BLIND VISION

Vaithe Subramanian.S[1], Sanjeev.R[2], Dinesh.P.S[3]
[1,2]UG Student, [3]Assistant Professor,
ECE, Prince Shri Venkateshwara Padmavathy Engineering College, Tamil Nadu, India.

Abstract
This paper describes the implementation of TensorFlow object detection API and COCO (Common Objects and Context) dataset to detect the common objects around a person. The objects and living beings present in an image can be detected and identified by using MobileNets and SSD (Single Shot Detector) algorithms. The main advantage of using MobileNets and SSD is that, unlike other methods it can be used in laptops and other personal devices. The real-life objects and living beings captured using a normal webcam can be detected by interfacing python and OpenCV library. TensorFlow Deep learning is the base of the object detection so it is tapped into NVIDIA GPU of PC or laptop so that it enhances the speed and accuracy of detection. The SSD algorithm uses semantic segmentation to create the bounding boxes of varying colors based on the different classes present in an image. The objects are identified by comparing with pre-trained objects present in the COCO dataset. Along with the name of each object, their confidence score is also obtained which can be used to compare the accuracy of detection. To provide vision to a Blind person or any other device which lacks vision, the detected object's name is converted into speech using PYTTSX3 speech engine in python.

Index Terms: Tensorflow, COCO(Common Objects And Context), SSD(Single Shot Detector), Mobilenets, OpenCV, Object Detection, Python.

I. INTRODUCTION

The aim of this project is to detect and to identify the objects similar to that of a human .Blind Vision detects everything around the person, with the information of depth, collision detection, speech alerts. Blind Vision also features the face recognition, system face training, face recollection and recognition alerts .It guides the blind person move in environment like shopping malls, traffic roads, railway station, hospitals, etc. Tensors are nothing but the factor for representing the data in deep learning. Tensors are just multidimensional arrays, that allows you to represent data having higher dimensions.

In general, Deep Learning deals with high dimensional data sets where dimensions refer to different features present in the data set. In fact, the name "TensorFlow" has been derived from the operations which neural networks perform on tensors. It's literally a flow of tensors .The objects in the image are detected by using MobileNets + Single Shot Detectors(SSD) along with OpenCV. MobileNet SSD was trained to detect, then generate a set of bounding box colors for each class(car, dog,..). If we combine both the MobileNet architecture and the Single Shot Detector (SSD) framework, we arrive at a fast, efficient deep learning based method to object detection. Semantic segmentation attempts to partition the image into semantically meaningful parts, and to classify each part into one of the pre-determined classes. We can also achieve the same goal by classifying each pixel (rather than the entire image/segment).

A. TENSORFLOW

It uses "TensorFlow object detection API" for object detection and it can be built over TensorFlow framework. The dataset used here is COCO (Common Object Context) dataset. Tensors are nothing but the factor for representing the data in deep learning. Tensors are just multidimensional arrays, that allows you to represent data having higher dimensions. In general, Deep Learning you deal with high dimensional data sets where dimensions refer to different features present in the data set. In fact, the name "TensorFlow" has been derived from the operations which neural networks perform on tensors. It's literally a flow of tensors.

B. BASICS OF TENSORFLOW

TensorFlow is a library based on Python that provides different types of functionality for implementing Deep Learning Models. As discussed earlier, the term TensorFlow is made up of two terms – Tensor & Flow. Fig 1.1 shows the relation of the tensor in a 3-Dimensional format and the computational graph. In TensorFlow, the term tensor refers to the representation of data as multi-dimensional array whereas the term flow refers to the series of operations that one performs.

Basically, the overall process of writing a TensorFlow program involves two steps:

1. Building a Computational Graph
2. Running a Computational Graph.

C. BUILDING A COMPUTATIONAL GRAPH

A computational graph is a series of TensorFlow operations arranged as nodes in the graph. Each nodes take 0 or more tensors as input and produces a tensor as output. Let me give you an example of a simple computational graph which consists of three nodes – ***a***, ***b*** & ***c*** as shown below:

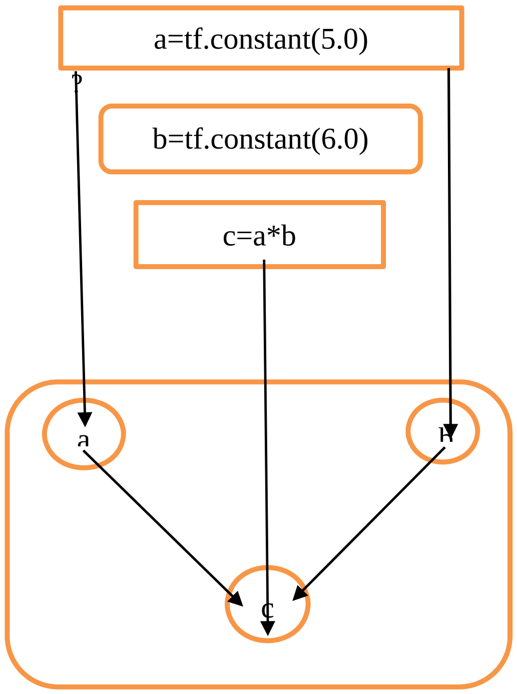

Fig 1:TensorFlow Code and Computational Graph

- Constant nodes are used to store constant values as it takes zero input, but produces the stored values as output. In the above example, a and b are constant nodes with values 5 and 6 respectively.
- The node c represents the operation of multiplying constant node a with b. Therefore, executing node c will result in multiplication of constant node a and b.
- Basically, one can think of a computational graph as an alternative way of conceptualizing mathematical calculations that takes place in a TensorFlow program. The operations assigned to different nodes of a Computational Graph can be performed in parallel, thus, providing a better performance in terms of computations.
- Here we just describe the computation, it doesn't compute anything, it does not hold any values, it just defines the operations specified in your code.

D. RUNNING A COMPUTATIONAL GRAPH

Example 1:

```
import tensorflow as tf
# Build a graph
a = tf.constant(5.0)
b = tf.constant(6.0)
c = a * b
```

- Now, in order to get the output of node c, we need to run the computational graph within a session. Session places the graph operations onto Devices, such as CPUs or GPUs, and provides methods to execute them.
- A session encapsulates the control and state of the TensorFlow runtime i.e. it stores the information about the order in which all the operations will be performed and passes the result of already computed operation to the next operation in the pipeline. Let me show you how to run the above computational graph within a session .

```
# Create the session object
sess = tf.Session()

#Run the graph within a session and store the output
 to a variable
output_c = sess.run(c)

#Print the output of node c
print(output_c)
```

```
10
11    #Close the session to free up some resources
      sess.close()
```

Output: 30

II. TENSORFLOW OBJECT DETECTION API

Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.

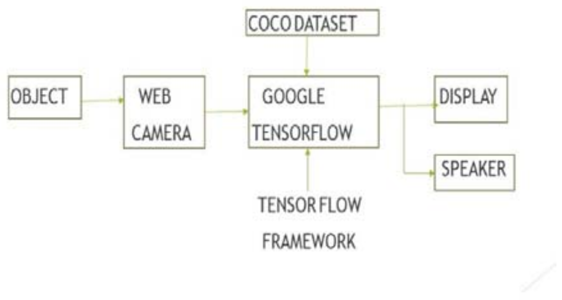

Fig 2: Block Diagram

A. SINGLE SHOT DETECTOR

Single Shot Detector (SSD) is an unified framework for object detection with a single network. You can use the code to train/evaluate a network for object detection task.By the combination of MobileNets and SSD algorithm the objects are sperated by diving into classes .SSD creates various bounding boxes with all shapes and sizes but, the objects with score greater than 0.5 are displayed.

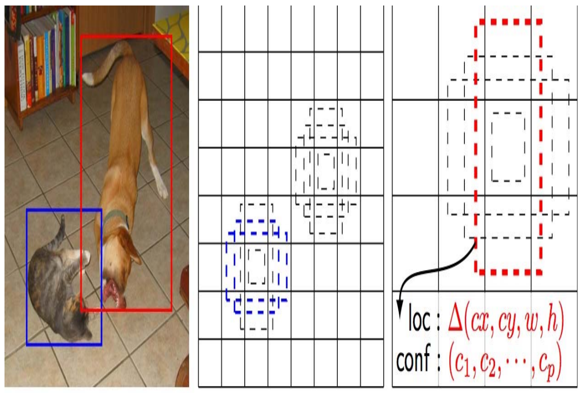
loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

Fig 3: Detection Using SSD

III. TOOLS USED

SOFTWARE REQUIREMENTS

- Windows 10 OS
- Tensor flow object detection API
- COCO Dataset
- Python
- Anaconda for python
- Spyder

HARDWARE REQUIREMENTS

- Web camera
- Laptop
- Nvidia graphic card
- Speaker

A. RESULTS AND DISCUSSIONS

The Detection of the Objects are first tested using a test image fig 4.After the compilation of codes we got fig 5. In this the animals are detected by providing bounding box with the name of the animal along with the confidence score in percentage.

Fig 4: Test Image

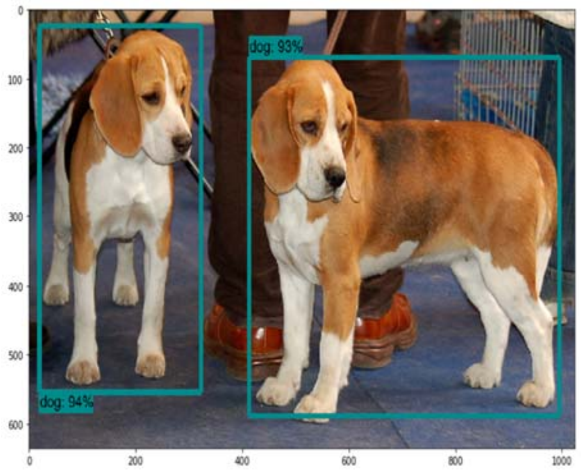

Fig 5: Detection of Objects in Test Image .

After the detection of test image, the live objects are tested using interfacing of webcam by using OpenCV library in python. fig 6 shows the detection of live common objects and person using webcam. Objects present in a frame overlapping each other are detected with varying colours and shapes of bounding boxes and confidence score for each object is obtained based on its visibility and accuracy of detection.

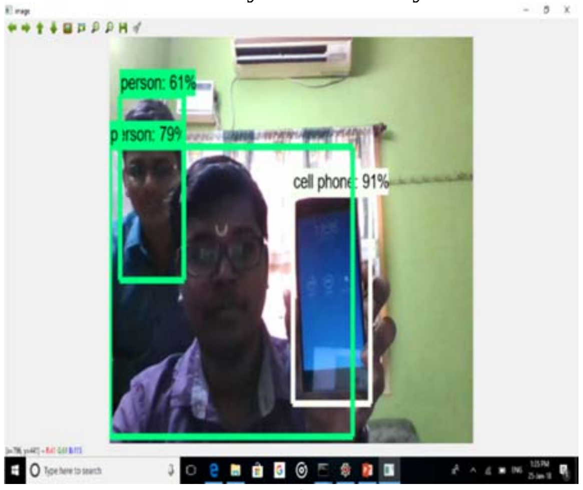

Fig 6: Detection of Live Objects Using Webcam.

Finally the text obtained is converted into speech by using PYTTSX3 python speech engine. This can be implemented in PC'S, Laptops and PDA'S and can guide blind person for the easy detection of common objects or an obstacle on their path.

IV. CONCLUSIONS

Thus the common objects are detected from the live Webcam with the help of TensorFlow API and OpenCV and the object is identified by its name and the text is converted to speech with the help of python speech engine. The future enhancement of our project can be used for facial recognition in order to take the attendance in the school to avoid mal-practices, to detect the movement of the eyes while driving in order to ensure that the driver is not sleeping to prevent the accidents by alerting the driver by vibrating or by using some sound alerts, it can be used to calculate the distance between two vehicles for parking the vehicle by speech alerts, and so on.

V. REFERENCES

[1] Evermann, J., Rehse, J.R., Fettke, P. (2016) 'A deep learning approach for predicting process behavior at runtime', PRAISE Workshop at the 14th International Conference on BPM.

[2] Evermann, J., Rehse, J.R., Fettke, P. (2017) 'Predicting process behavior using deep learning and Decision Support Systems', in Press.

[3] Hochreiter, S., Schmidhuber, J. (1997) 'Long short-term memory and Neural Computation', 9(8), 1735–1780.

[4] LeCun, Y., Bengio, Y., Hinton, G. (2015) 'Deep learning and Nature', 521, 436–444.

[5] Schmidhuber, J. (2015) 'Deep learning in neural networks: An overview', 61, 85–117.

[6] Tax, N., Verenich, I., Rosa, M.L., Dumas, M. (2017) 'Predictive business process monitoring with LSTM neural networks', Conference on Advanced Information Systems Engineering (CAiSE), Essen, Germany.

[7] Baldi .P, Sadowski .P, and Whiteson .D, (2014) 'Searching for Exotic Particles in High-Energy Physics with Deep Learning', Nature Commun., vol. 5, p.4308.

[8] Bergstra .J, Bastien .F, Breuleux .O, Lamblin .P, Pascanu .R, Delalleau .O, Desjardins .G, Warde Farley .D, Goodfellow .I, Bergeron .A, (2011) 'Theano: Deep learning on gpus with python', NIPS, BigLearning Workshop, Granada, Spain.

[9] Collobert .R, Weston .J, Bottou .L, Karlen .M, Kavukcuoglu .K, and Kuksa.P, (2011) 'Natural language processing', The Journal of Machine Learning Research, 12:2493–2537.

[10] Deepajothi .S and Selvarajan .S, (2012) 'A comparative study of classification techniques on adult data set', International Journal of Engineering Research and Technology, vol. 1.

[11] Deng .J, Dong .W, Socher .R, Li .L .J, Li .K, and Fei Fei .L, (2009) 'Imagenet: A large-scale hierarchical image database in Computer Vision and Pattern Recognition', IEEE, pages 248–255.

[12] Hall .M, Frank .E, Holmes .G, Pfahringer .B, Reutemann .P, and Witten .I .H, (2009) 'The weka data mining software: An update', SIGKDD Explor. Newsl., vol. 11, no. 1, pp. 10–18.

[13] Jia .Y, Shelhamer .E, Donahue .J, Karayev .S, Long .J, Girshick .R, Guadarrama .S, and Darrell .T, (2014) 'Caffe: Convolutional architecture for fast feature embedding', ACM International Conference on Multimedia, pages 675–678.

[14] LeCun .Y, Bottou .L, Bengio .Y, and Haffner.P, (1998) 'Gradient-based learning applied to document recognition', IEEE, 86(11):2278–2324.

[15] Szegedy .C, Liu .W, Jia .Y, Sermanet .P, Reed .S, Anguelov .D, Erhan .D, Vanhoucke .V, and Rabinovich .A, (2015) 'Going deeper with convolutions', Conference on Computer Vision and Pattern Recognition.