PROJECT REPORT

ON

"DARSHAN – Device for Avoidance of Remotely Sensed Hurdles and Navigation"

Submitted in partial fulfilment of the requirements for the partial completion of

MAJOR PROJECT [16EC8DCMPJ]

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

SUBMITTED BY:

Nikhil Anand B	**1BM16EC065**
Prajwal V Holla	**1BM16EC079**
Pranava YN	**1BM16EC005**
Ranjith D	**1BM16EC087**

Under the Guidance of

Dr. Sudhindra K R

(Associate Professor, ECE, BMSCE)

JAN - MAY 2020

Department of Electronics and Communication Engineering

B.M.S COLLEGE OF ENGINEERING

(Autonomous College Affiliated to Visvesvaraya Technological University, Belgaum)

Bull Temple Road, Basavanagudi, Bangalore-560019

DECLARATION

We undersigned students of final semester B.E in Electronics and Communication Engineering, BMS College of Engineering, Bangalore, hereby declare that the dissertation entitled “DARSHAN – Device for Avoidance of Remotely Sensed Hurdles And Navigation”, embodies the report of my project work carried out independently by us under the guidance of Dr. Sudhindra K R, Associate Professor, E&C Department, BMSCE, Bangalore in partial fulfilment for the award of Bachelor of Engineering in Electronics and Communication from Visvesvaraya Technological University, Belgaum during the academic year 2019-2020.

We also declare that to the best of our knowledge and belief, this project has not been submitted for the award of any other degree on earlier occasion by any student.

Place: Bangalore
Date: 27th June 2020

Nikhil Anand B	1BM16EC065
Prajwal V Holla	1BM16EC079
Pranava YN	1BM16EC005
Ranjith D	1BM16EC087

B.M.S COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that the project entitled **"DARSHAN – Device for Avoidance of Remotely Sensed Hurdles And Navigation"** is a bonafide work carried out by **Nikhil Anand B** (USN:1BM16EC065), **Prajwal V Holla** (USN:1BM16EC079) **Pranava Y N** (USN:1BM16EC005) and **Ranjith D** (USN:1BM16EC087) in partial fulfillment for the partial completion of **Major Project** [16EC8DCMPJ] during the academic year 2019-2020.

Dr. Sudhindra K R	**Dr. Arathi R Shankar**	**Dr. B. V. Ravishankar**
Associate Prof., ECE, BMSCE	HOD, ECE, BMSCE	Principal, BMSCE

External Examination: **Signature with date:**

1.

2.

ABSTRACT

Real time object detection and classification has been a real challenge due to the resource constraints at end devices. These applications require millions of arithmetic operations to be performed per frame of image to give effective output. Also cost of efficient devices that can do these operations were a bottleneck for its implementation. With advent of Raspberry pi, Nvidia jetson and other computational units now it has become feasible to implement these functionalities. These devices provide considerable computational performance together in an agreeable size factor of the board. Development of modern neural network architectures such as Fast RCNN, Darknet (Yolo) and Tensorflow etc have reduced computations required drastically also reducing the time of inference required for object detection.

In this project we utilize Yolo object detection architecture to implement a tool that can assist blind person to procure information about his surroundings. It utilizes Raspberry Pi as a base hardware, Raspberry Pi Camera to capture images. Text to Speech and Speech to Text engines are implemented to take in voice commands from the user and give outputs over a speaker (ear phones). We believe that this system can help blind person to explore his surrounding in a better way.

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We express profound gratitude to respected principal **Dr. B. V. Ravishankar ,** BMS College of Engineering for providing a congenial environment to work in. Our sincere gratitude to **Dr. Arathi R. Shankar,** Head of the Department, Electronics and Communication Engineering for encouraging and providing this opportunity to carry out the project in the department.

We would like to thank our guide **Dr. Sudhindra K R,** Associate Professor, Department of ECE who helped us in all the ways to carry out the project work. He stood beside and guided us in every step.

We thank all our professors for providing the basic knowledge without which this project wouldn't have been possible. Last but not the least we thank our family and friends, who made their valuable support compelled us to maintain a standard throughout our endeavour.

-

Nikhil Anand B

Prajwal V Holla

Pranava YN

Ranjith D

LIST OF FIGURES

LIST OF ABBREVIATION

1. CNN ....Convolutional Neural Network
2. YOLO ....You Only Look Once
3. ReLU ……………………………………………………………Rectified Linear Unit

CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction:

Neural Network:

A neural network is a series of algorithms that strives to recognize relationships in a set of data through a process which is very similar to how the human brain processes the data. Neural networks can adapt to changing inputs and hence there is very minimal if not any change in the design required to accommodate all kinds of data. The network is made up of multiple layers and interconnected nodes. Each node represents a data item from the input. A typical neural network consists of one input layer, one output layer and multiple hidden layers. Data is passed from one layer to the next, by combining it with a weight.

Here is an example of a very basic neural network –

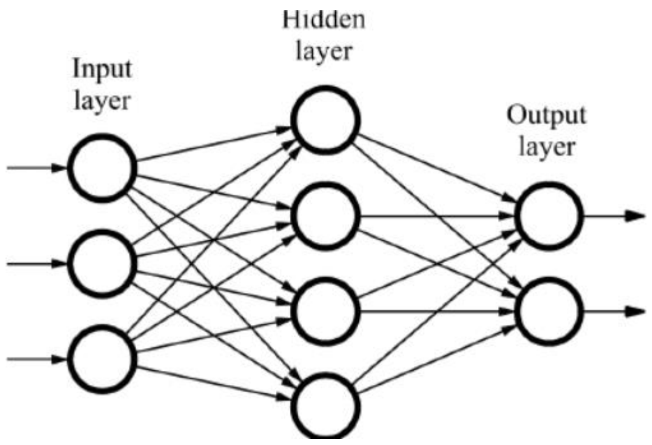


Fig 1. Simple Neural Network

Convolutional Neural Network

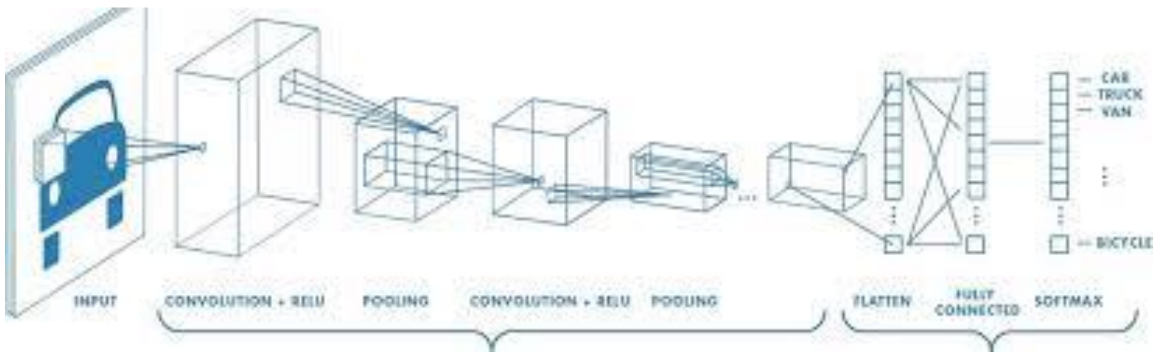


Fig 2. Convolutional Neural Network

Convolutional Neural Networks or CNNs, are a class of deep neural networks generally used for analyzing visual images. CNNs make use of fully connected nodes architecture. “Fully Connected nodes” implies that every node in one layer is connected to all nodes of the next layer. Due to this complex interconnection between the layers, CNNs are generally prone to over-fitting the data.

Considering the disadvantages, CNNs have lots of advantages that outweigh the faults. The most important advantage being that CNNs require relatively lesser pre-processing compared to other visual analysis algorithms.

The CNN in the basic sense is just a set of filters that the image is put through the final result is pointed out by the most filtered value. The ‘filters’ that are employed in the network are considered as ‘layers’. There are four major types of layers that are employed in the Convolutional Neural Network, and they are –

- Convolutional Layer
- ReLU Layer
- Pooling Layer
- Fully Connected Layer

Working of CNN

Let us now dive into the details regarding all the layers, by taking an example. Consider an image which has 9x9 pixels. The objective is to build a model which can classify an image as ‘X’ or ‘O’. Considering the straight-forward case, it will be the following way –

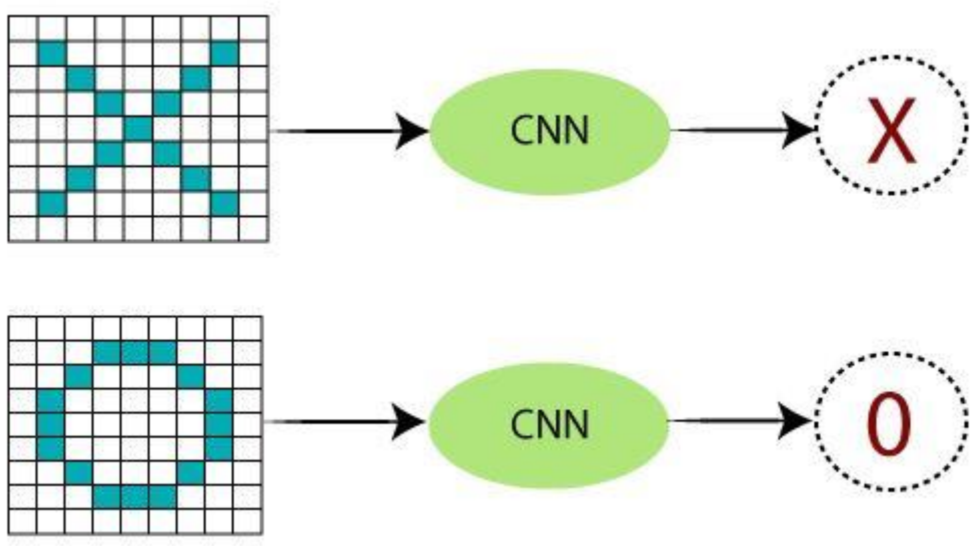


Fig 3. Working of CNN

This is a very straight forward as the ‘X’ and ‘O’ is centered in the image and the lines are very recognizable. But there are cases where the ‘X’ and ‘O’ in the image might come in different shapes and sizes and they may be placed in different parts of the image, as shown below –

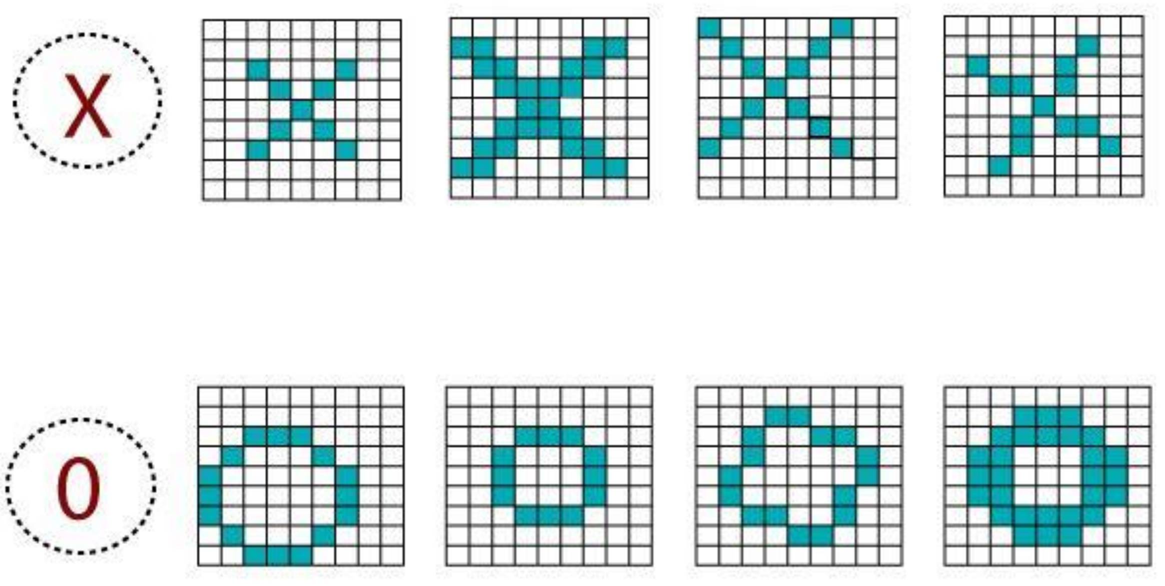


Fig. Output of CNN

These images can be easily classified by humans but a computer compares images and recognizes them based on the comparison between the pixel values. Although employing a computer to compare every pixel with the original one to classify the image to one of the classes, it is not feasible. This is proven by how the object in the image can change form. Hence it is more practical to consider using features from the original image to the test image to perform classification.

Convolution Layer

Consider the ‘X’ image. We can break it down to three major features –

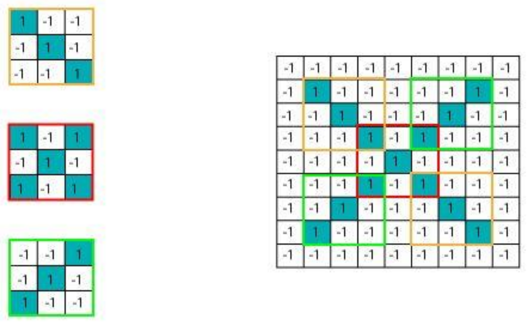


Fig 5. Convoultional Layer

Using the above three features, we can recreate the object 'X'. Now, we are going to move each feature over the image and in each location we will perform the following operation. The pixel value in the feature is multiplied with the pixel value in image. This output of all the pixels in the feature is added and divided by the number of pixels present in the feature to get a normalized value. This is basically what the convolution layer does. This layer slides the feature, or, to be technical, the "filter" over the entire image and get the convolved values.

ReLU Layer –

ReLU stands for Rectified Linear Unit. After the Convolution Layer is done with the image, we get the following –

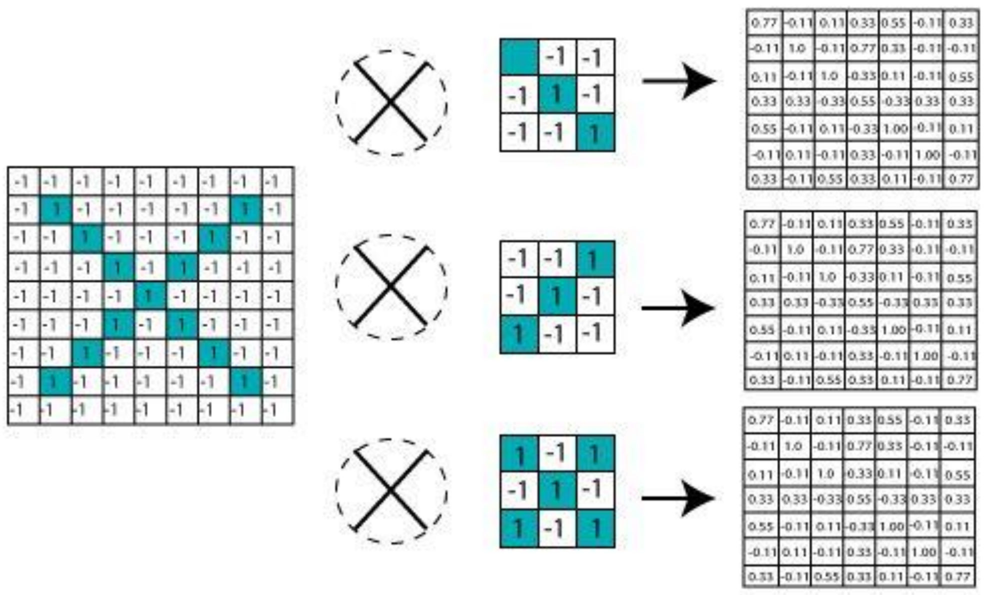


Fig 6. Output of Convolutional layer

To introduce non-linearity in the image, we use the ReLU layer which is again a filter function which replaces all negative values in the image with a '0'. To be technical, the ReLU layer activates a certain node, if the value on that node is above '0'. This introduces non-linearity within the image and provides for more differentiation in the image. The following image is the transformation function –

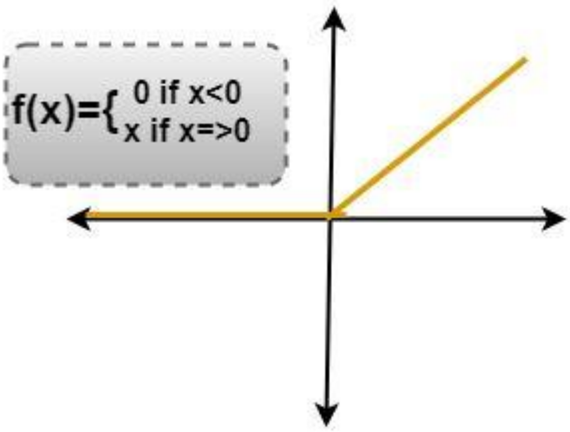
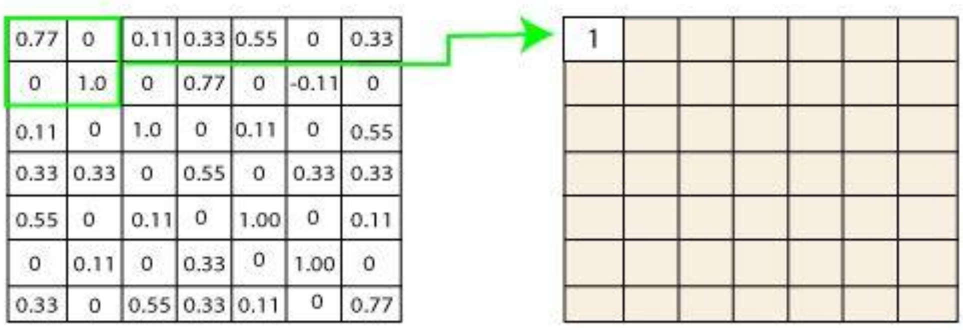


Fig 7. ReLU function

Max Pooling Layer –

Even with introducing non-linearity within the image, the amount of data to be processed is too high. To reduce this, we use the pooling layer. First, a window size is selected, and a stride value is selected. The pooling layer picks the maximum value within the window and then the window is walked over to the next part of the image and the process is repeated. The amount of walk is equal to the stride value.



0.77	0	0.11	0.33	0.55	0	0.33
0	1.0	0	0.77	0	-0.11	0
0.11	0	1.0	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Fig 8. Concept of Max Pooling

The outputs of the pooling layer of all features are as shown below –

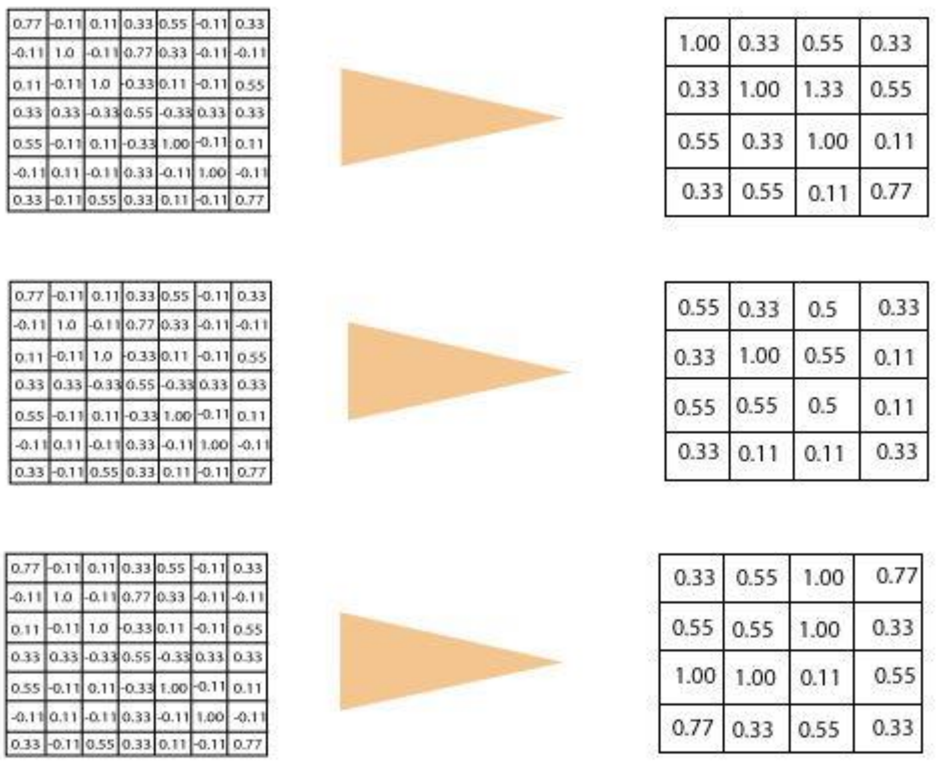


Fig 9. Max Pooling layer output

Considering the function of all the three layers, we went from a 9x9 image to a 4x4 image. We can further reduce this 4x4 image into a smaller size by stacking the layers one after the other. Therefore we finally end up with a neural network which is stacked with varied combination of the above three layers.

After all the filtering is done by some combination of the above layers, we come with a last layer called the fully connected layer where the nodes either point to one class, i.e, ‘X’ or to the other, i.e, ‘O’.

This is the entire working of the CNN. CNN offer lots of possibilities for customization – the stacking order of layers can be customized, the weights at the fully connected layer which decide the final classification, the window size of the pooling layers, the feature size and the number of each of the layers. All these factors allow for multiple possibilities.

YOLO – You Only Look Once

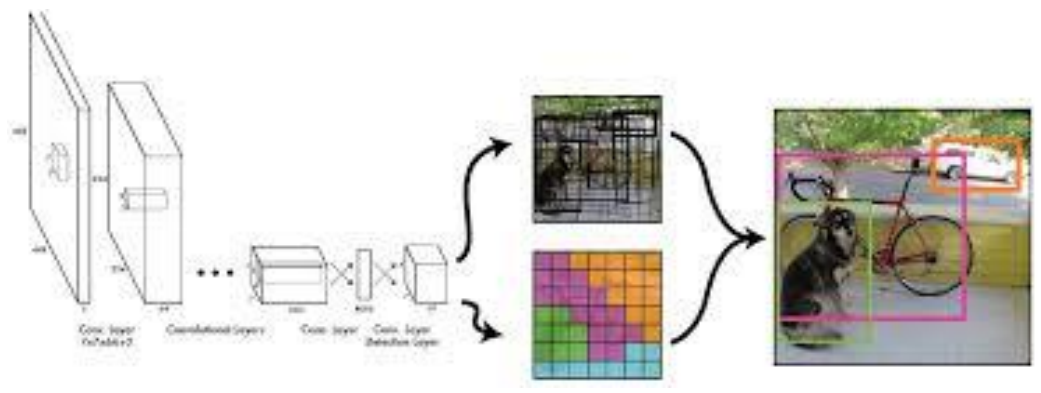


Fig 10. Yolo Architecture

The YOLO algorithm is one such example of a CNN. But it has been tuned and perfected to be the fastest object recognition algorithm compared to all other CNNs. The algorithm provides for analysis speeds as fast as 45 frames per second. This provides for a real time analysis on the environment of the visually challenged person.

Training the model:

Creating a custom dataset: Open Images dataset(OID) is a collaborative release of about ~9 million images annotated with image-level labels, object bounding boxes, object segmentation masks, and visual relationships. This dataset has been set up by google & it consists of over 600 classes of different objects. We have made use of the OIDv4 toolkit to procure the data. A Python script is used to convert the labels from the OIDv4 toolkit version to yolo version. Some of the classes that our apllication uses are chair, monitor, book, water bottle, mouse, etc.

400 labelled images are obtained for each class. 2 Csv files are used for the purpose of indexing the images. The Images with labelled with annotations(co-ordinates which indicate the bounding boxes). The OIDv4 toolkit Format makes use of the Top left & bottom right co-ordinates. These annotations are converted into yolo v3 normalized version. The labels(names like book, mouse, etc) are converted into a numbered value for the ease of computations.

Google colaboratory: Google colaboratory is used for the process of training the model. Colaboratory is a Google research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud. So it is possible to use this free cloud gpu without having to worry about the lack of gpu power on the local machines. The Darknet Framework is used to run the YOLO algorithmn. The

Darknet is an open source neural network framework written in C. The labelled custom dataset is moved into the Virtual Machine. Pre-trained weights are used to speed up the process of training and also to improve accuracy. After About 7 hours to training on Goggle Colab, the model is ready to be tested.

Problem analysis and discussion:

There exists a dire need for the visually impaired persons to gain a detailed insight of the various objects in their surroundings so that they can manoeuvre past them in a safe manner. It also imperative for them to gain the details of some objects around them with which they might be wanting to interact with such as chairs, windows, books, etc.

The project deals with an efficient and fast methodology that can identify the objects in the surroundings using the fast YOLO algorithm and also intimate the proximity of these objects from the user using proximity sensors. We aim to re-frame object detection as a single regression problem, straight from image pixels to bounding box co-ordinates and class probabilities. Using our system, You Only Look Once (YOLO), at an image to predict what objects are present and where they are.

CHAPTER 2: LITERATURE SURVEY

The paper [1] discusses a General Regression Neural Network (GRNN) which is a memory based neural network that converges to the underlying (linear or non-linear) regression surface. The GRNN is a one-pass learning algorithm with a parallel structure.

The paper [2] describes a set of concrete best practices for getting good results with neural networks. The paper claims that a Convoluted Neural Network is best suited for visual tasks where the network is not fully connected and is flexible and also guides us to build a simple "do-it-yourself" CNN.

The paper [3] proposes Complex-YOLO, a state-of-the-art real-time 3D object detection network on point clouds only. The paper also describes a network that expands YOLOv2, a fast 2D standard object detector for RGB images by a specific complex regression strategy to estimate multi-class 3D boxes in Cartesian space. For real-time object detection with high throughput and power efficiency, this paper [4] presents a Tera-OPS streaming hardware accelerator implementing a You Only Look Once (YOLO) CNN. The parameters of the YOLO CNN are retrained and quantized with the PASCAL VOC data set using binary weight and flexible low-bit activation.

The paper [5] claims to have trained a deep convolutional neural network to classify the 1.3 million high-resolution images in the LSVRC-2010 ImageNet training set. The paper also claims to have 60 million parameters and 500,000 nuerons, consisting of five convolutional layers.

Book [6] provides a very knowledgeable insight on various techniques and algorithms used for digital image processing.

Books [7] and [8] help in learning the fundamentals of neural networks and machine learning.

CHAPTER 3: METHODOLOGY AND IMPLEMENTATION

Hardware Architecture:

The project is implemented on a Raspberry Pi. Raspberry Pi is a single-board computer with an Arm cortex CPU. Several generations of Raspberry Pi have been released namely Pi Zero, Pi 2, Pi 3 and Pi 4.

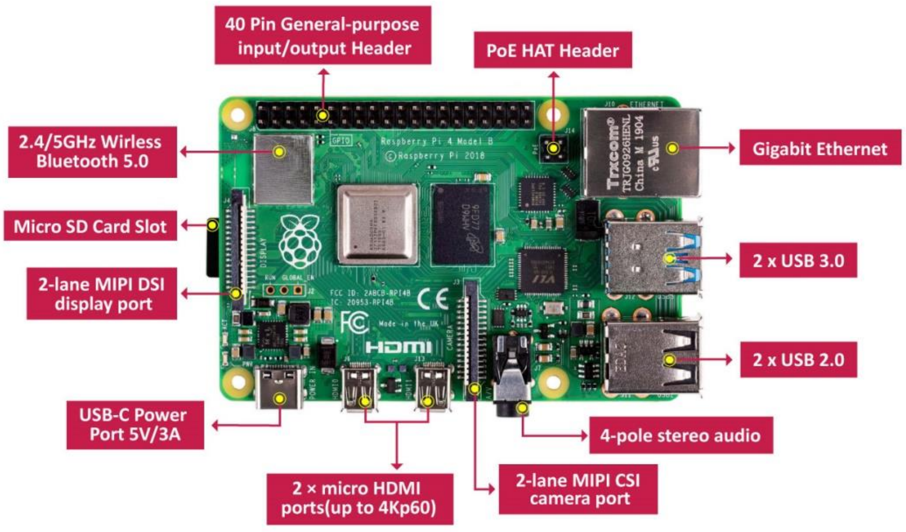

40 Pin General-purpose input/output Header
PoE HAT Header
2.4/5GHz Wirless Bluetooth 5.0
Gigabit Ethernet
Micro SD Card Slot
2 x USB 3.0
2-lane MIPI DSI display port
2 x USB 2.0
USB-C Power Port 5V/3A
4-pole stereo audio
2 × micro HDMI ports(up to 4Kp60)
2-lane MIPI CSI camera port

Fig 11. Raspberry Pi 4 Model B

Raspberry Pi 4 Model B chosen for this project is the latest release with following specs,

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB or 8GB LPDDR4-3200 SDRAM
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- 2 × micro HDMI ports
- Raspberry Pi standard 40 pin GPIO header
- 2-lane MIPI CSI camera port
- 5V DC via USB-C connector (minimum 3A)
- Operating temperature: 0 – 50 degrees C ambient

The generic hardware block diagram of the system is as shown below,

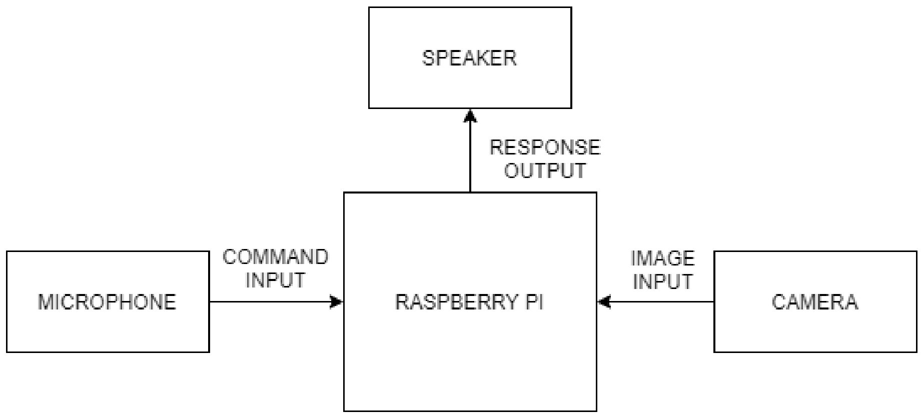


Fig 12. Hardware block diagram

Since Raspberry Pi doesn't contain an audio input jack, USB audio card is used to connect a microphone to the system.

Raspberry Pi runs Debian Operating System (known as Raspberry pi OS) which is a linux based OS with a good UI. It supports all Linux drivers and applications.

Raspberry Pi is powered via the USB Type-C port which takes in 5V DC and consumes upto 3A of current. A 10000mAh power bank is used to supply power and sustain the system for a long duration of operations.

Raspberry Pi camera famously known as PiCam is used. PiCam gives flexibility in the system in terms of hardware usage due to its small size and compatibility with the Raspberry Pi.

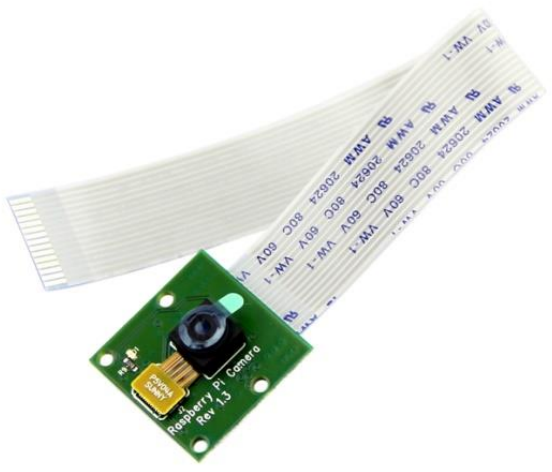


Fig 13. Raspberry Pi Camera

PiCam version 1.3 with the following specs are used,

- Still resolution 5 MegaPixels
- Video modes 1080p30, 720p60 and 640 × 480p60/90
- V4L2 driver support for Linux OS
- Sensor resolution of 2592 × 1944 pixels
- View angle of 65 degrees

For this particular system image is captured from the PiCam with 416 x 416 resolution.

Software architecture:

Programming language used:

Python 3.7

Python is an interpreted, general purpose, high level programming language. Python provides easy of usage for users in many ways. It has become one of the most famous languages in recent times and is widely used in scientific research, machine learning, object-oriented programming and scripting. There are 2 major releases of python, Python 2 and Python 3 with many sub-releases. Python is chosen as the language for the project because of its ease of use, wide support through availability of packages and libraries, level of abstraction and many more advantages. Python programs on a Raspberry Pi can also access the GPIOs and other peripherals available such as Camera, Display, USB ports etc.

Libraries used for the project are,

Opencv

Opencv is a popular library extensively used in image processing and computer vision applications. Mainly written in C and C++ languages, it offers wide range of API functions which are very well optimized and widely documented. In the project this library is used to import the Convolutional Neural Network trained on Darknet framework, to capture videostream from the camera, computing the output of the network for a given frame of video and to display those images. Opencv offers special build with optimised functions which runs faster on ARM architecture. Opencv provides most optimized functions for usage of Darknet framework.

speech_recognition

The commands input by the users are in form of audio. This audio is converted into text using speech_recognition library. This library uses Google's online speech to text engine. This produces most reliable results during the conversion.

pyttsx3

pyttsx3 library is used to convert text outputs into audio outputs that can be conveyed to the user over an earphone. It works offline and supports multiple TTS (Test To Speech) engines such as Sapi5, espeak, nsss etc. This library consists of functions that are very easy to use.

numpy

NumPy or Numerical Python package is a library which contains functions to handle multidimensional arrays and their manipulation. NumPy facilitates mathematical operations on large array within optimized time.

Flowchart of the Project:

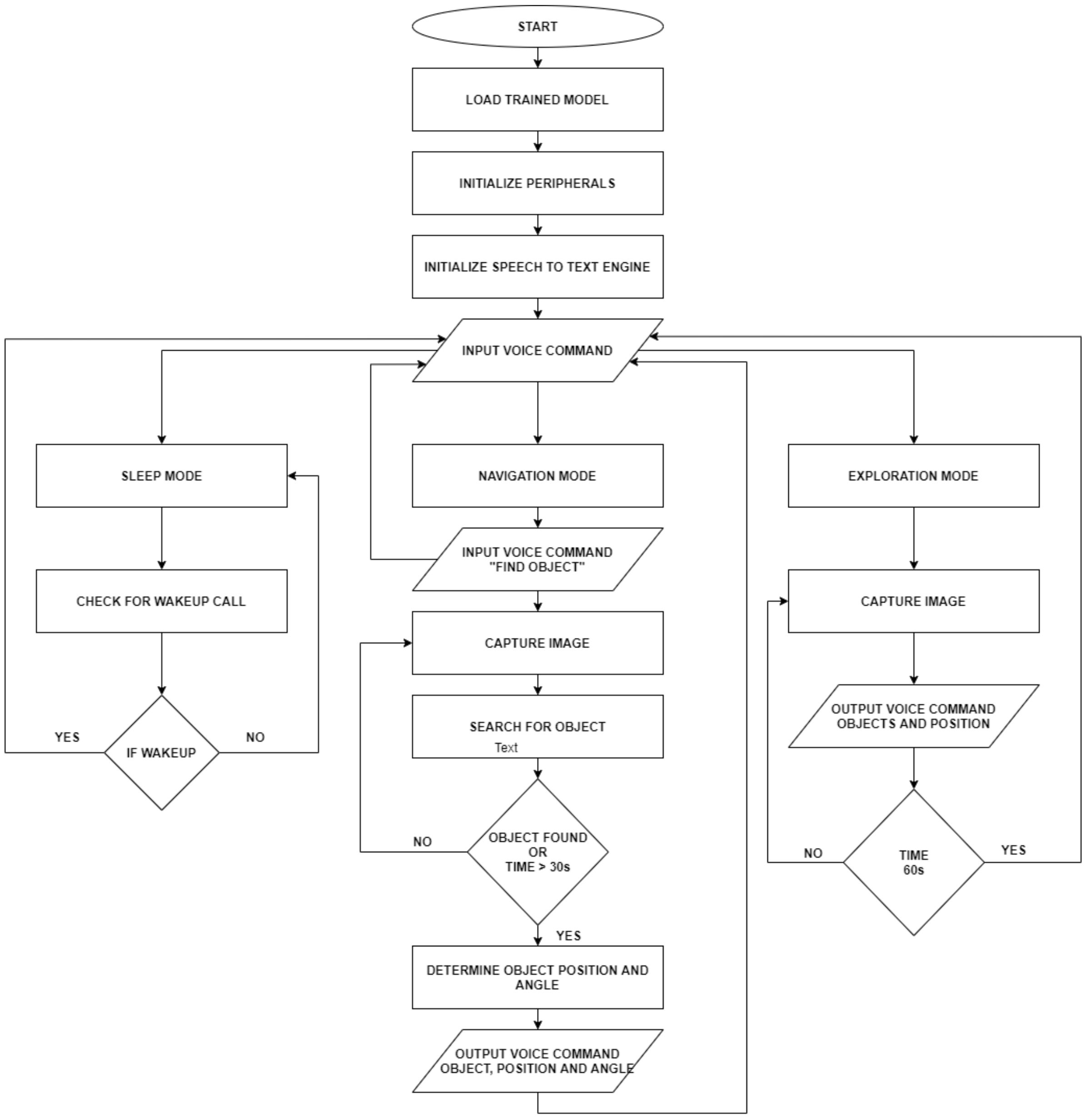

START
LOAD TRAINED MODEL
INITIALIZE PERIPHERALS
INITIALIZE SPEECH TO TEXT ENGINE
INPUT VOICE COMMAND
SLEEP MODE
CHECK FOR WAKEUP CALL
IF WAKEUP
YES
NO
NAVIGATION MODE
INPUT VOICE COMMAND "FIND OBJECT"
CAPTURE IMAGE
SEARCH FOR OBJECT
Text
OBJECT FOUND OR TIME > 30s
NO
YES
DETERMINE OBJECT POSITION AND ANGLE
OUTPUT VOICE COMMAND OBJECT, POSITION AND ANGLE
EXPLORATION MODE
CAPTURE IMAGE
OUTPUT VOICE COMMAND OBJECTS AND POSITION
TIME 60s
NO
YES

Fig 14. Software flow chart

Project Flow:

- Once the user starts the program, the model trained gets imported into the RAM.
- Other peripherals such as camera, speaker and microphones are initialized. Text To Speech (TTS) engine and speech to text engine are also initialized.
- After all initializations user hears a welcome note saying “Welcome to DARSHAN, speak ‘help me’ to activate” followed by a beep sound.

- At this point if user wants to use the system can give “help me” command through the microphone. Else the system will be in sleep mode.
- Upon reading the user’s command the system asks user to choose between the navigation mode, exploration mode and sleep mode.
- As per user command the mode chosen gets activated. After the execution of the command the system reiterates the same process by asking user for the mode to activate.
- Different modes of usages are explained below,

Sleep mode:

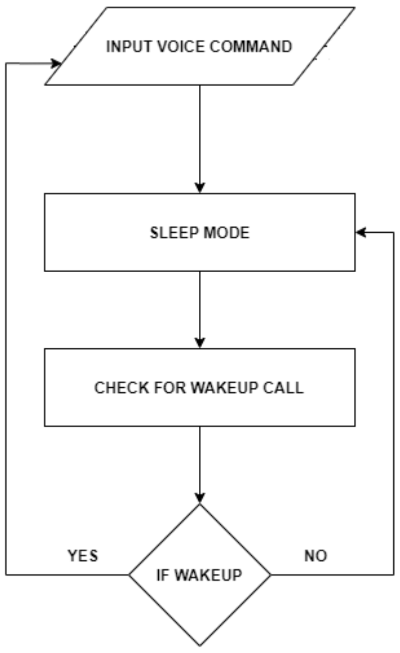


Fig 15. Sleep mode operation

- Sleep mode is the power saving mode that can be activated by the user when camera and other peripherals except microphone is turned off to save system power.
- Once this mode is activated the audio input command is monitored continuously by the system.
- If a user wants to reactivate his system, he should give “help me” command. This command activates the system and it asks user to choose one functional mode for execution.

Navigation mode:

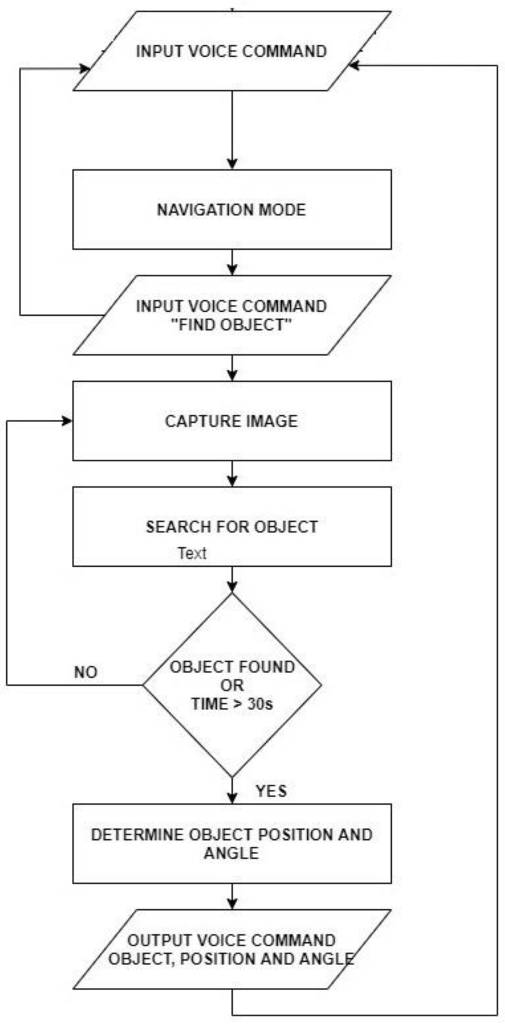

INPUT VOICE COMMAND
NAVIGATION MODE
INPUT VOICE COMMAND "FIND OBJECT"
CAPTURE IMAGE
SEARCH FOR OBJECT
Text
OBJECT FOUND OR TIME > 30s
NO
YES
DETERMINE OBJECT POSITION AND ANGLE
OUTPUT VOICE COMMAND OBJECT, POSITION AND ANGLE

Fig 16. Navigation mode operation

- Navigation mode helps the user to find a particular object he wants to access from the surrounding.
- Entering this mode user has to give command “Find Object” where Object is the name of the object he wishes to locate.
- After taking input from the user the system turns the camera capture on, captures a frame from the camera feed and passes it through the loaded network.
- The output obtained by the network is analysed to locate the “Object” required by the user.
- If the object is present in the frame the system gives out voice output specifying the object asked, its position (left or right) and the estimated angle of object from centre of the frame.
- The following image specifies a frame analysed and a bounding box located by the network.

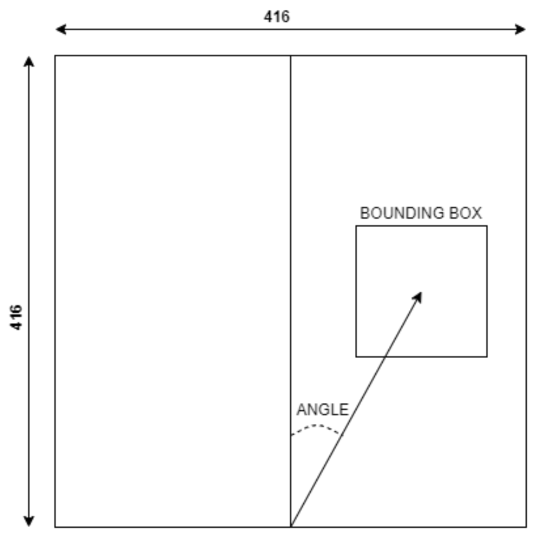


Fig 17. Single frame in navigation mode

- The angle of bounding box is estimated from the following formula,
- deg = math.degrees(math.atan(abs(centerX-user_x)/abs(centerY-user_y)))

where centerX and centerY are the co-ordinates of the centre of the bounding box obtained. user_x and user_y are the co-ordinates of bottom centre of the image which specify user position

- The system stays in this mode for 30s, if the object is not found in the time the system exits the mode and gives out the output “couldn’t find object”
- This mode is very handy if the user wants to locate a particular object from his surroundings.

Exploration mode:

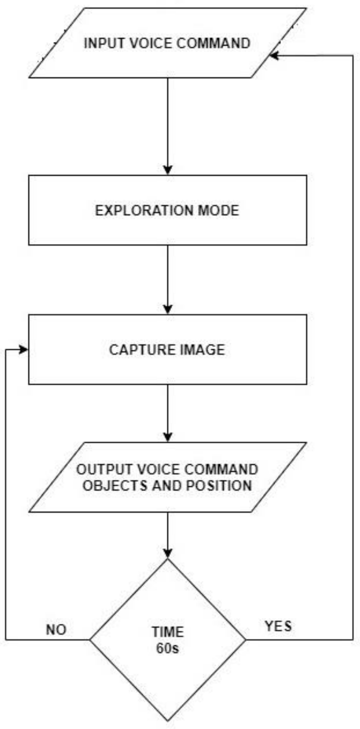


Fig 18. Exploration mode operation

- Exploration mode helps user understand his surrounding by outputting the name and position of objects around him.
- Once the user enters this mode, camera feed is turned on and continuous frames are received and every 60th frame is analysed by the network.
- Analysing every 60th frame provides sufficient delay between the outputs making user feel comfortable.
- Below image shows a frame and its output,

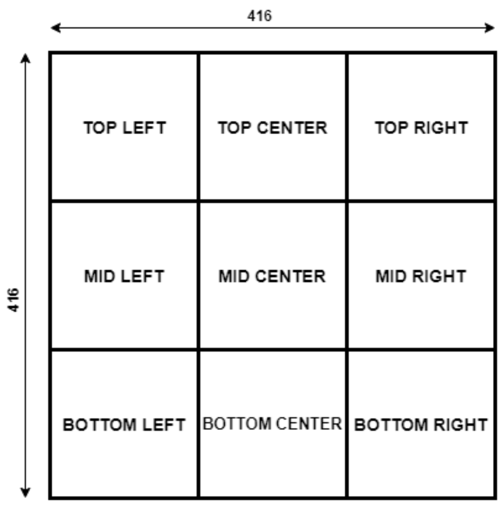


Fig 19. Single frame in exploration mode.

- The entire frame is classified into top, mid and bottom rows and left right and centre columns.
- Combination of row and column positions of the object gives a crude estimation of the object location.
- This mode is helpful for the user if he is in some unfamiliar location and wants to get a fair picture of his environment.
- This mode once activated runs for 60s and then returns back to sleep mode.

Image processing and implementation:

In our project we used the opencv deep neural network module to run the YOLO neural network we created. We choose the opencv deep neural network as it had supported YOLO and Darknet. The opencv deep neural network functions used and their brief description are as follows:

1. cv2.dnn.blobFromImage
2. cv2.dnn.readNetFromDarknet
3. net.setInput
4. net.forward

cv2.dnn.blobFromImage:

In order to obtain a higher accuracy in predictions from deep neural network the input image must be preprocessed. In our context of deep learning and image classification the preprocesses include mainly two processes:

1 Mean Subtraction

2 Scaling by some factor

The opencv function cv2.dnn.blobFromImage is used to do both these operations on the input image. The function also provides optional swapping of channels where this has been done in the program.

Mean subtraction is used to combat illumination changes in the images and thus aid the convolutional neural network we created. The first step involves the computation of average pixel intensity over the entire image. Now we end up with the average RGB values as μ_{R}, μ_{G} and μ_{B} .

In some cases, the mean RGB values may be computed channel-wise rather than pixel-wise which results in an MxN matrix. In this case the MxN matrix for each channel is then subtracted from the input image.

We here have used the pixel-wise version and have computed the mean RGB values in the program. When the image is to passed on to the neural network we subtract the mean from each channel to obtain new RGB values as follows:

$$R_{new} = R - \mu_R$$

$$G_{new} = G - \mu_G$$

$$B_{new} = B - \mu_B$$

We can also have a scaling factor β which adds in normalization:

$$R_{new} = (R - \mu_R)/ \beta$$

$$G_{new} = (G - \mu_G)/ \beta$$

$$B_{new} = (B - \mu_B)/ \beta$$

The value β can be the standard deviation across the training set or can be set to a prefixed value manually which is done in our case to (1/255).

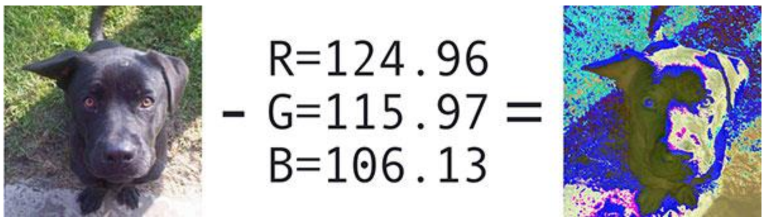


Fig 20. Subtracting mean from image

This is a sample image where mean subtraction is done. The left image is the original image, the center image shows the mean RGB values calculated and the right image is the output image after the mean subtraction process.

The format of this function is as follows:

blob = cv2.dnn.blobFromImage(image, scalefactor=1.0, size, swapRB=True)

image: The input image that has to be preprocessed

scalefactor: As discussed above the factor β (which in our case is 1/255)

size: The spatial size that the CNN expects.

swapRB: The opencv assumes the images are in BGR order but the mean can be in RGB order. In order to resolve the discrepancy, we have the option of swapping R and B channels.

cv2.dnn.readNetFromDarknet:

The function cv2.dnn.readNetFromDarknet is a function used to read the neural network model stored in Darknet model files. The function format is as follows:

net = cv2.dnn.readNetFromDarknet(config_file, darknetModel)

The config_file is a text description of the neural network architecture and the darknetModel is the model that we have trained in Darknet platform.

The function returns a Network object that is ready to launch the input through the neural network.

The Darknet platform is written in C and is used for creating the neural networks by training and as explained earlier we have used the Darknet model for its extensive support for YOLO.

net.setInput:

The function net.setInput is used to set the input value to the neural network. The format of the function is as follows:

net.setInput(blob)

The function is of void type and returns nothing while the net is of type Network object. Here the input parameter is the blob generated from the function *cv2.dnn.blobFromImage*. The blob is very similar to the image but with four dimensions. An image has two dimensions that is width and height whereas, the blob has four dimensions namely num_images (number of images), num_of_channels (three in our case) and the two dimensions from image (width and height).

net.forward:

The net.forward function is used to traverse the neural network and do the computations for each layer of the network. The format of the function is as follows:

layerOutputs = net.forward(outputName)

The parameter passed into the function is the name of the output layer for which the output is needed for the user and object retuned is a blob object of that output layer name specified (outputName).

From here we for loop the layerOutputs in order to extract objects and their confidence values.

CHAPTER 4: RESULTS AND DISCUSSION

The result from the DARSHAN is of audio for the visually impaired people to process but we have captured frames from the processed video stream and are highlighted here. DARSHAN is capable of detecting the following objects:

person	wine glass	elephant	Dining table
bicycle	cup	bear	toilet
car	fork	zebra	Tv monitor
motorbike	knife	giraffe	laptop
aeroplane	spoon	backpack	mouse
bus	bowl	umbrella	remote
train	banana	handbag	keyboard
truck	apple	tie	cell phone
boat	sandwich	suitcase	microwave
traffic light	orange	Frisbee	oven
fire hydrant	broccoli	skis	toaster
stop sign	carrot	snowboard	sink
parking meter	hot dog	sports ball	refrigerator
bench	pizza	kite	book
bird	donut	baseball bat	clock
cat	cake	baseball glove	vase
dog	chair	skateboard	scissors
horse	sofa	surfboard	teddy bear
sheep	potted plant	tennis racket	hair drier
cow	bed	bottle	toothbrush

Some of the images captured from the processed video stream are as follows:

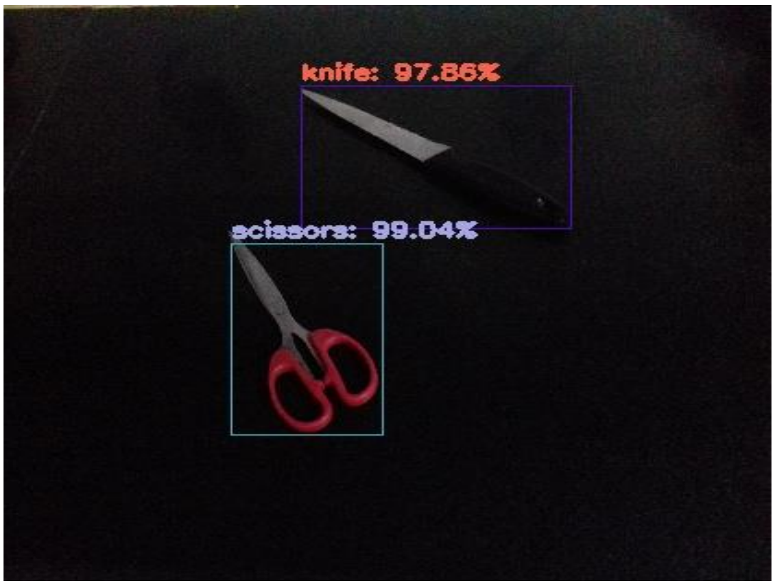


Fig 21. Output 1

As seen in the above image the scissors and knife are detected at a high rate of 99.04% and 97.86% respectively.

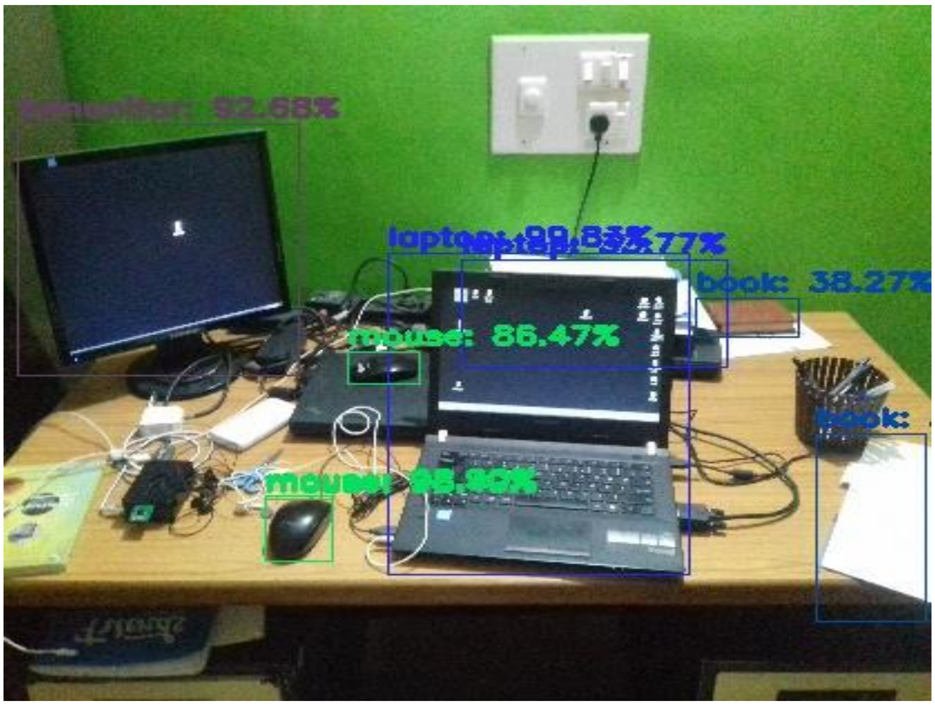


Fig 22. Output 2

In this frame from a processed video stream two mouse, laptop, tvmonitor and two books are detected (some with high confidences and some objects in the background with lower confidences).

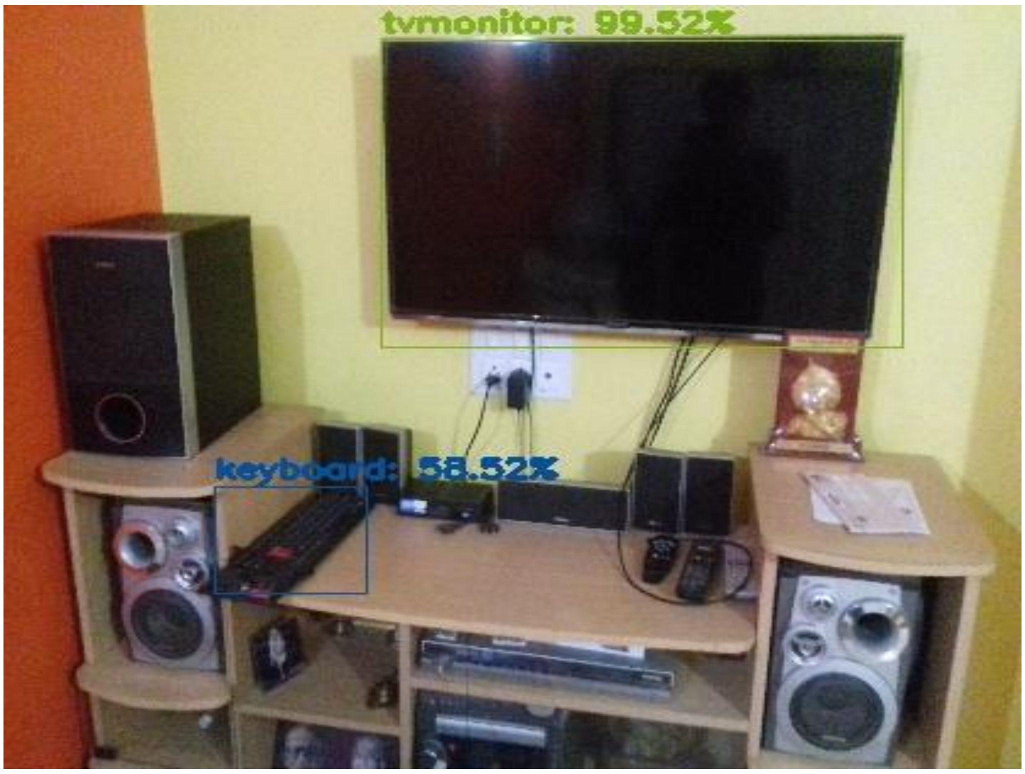


Fig 23. Output 3

In this frame of a processed video stream the LED TV and keyboard are detected with 99.52% and 58.52% respectively.

CHAPTER 5: FUTURE TRENDS

Some of the future trends of DARSHAN project can be as follows:

1 **Increasing speed and accuracy**

DARSHAN for now runs at an acceptable speed of 3seconds per frame but this has to be improved by a far stretch to make it fully real time and enabling the visually impaired people to navigate even through the traffics and other potentially dangerous area.

2 **Integrating sensors to increase functionality**

We can integrate proximity sensors to the DARSHAN to enable visually impaired people to not only detect objects but also detect the distance to the detected object. This is just an example on how we can integrate the sensors into DARSHAN.

3 **Android Application Development**

A future trend of project DARSHAN is to implement DARSHAN on an android/iOS where the project can be much simpler in terms of hardware as a phone already has a camera and required computation power to process video streams. It can be designed in a way that doesn’t irritate users with bulky equipment.

4 **Developing faster models for specific scenarios**

As stated above DARSHAN operates at an acceptable speed of 3seconds per frame but in scenarios which require quick recognition (like spotting animals like snake and predicting the animal’s behavior also for traffic incidents to help navigation for visually impaired people).

REFERENCES

[1] Specht, Donald F. *"A general regression neural network."* IEEE transactions on neural networks 2, no. 6 (1991): 568-576.

[2] Simard, Patrice Y., David Steinkraus, and John C. Platt. *"Best practices for convolutional neural networks applied to visual document analysis."* In Icdar, vol. 3, no. 2003. 2003.

[3] Duy Thanh Nguyen, Tuan Nghia Nguyen, Hyun Kim and Hyuk-Jae Lee *"IEEE TRANSACTION ON VERY LARGE-SCALE INTEGRATION (VLSI) SYSTEMS-A High Throughput and PowerEfficient FPGA Implementation of YOLO CNN of Object Detection."*

[4] Martin Simony, Stefan Milzy, Karl Amendy, Horst-Micheal Gross *"Complex-YOLO:An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds"*

[5] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton *"ImageNet Classification with Deep Convolutional Neural Networks"*

[6] Gonzales, Rafael C., and Richard E. Woods. *"Digital image processing"* (2002).

[7] Haykin, Simon. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[8] Kumar, Satish. *Neural networks: a classroom approach*. Tata McGraw-Hill Education, 2004