while True:
```

```
re,frame = video_capture.read()

Imagenp=show_inference(detection_model, frame)

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

if cv2.waitKey(1) and 0xFF == ord('q'):

break
video_capture.release()

cv2.destroyAllWindows()
```

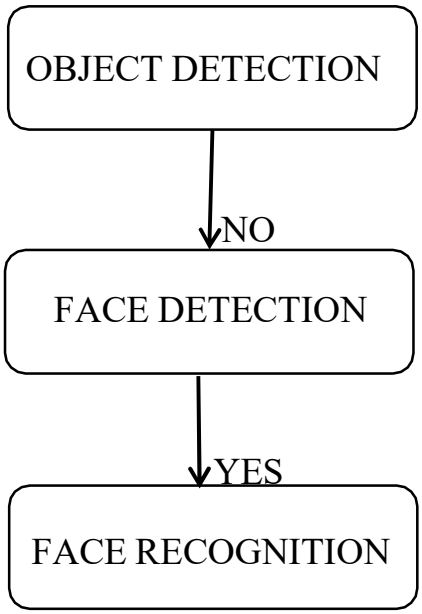
OUTPUT:




object detection
chair: 57%
backpack: 51%


object detection
chair: 55%
cat: 54%

WORKFLOW



Object detection

This is done using a process called object detection using tensorflow.Google's Tensorflow ObjectDetection API is an open source framework for object detection; it is based on Tensorflow and allows a user to define, train and utilize the models for object detection. The Tensorflow open source software library, developed by Google Brain Team, utilizes data flow graphs for numerical computations.

This process is achieved using the following steps:

- Data collection
- Image labeling
- Training data generation
- Label mapping
- Training configuration
- Inference graph
- Object detection

CONCLUSION

A pre-trained SSD object detection model was tested for detecting ID Cards, both as-is and as fine-tuned with a manually tagged data set. Model testing and training was done using features provided by TensorFlow Object Detection API, a freely available framework for object detection. Following conclusions were drawn based on the results:

1) The pre-trained model was very accurate in testing set.

2) A fine-tuned model worked reasonably well.

3) Problem areas were moving object and object of same size.

4) A model trained with data from one background was able to detect in another. The overall model performance did not much improve by training the model with different frame rate.

5) The model performance evaluated as mAP improved when the model was trained from 17 400 to 200 000 timesteps. Optimal number of timesteps is likely to be around 100 000 timesteps.

6) Face recognition using LBPH is a trenmendous success.