VISION-Z: AI BASED AIDING DEVICE FOR VISUALLY IMPAIRED AND BLIND PEOPLE

SIMRA KAUSAR RAJA

WAJIHA JAWWAD

M. ZARGHAM BAIG

PERVAZ ALAM

Supervisor

Brig. Dr. Abdul Ghafoor

Submitted to the faculty of Department of Electrical Engineering,
Military College of Signals, National University of Sciences and Technology,
in partial fulfillment for the requirements of B.E Degree in Electrical Engineering

June 2021

By

SIMRA KAUSAR RAJA

WAJIHA JAWWAD

M. ZARGHAM BAIG

PERVAZ ALAM

Supervisor

Brig. Dr. Abdul Ghafoor

Submitted to the faculty of Department of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology,

in partial fulfillment for the requirements of B.E Degree in Electrical Engineering

June 2021

CERTIFICATE OF CORRECTIONS & APPROVAL

Certified that work contained in this thesis titled "AI Based Aiding Device For Visually Impaired And Blind People" carried out by Simra Kausar Raja, Wajiha Jawwad, M. Zargham Baig, Pervaz Alam under the supervision of Brig Dr. Abdul Ghafoor for partial fulfillment of Degree of Bachelors of Electrical Engineering, in Military College of Signals, National University of Sciences and Technology, Islamabad during the academic year 2020-2021 is correct and approved. The material that has been used from other sources it has been properly acknowledged / referred.

Approved by

Project Supervisor

[illegible]

Brig. Dr. Abdul Ghafoor

Date: 15TH JUNE 2021

DECLARATION

No portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

[illegible]

Signature of Supervisor

Acknowledgements

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You setup in my mind to improve it. Indeed, I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout in every department of my life.

I would also like to express special thanks to my supervisor Dr. Abdul Ghafoor and co-supervisor Dr. Mohsin Riaz (Comsats) for his help throughout the thesis.

I would also like to pay special thanks to Naveed Mazhar for his tremendous support and cooperation. Each time I got stuck in something, he came up with the solution. Without his help I wouldn’t have been able to complete my thesis. I appreciate his patience and guidance throughout the whole thesis.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Dedicated to my exceptional parents and adored siblings whose tremendous support and cooperation led me to this wonderful accomplishment.

Abstract

Traditionally, a visually impaired person uses a white cane as a tool for directing them while moving or walking. Although, this cane is useful, it cannot guarantee high level of protection of blind person away from all obstacles and help them in self-navigating in complex environments. Mostly such handheld devices aim at utilizing the sensory ability of the person.

Smart Assistive devices are the current technological development in finding easier navigation solutions for visually impaired people, as these devices provide solutions that work in both indoor and outdoor environments leading to a much-improved life quality.

This research focuses on a wearable assistive device for the blind. This project introduces a smart assistive device that provides complete autonomy to the visually impaired people by providing detection and recognition of objects and people around them. It also monitors the health of the person. This multipurpose system is designed to help the blind person to navigate alone safely and to avoid any obstacles that may be encountered, whether fixed or mobile, to prevent any possible accident. The device provides audio output for every functionality. The system is designed to be convenient and accessible, so that visually impaired person can operate it easily.

By this project, we aim to provide a unique, cost effective and completely upgraded system for visually impaired people.

Key Words: *Visually impaired people, Smart Assistive devices, Navigation*

Table of Contents

List of Figures

List of Tables

CHAPTER: 1
INTRODUCTION

This chapter provides a comprehensive introduction of the project "VISION Z – AI Based Aiding Device for Visually Impaired People".

1.1 Overview:

Right now, visually impaired individuals face many difficulties and a common one is when they involve in self-navigation in environments that are new or complex for them. They do not have the ability to self-navigate without making contact with the surrounding using an aiding object or listen to the surrounding environment. This is the main focus of this project which addresses this issue by providing a better portable and wearable device for them to become autonomous in their daily living lives.

1.2 Background and Motivation

Out of the world's 7.79 billion population, an estimated 49.1 million are blind ,221.4 million people have moderate visual impairment, and 33.6 million people have severe visual impairments till date. The estimated number of blind persons has increased from 34.4 million in 1990 to 49.1 million in 2020 which is a 42.9% increase.

Analysis of data from all these countries over the world says that there are more than 200 million people with moderate to severe vision impairment out of which approximately 49 million are completely blind. Also, according to some research, the number of blind people across the world is set to triple within the next four decades the worst affected areas for visual impairment are in South and East Asia. In Pakistan only from the last studies, we know that 1.12 million people are blind, 1.09 million people had severe vision loss and 6.79 million people had moderate vision loss. Looking at these global and local figures the burden of vision loss is becoming hard to tackle.

When we look at existing solutions for such people on the market some of them are outdated, some are too expensive keeping in mind the low and middle-income people and most of them offer singular and limited functionality.

Such impairment can notably affect a person's life. Visually impaired people should be able to communicate with the world around them seamlessly and effortlessly, with a smart assistive device with handheld control that brings the surroundings to them in an audible form.

1.3 Scope and Deliverables

We propose an AI-based recognition and detection system incorporated with a group of sensors that provides multiple functionalities. This device is wearable and portable which makes it easy for the person to perform daily routine tasks independently. This device is a mid-range solution, which is not too expensive and is kept as economical as possible without compromising its functionality too much making it affordable. It includes using deep learning algorithms and libraries used in python language for recognition and detection. Its processing unit is the Nvidia Jetson Nano developer kit used in concurrence with Arduino Uno board, USB camera, USB headphones mounted along with a temperature and pulse rate sensor. Its functionalities include obstacle detection via a multitude of ultrasonic sensors in the path view of the user. Known people in front of the user are recognized through a face recognition scheme and anyone elsewise as unknown. Identification of Pakistani paper currency notes is done using a currency recognition scheme. The device will use an image classifying model for image captioning of the surrounding. The thus developed codes will be integrated into the processor and tested via a live USB camera feed. All output, in the end, is provided to the user as audio through USB headphones based on the functionality the user selects through the keyboard/keypad at that time.

In conclusion, the goal of this project is to act as a secondary vision in audible form for the visually impaired person. This is achieved by incorporating both our software and hardware skills and implementing them practically in a system.

CHAPTER: 2
LITERATURE REVIEW

Various systems are developed to reduce the problems faced by impaired people. One of the models is developed in [1].

In this model obstacle is detected by implementing multi-sonar system. In 2000, a tiny mobility design is used for impaired people. This design was based with wheels [2].

In 2007 designed a microprocessor and a PDA called Design of a Wearable Walking Guide System for the Blind. This design does not have a very vast scope because of its very limited functionality [3].

We provided a concept for a mobile navigation aid that uses Microsoft Kinect and optical marker tracking to help visually impaired people navigate the building. This is the result of a student project and is totally based on lower-cost hardware and software. It gives none stop tactile information about a person's waist, gives an impression of the person's surroundings and warns of objects. [4].

Blind people use white sticks as a tool to guide them while moving. also, the white cane is useful, it is not fully guaranteed to protect the blinds from various objects. This article presents an obstacle avoidance method that uses an electronic cane as a walking tool for obstacle avoidance practitioners. It uses infrared sensors to detect obstacles on the way. With the help of various obstacles [5].

Introduce YOLO, which is a new method. Previous work will reuse classification technique to perform detection. Whereas in yolo we frame the object detection into a regression problem of by using grid boxes and compare class probabilities [6].

This article presents the FPGA which is a real-time vision system that simulates this method. This is cheaper mobile system consisting of a CMOS camera. The deployment technic used in this article are suitable for other smart mobile sensors and machine vision applications such as the power, speed and the latency are important factors [7].

Batavia et all discussed a technique that is the combination of two techniques: adaptive color segmentation and stereo-based color homograph. That algorithm is specifically suitable for such surroundings where the terrain is relatively flat, and the colors are same [8].

In 2014," automatic obstacle detection using image segmentation" by Pankaj Jain and Dr Mohan Awasthy implemented the method by dividing it into two parts: segmenting the obstacles containing images, and then finding the obstacles from those obstacles containing images [9].

1.4 Use of Background study

These documents are about obstacle detection and object recognition. Each document focuses on a particular aspect of the project, discussing the positives and negatives, and then suggesting the most appropriate method or algorithm. These background studies not only highlight many of the problems encountered by developers in the past, but also suggest what improvements can be made in terms of functionality and cost effectiveness in the future.

CHAPTER: 3
DESIGN AND SPECIFICATIONS

3.1 Project Description and Salient Features

Vision Z is an amalgamate of AI technology which has inculcated a multitude of features with the goal to provide assistance to blind and visually impaired people. This would ultimately help them in self-navigation of their surroundings. It is composed of six main modules, which are:

- ✓ Facial Recognition
- ✓ Image captioning
- ✓ Currency Detection
- ✓ Object Detection
- ✓ Health Monitoring
- ✓ Audio Output

The edge device which we are making use of is the **Nvidia Jetson Nano A1 Development Kit.** It's a videlicet GPU-powered compact computer upon which we have deployed our whole project. Our device is controlled by the user itself who decides which functionality he wants to utilize at the moment.

One can use the facial recognition feature which is used to detect and identify people. It can also be used to describe the current surroundings of the user using the image captioning feature. Output results are given through audio which proves to be very beneficial for a user who is visually impaired or blind. It even has currency detection which will identify the current Pakistani currency present with the user. Another useful capability offered by Vision-Z is that it detects different kinds of obstacles which are in the path of the blind or the visually impaired person with the help of ultrasonic sensors. Health monitoring for the user is also available which gives the pulse rate and the temperature of the user using the pulse sensor and the temperature sensor.

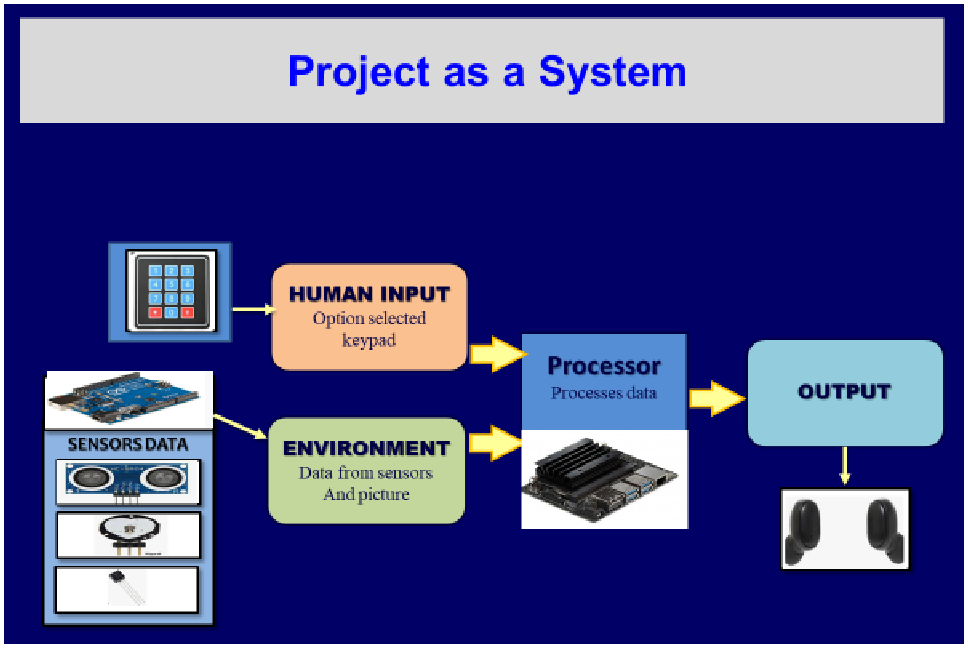


Figure 1 Project as a System

The diagram below is depicting a schematic form of the general arrangement of the different peripherals and sensors of our project.

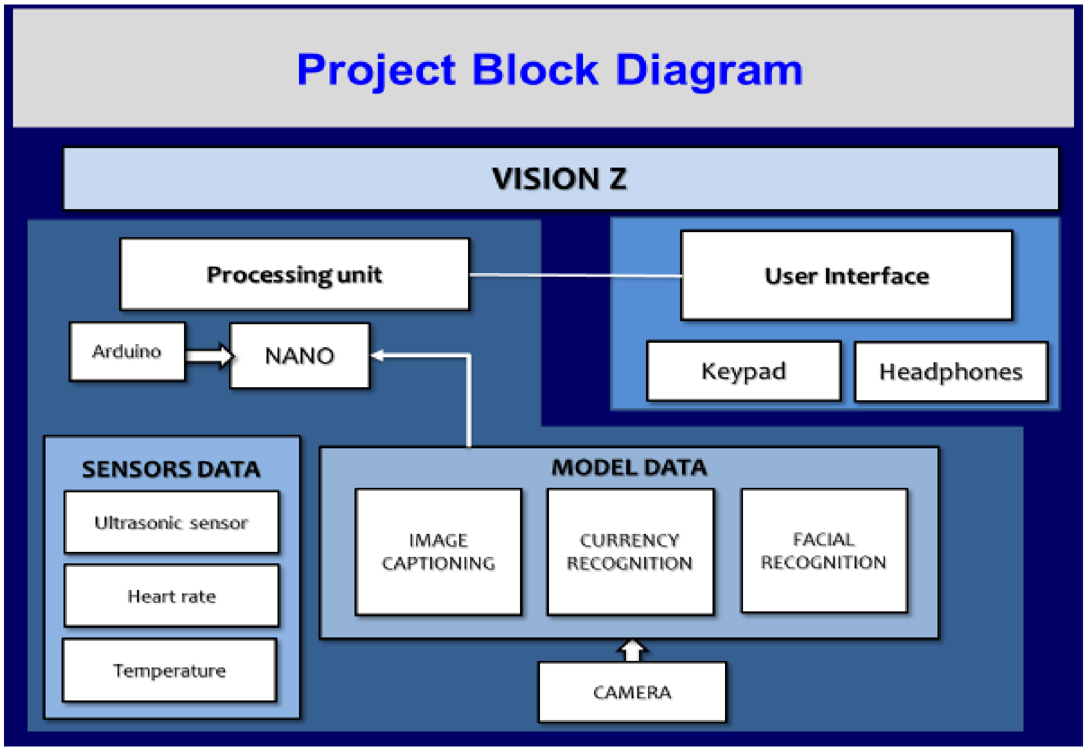


Figure 2 Project Block Diagram

3.2 Setting up the NVIDIA Jetson Nano Developer Kit

Jetson Nano is an A1-based developer kit that is a compact yet sturdy edge device or computer that enables the user to execute several neural networks in parallel. This edge computing device is GPU-enabled and is an excellent candidate for the applications of AI and deep learning applications like speech processing, object detection, image classification, and segmentation, and many more.

3.2.1 Specifications Overview:

Table 1 Specs Overview

GPU	128-CORE, MAXWELL BASED ARCHITECTURE
CPU	QUAD-CORE ARM A57 @ 1.43 GHZ
STORAGE	MICROSD CARD
MEMORY	4 GB RAM, 64-BIT LPDDR4 25.6 GB/S
USB	4X USB 3.0, USB 2.0 MICRO-B
OTHERS	GPIO, UART, I2C, I2S, SPI
DISPLAY	HDMI AND DISPLAY PORT
MECHANICAL	69 MM X 45 MM, 260-PIN EDGE CONNECTOR

The official OS on which the Jetson Nano operates is the **Linux4Tegra**, which is based on **Ubuntu 18.04.** The SDK supported by NVIDIA Jetson Nano is called **JetPack.**

3.2.2 Basic Setup:

The steps in setting up our edge device are:

- Download the *Jetson Nano Developer Kit SD Card Image* from the Jetson Download Center (The Official Source).

- Flash it to the microSD card using the *Etcher* S*oftware* (a software which writes nano program image onto the SD card).
- Now we will load the microSD card by inserting it inside the SD card slot.
- Now we can connect the power source to our nano, which will automatically boot up our system.

3.2.3 Prerequisites For Python

We type the following commands for the installation of python prerequisites :

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install git cmake python3-dev nano
sudo apt-get install libhdf5-serial-dev hdf5-tools libhdf5-dev zlib1g-dev zip libjpeg8-dev
```

Now we have to install ***pip3***. Pip is a tool used for installing python packages from *Python Package Index.*

```
sudo apt-get install python3-pip
sudo pip3 install -U pip testresources setuptools
sudo pip install virtualenv virtualenvwrapper
```

The next the next step is to set up a virtual environment for Python so that we have a completely isolated virtual environment for our project. To set up a python virtual environment we will use the ***virtualenv*.** It's a python tool that enables its users to build isolated environments in python.

```
mkvirtualenv ml -p python3
workon ml
```

Now we just have to download python v3 and install it.

3.3 Increasing Swap Memory

Memory swapping is vital for our project. It is such a computer technology which permits an OS to provide additional memory to an executing application or a process than it is present in the physical RAM. Memory swapping is done when RAM space is limited or exhausted. The OS utilizes memory swapping to get additional memory from the *secondary memory* which, in our case, is the microSD card.

In Jetson Nano, to swap portions we require to build a *swapfile*. Currently, the Nano has a RAM space limited to 4GB that will not be adequate to compile all our AI models.

By default, there is 2GB of swap memory, but it is still not enough to fulfill the needs of our project. Hence, we’ll set up a ***swapfile*** to increase the swap memory from 2 GB to 6 GB. Afterwards, we must reboot our system to confirm that the *swapfile* is executing aptly.

3.4 Peripherals Interfacing

Since a multitude of sensors is utilized to perform the various functions, hence they must be interfaced first with the Jetson Nano. All the health and ultrasonic sensors are connected to the Arduino. It processes the data from the sensors and redirects it to the Jetson Nano for further processing. For interfacing, the Arduino to Jetson Nano, coding in python language is to be done. Programming and interfacing the sensors to Arduino is done using the Arduino IDE.
We are making use of a keypad to control and select the different tasks required. each button corresponds to a certain task.
The Jetson Nano is programmed in Python using a specific library audio library to give the speech output. The audio output is given through the speakers or wireless headphones to user.

3.4.1 PyAuto GUI

It’s a GUI automation Python module with cross-platform support automatically enabled. It allows python to control the mouse and keyboard, and other peripherals for GUI automation tasks. To install it we use the command:

pip install pyautogui

3.4.2 Interfacing Arduino to Jetson Nano

We are working with Arduino along with Jetson Nano because we need to retrieve sensory data from the Arduino and serially send that to the main processor. The method we are adopting for the interfacing is to connect your Arduino simply via a USB cable using Python.
First, connect your Arduino with Jetson Nano through the USB cable.

After that access, we will open the command terminal on Jetson Nano and type in the following command.

```
ls /dev/ttyA*
```

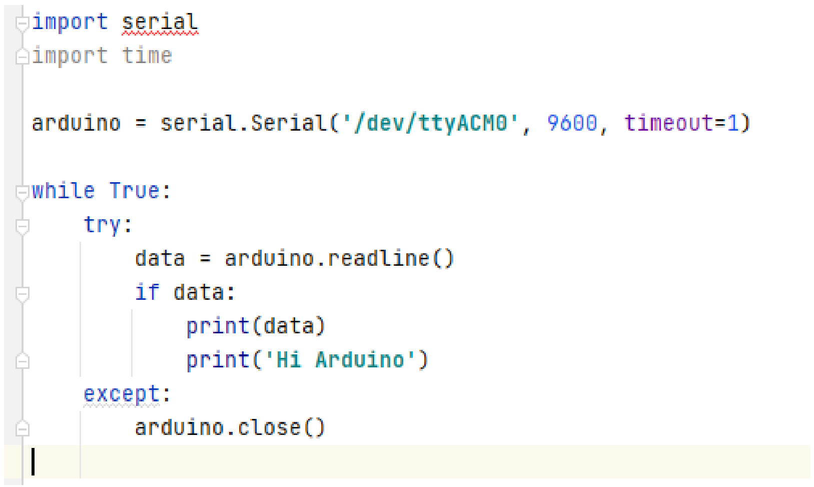
An Output received from the ***/dev/ttyACM0***, implies that the Arduino has been configured to Nano and will now be recognizable by it because ***ttyACM0*** is an indication of the connection between two USB devices.
Afterwards we need to install *PySerial* which is a library that provides support for serial connections between devices. For installation type in the following command

```
pip3 install pyserial.
```

Now the library is installed so we can import it later on. Opening the Arduino IDE and write the following

```
void setup()
{ Serial.begin(9600); }

void loop()
{
  if (Serial.available())
  {  Serial.println("Hello Jetson!"); }
}
```

Create a python file with the following code



```
import serial
import time

arduino = serial.Serial('/dev/ttyACM0', 9600, timeout=1)

while True:
    try:
        data = arduino.readline()
        if data:
            print(data)
            print('Hi Arduino')
    except:
        arduino.close()
```

On running this python script, serial communication between Arduino and Jetson Nano will start.

3.4.3 Interfacing Keypad to Jetson Nano

We have used a *numeric* keypad to select the functionalities. To user can select the options he requires on his demand by pressing the specified keys. The keypad has a USB 2.0 port attached with plug and play feature, hence making its interfacing simple and clear.

3.4.4 Interfacing Camera to Jetson Nano

We are working with a camera along with Jetson Nano because we need a live camera feed as input to all our models i.e., face recognition, currency recognition and image captioning models. The way adopted for the camera interfacing is to connect the camera via its USB connector.
The camera used for all live feed is the ***Philips Real CMOS S988 USB*** camera. The Philips CMOS supports still images as well as HD videos with additional several resolutions included such as 1080p@30FPS, 720p@60FPS, and VGA90 but we are using the 30 FPS. The main edge which we have while using this camera is that it is USB 2.0 supported and has a plug-and-play feature.

3.5 Speech Output with Jetson Nano

For the audio output, we are employing a python text to speech library known as ***pyttsx3***. The main advantage to using this is that it works offline and provides support for multiple Text to Speech engines including: -

- sapi5
- nsss
- espeak

This library has three fundamental parts which are

- The Engine Factory
- The Engine Interface
- The Voice Metadata

The pyttsx3 library includes the ***pyttsx3.init()*** factory function which refers to a ***pyttsx3.Engine*** ; this is the main engine or the engine factory for our text to speech. Amid the construction, the engine initializes a ***pyttsx3.driver.DriverProxy*** object whose primary responsibility is to load a speech engine driver enacting from the ***pyttsx3.drivers*** module. The Engine Interface provides application access to text-to-speech synthesis whereas the ***pyttsx3.voice.Voice*** is the voice metadata which holds information about a speech synthesizer voice. After the construction, the application utilizes the engine object to register the event callbacks, controlling the event queue, production of speech, initializing and stopping event loops and other important functionalities.

CHAPTER 4: OBSTACLE DETECTION

4.1 Ultrasonic Sensor

Ultrasonic sensor is such equipment or sensor which is utilized in measuring the distance to an object or an obstacle with the help of ultrasonic sound waves. It utilizes a transducer to send and receive ultrasonic pulses. Through the reflection time of pulses, we come to know about the object's position. A distinct echo pattern is produced when high-frequency sound waves are reflected from boundaries or edges.

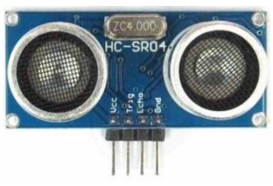


Figure 3 HC-SR04

The working principle upon which this sensor is based is fairly simple. An ultrasonic pulse (40kHz) is transmitted in the air and if an obstacle/object is encountered then the pulse is reflected back to it. The distance of the object is obtained by calculating the travel time period and the speed of sound.
The formula used is simply the speed formula

Speed=Distance/Time

we need to calculate time, so it becomes

Time=Distance/Speed

Distance is 2 meters (1 meter to travel forward and 1 to reflect back) and speed is **343m/s.**
By using the above time equation:
The speed of sound in air is 343 m/s and since we are detecting any obstacle in 1-meter range so that would t=2/343 which nearly 4 milliseconds period between transmitter and receiver.

If the receiver detects the wave after 4 milliseconds of the wave being transmitted, then that means there is some obstacle within 1 meter.

4.1.1 Connecting Ultrasonic Sensor to Arduino

We used an Arduino. Firstly, we need to connect the ultrasonic sensor with the Arduino.

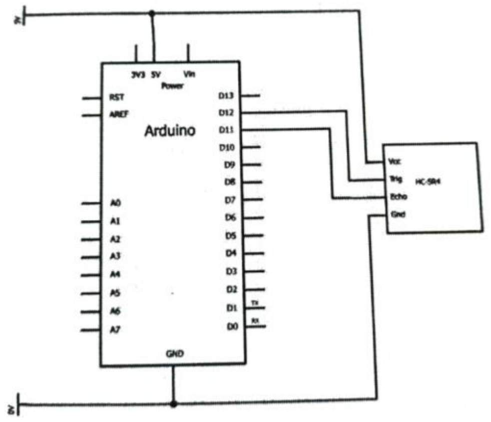


Figure 4 Arduino Circuit Diagram

To type our code in the Nano we have to first install *Arduino Sketch* coding. After typing the code we will compile it and forward it to the Arduino board.

Plug Arduino into the USB cable and into the nano. Once we upload Arduino, we can then compile and activate the code.

4.1.2 Compilation and Code Execution

The code used in our device gives the distance (in cm) to the closest object/obstacle.

```
// This uses Serial Monitor to display Range Finder distance readings


// Include NewPing Library
#include "NewPing.h"

// Hook up HC-SR04 with Trig to Arduino Pin 9, Echo to Arduino pin 10
#define TRIGGER_PIN 9
#define ECHO_PIN 10


// Maximum distance we want to ping for (in centimeters).
#define MAX_DISTANCE 400

// NewPing setup of pins and maximum distance.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
float duration, distance;


void setup()
{
Serial.begin(9600);
}

void loop()
{
// Send ping, get distance in cm
distance = sonar.ping_cm();

// Send results to Serial Monitor
Serial.print("Distance = ");

if (distance >= 400 || distance <= 2)
{
Serial.println("Out of range");
}
else
{
Serial.print(distance);

Serial.println(" cm");
}
delay(500);
}
```

CHAPTER 5:
HEALTH MONITORING SYSTEM

5.1 Temperature sensor

DS18B20 is a temperature sensor that provides 9-to-12-digit temperature readings. These values show the temperature of a specific device. The communication of this sensor can be completed through the single-wire bus protocol, which uses the data line to communicate with the internal microprocessor. Furthermore, it obtains its power from the data line hence no external power source is required. The range from which the temperature can be calculated is from -55°C to +125°C.

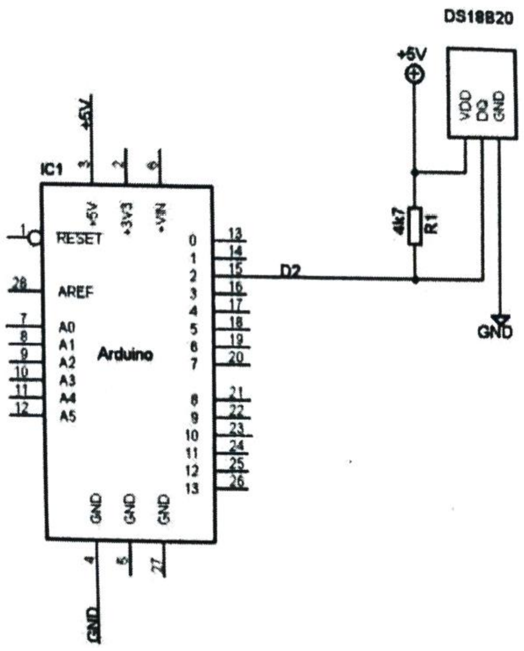


Figure 5 Temperature Sensor Circuit Diagram

5.1.1 DS18B20 Pin Configuration

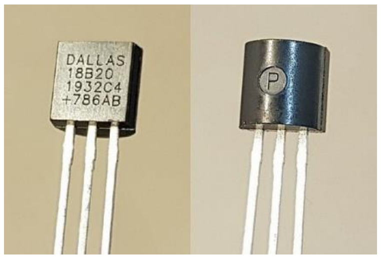


Figure 6 DS18B20

Pin 1: Ground terminal

Pin 2: Power supply (Vcc) from 3.5V to 5V

Pin 3: Data pin: It provides temperature value, which communicates using the single-wired method.

5.2 Working Principle:

DS18B20 has three pins, which are power, data, and ground terminals. We connect the sensor's power and ground terminals to the Arduino's power and ground and the sensor's data pin to the Arduino's digital I \/ O pin 2. Then we can use our code to calculate the user's temperature. To get the temperature measurement, we need to issue a command. When received by the sensor, it will initiate a data dialog and all measured values are stored in the sensor's RAM. We can read it to get data or write to it to specify the resolution of the sensor. To read the data, we issue a command and receive 9 bytes of data. Then use the following formula to determine the temperature.

Temperature = ((high byte << 8) low byte) * 0.0625

Then convert this value to Fahrenheit and display it to the user.

The code for this is:

```
#include <OneWire.h>
#include <DallasTemperature.h>


// Data wire is plugged into digital pin 2 on the Arduino

#define ONE_WIRE_BUS 2


// Setup a oneWire instance to communicate with any OneWire device
OneWire oneWire(ONE_WIRE_BUS);

// Pass oneWire reference to DallasTemperature library DallasTemperature sensors(&oneWire);

void setup(void)
{
sensors.begin(); // Start up the library
Serial.begin(9600);
}


void loop(void)
{
// Send the command to get temperatures
sensors.requestTemperatures();

//print the temperature in Celsius
Serial.print("Temperature: ");
Serial.print(sensors.getTempCByIndex(0));
Serial.print((char)176);//shows degrees character
Serial.print("C | ");

//print the temperature in Fahrenheit
Serial.print((sensors.getTempCByIndex(0) * 9.0) / 5.0 + 32.0);
Serial.print((char)176);//shows degrees character
Serial.println("F");

delay(500);
}
```

5.3 Pulse Sensor:

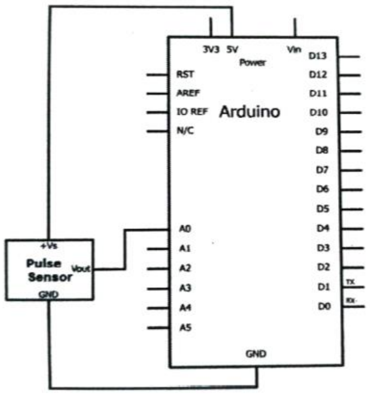


Figure 7 Circuit Diagram for Pulse Sensor

Pulse wave refers to a change in volume of a blood vessel that ensues while the heart starts pumping the blood. The pulse sensor is such a device that detects as well as monitors the change in volume. Output is obtained from the pulse sensor when the user connects his fingertip to it. The sensor consists of 24 inches' color code wire, ear clip, Velcro Dots-2 transparent stickers-3.

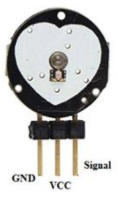


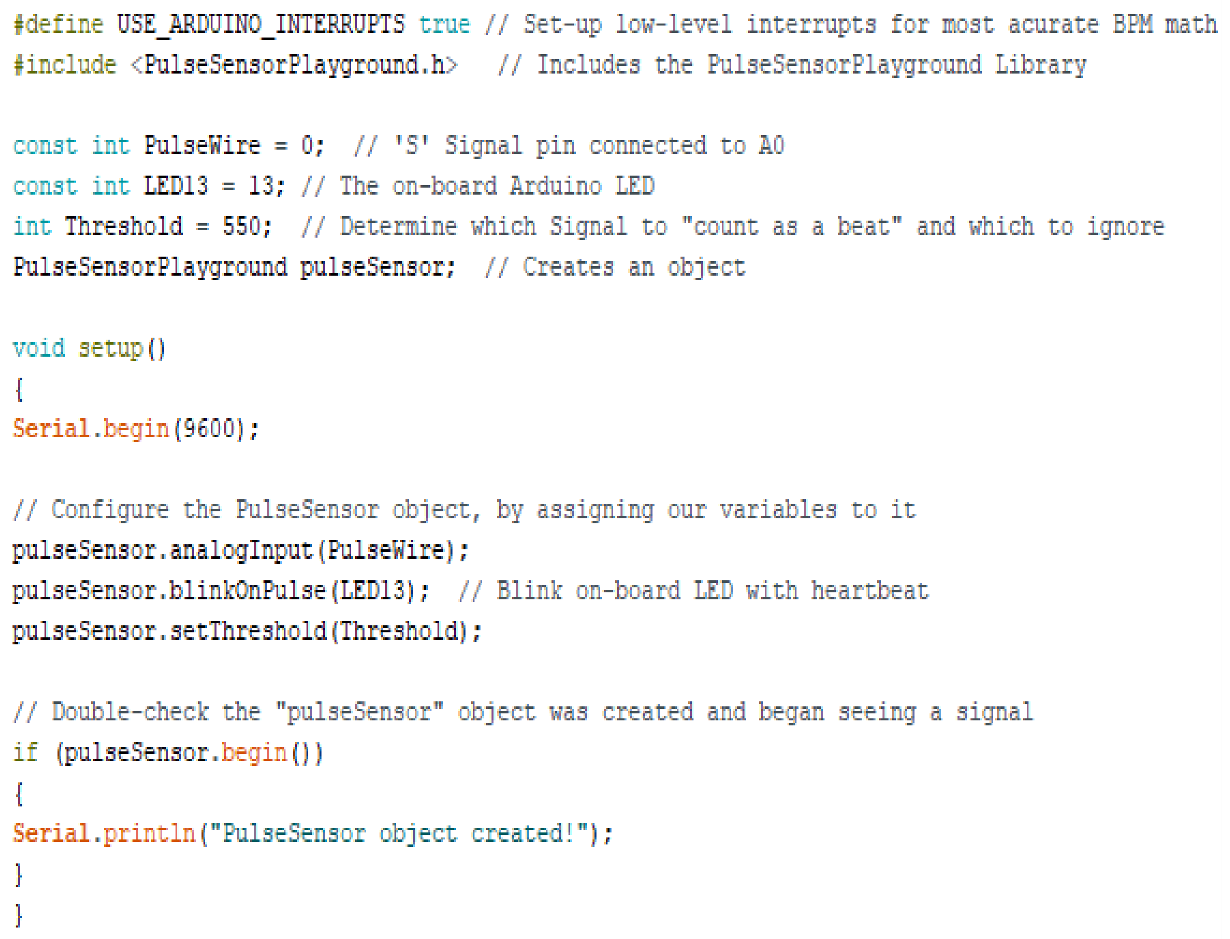
Figure 8 Pulse Sensor Pin Configuration

Pulse sensor is open source, with plug-and-play hardware. It effortlessly integrates real-time heart rate information into our project. The sensor consists of two circuits which act similar to an optical amplifier with noise eliminating too. Due to these circuits, pulse reading is very easy and fast. The sips at 5v power draw just 4mA.

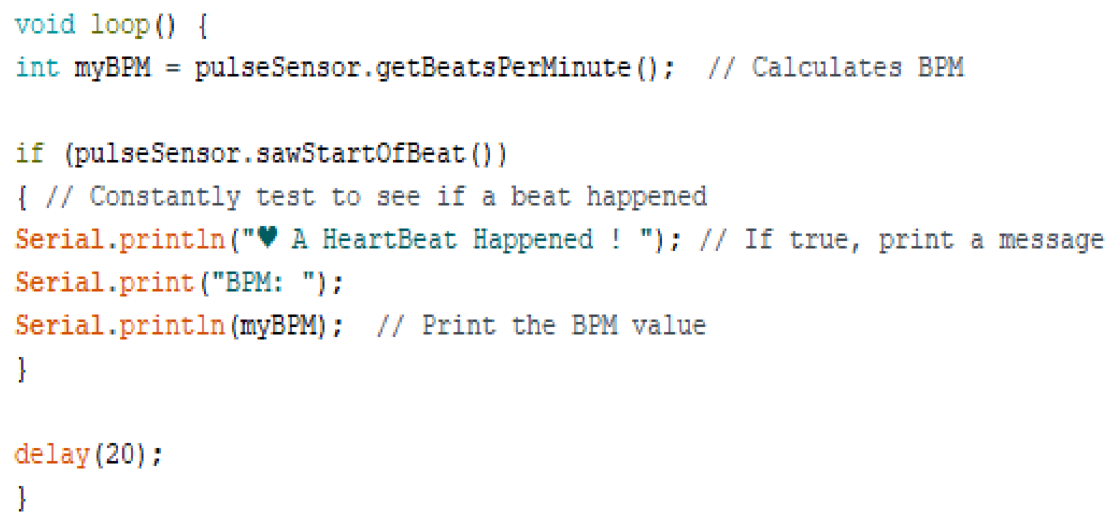
Basically, the pulse sensor consists of three pins which are GND, VCC and SIGNAL. The black wire is the ground terminal, the red is the supply wire, and the third purple is the output signal wire

After connection, we use the VCC pin and GND pin to supply power. The pulse sensor operating voltage is 5V or 3.3V. First, we connect the sensor to an Arduino, then we can run the code.

The Arduino code for this sensor is:



```
#define USE_ARDUINO_INTERRUPTS true // Set-up low-level interrupts for most acurate BPM math
#include <PulseSensorPlayground.h>   // Includes the PulseSensorPlayground Library

const int PulseWire = 0;  // 'S' Signal pin connected to A0
const int LED13 = 13; // The on-board Arduino LED
int Threshold = 550;  // Determine which Signal to "count as a beat" and which to ignore
PulseSensorPlayground pulseSensor;  // Creates an object

void setup()
{
Serial.begin(9600);

// Configure the PulseSensor object, by assigning our variables to it
pulseSensor.analogInput(PulseWire);
pulseSensor.blinkOnPulse(LED13);  // Blink on-board LED with heartbeat
pulseSensor.setThreshold(Threshold);

// Double-check the "pulseSensor" object was created and began seeing a signal
if (pulseSensor.begin())
{
Serial.println("PulseSensor object created!");
}
}
```



```
void loop() {
int myBPM = pulseSensor.getBeatsPerMinute();  // Calculates BPM

if (pulseSensor.sawStartOfBeat())
{ // Constantly test to see if a beat happened
Serial.println("♥ A HeartBeat Happened ! "); // If true, print a message
Serial.print("BPM: ");
Serial.println(myBPM);  // Print the BPM value
}

delay(20);
}
```

CHAPTER 6: FACE RECOGNITION SYSTEM

6.1 Introduction

Facial recognition insinuates the detection and identification of a person. Whenever the user requires this functionality, he will press the dedicated button for it which in turn invokes it. The faces are detected using the mounted camera with the help of Jetson Nano. The result is then shared in the form of audio given via the headphones.

Face recognition harnesses the machine learning (ML) algorithms and deep learning. ML can be regarded as a sub-branch of AI in which the system learns to identify data and make decisions based on a provided dataset. Deep learning is referred to as a sub-branch of ML which employs ML algorithms and data sets to train/instruct its deep neural networks for enhanced accuracy. Our system is set up in such a manner that almost any task can be completed or automated with minimal human intervention.

NVIDIA Jetson Nano developer kit has a *Mobile Industry Processor Interface* (MIPI) powered *Camera Serial Interface* (CSI) port. It can support several camera modules like the Raspberry Pi camera and others. The camera used in our device is the Philips CMOS, model s988, which is compact yet suitable for machine learning and computer vision applications like facial recognition. Our device supports facial recognition for four persons.

6.2 Algorithm

The main scheme on which the facial recognition model works is Machine Learning and Deep Learning. The main difference between ML and DL is that ML makes use of different algorithms to learn from the data set and make well-informed decisions based on what it has learned whereas DL organizes the algorithms in such a manner that they form an ***Artificial Neural Network*** (ANN) which has the ability to learn and educate itself and additionally make its own decisions intelligently.

6.2.1 Neural Networks

They are a series of algorithms that identify relationships and patterns in data. ANN has similar functionality like that of a human. It is made up of hundreds and/or thousands, sometimes millions of artificial neurons known as the *processing units*. They are interlinked to each other with the help of *nodes*. Traditionally, all the inputs and outputs are independent of each other. It employs a set of learning algorithms known as ***backpropagation*** to learn regarding the information fed in it and produce a result or output.

The main advantage of using backpropagation is that the network can also work backward i.e it can go from the output unit back to the input unit to adjust weights of the connections between its units hence producing the lowest possible error between the actual and the desired outcome.

6.2.2 Processing Flow

Data acts as the main fuel since ML algorithm learns from these enormous data set. These datasets are used to educate the deep neural networks. The main phases that the ML algorithm undergoes for facial recognition are :

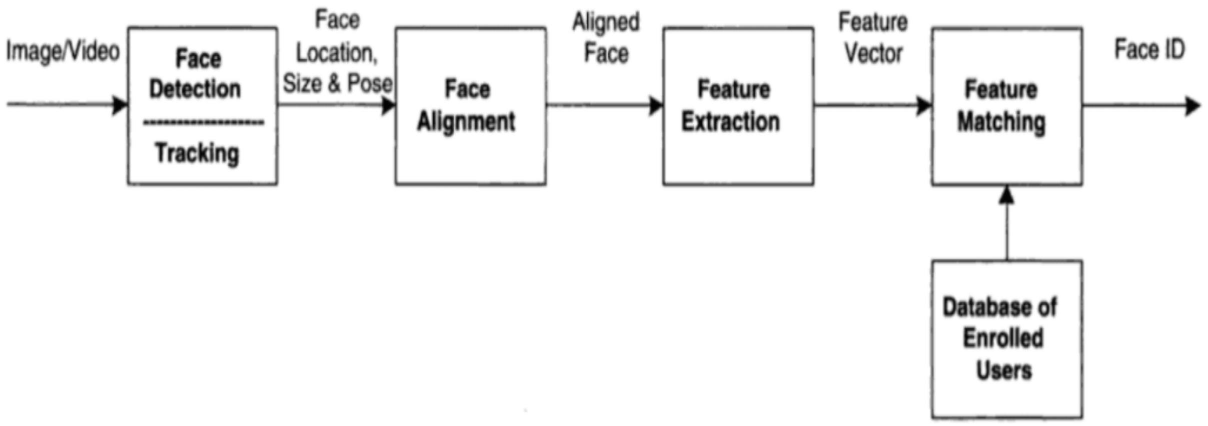


Figure 9 Facial Recognition Processing Flow [12]

- **Face Detection :** Locating faces in an image.
- **Face Alignment:** Adjusting or normalizing the faces found.

- **Feature Measurement and Extraction**: The algorithm uses data to identify different patterns and features from the face to form a *feature vector*. ML algorithm also decides which classes are to be and which are relatively close. This process is known as *embedding*.
- **Face Recognition or Feature Matching:** ML algorithm will match the feature vector of the input face against the feature vector of known faces in the database. If the difference between them is close then it means that the face is matched.

The algorithm uses data to identify different patterns and form a feature vector. It implies that input is in the form of an image and the output is in the form of a feature vector. A ***feature vector*** is regarded as an N-dimensional vector containing numbers, that manifest the different features of the image or generally the object. ***Embedding*** is the formation of such a vector that contains data of features which are semantically close to each other. The main classes formed in our feature vector are:

1. Height of face
2. Width of face
3. The average color of the face (R, G, B)
4. Width of lips
5. Height of nose
6. Other Secondary classes

Once every image is encoded into a feature vector the next step is the comparison. The feature vector is then matched with the databases of feature vectors of a known person. If the difference between them is close then it means that the face is matched. The difference is according to the set threshold which is known as the ***tolerance level*** in our case the tolerance level is 0.3.

This technique can be applied using image processing techniques such as ***haar*** but the main reason for using machine learning is that it can help out with two main things:

1. **Derivation of Feature Vector:** Listing all the classes of the features is an extensive and tedious job as an enormous number of classes exist. ML algorithms can tremendously help us out as it intelligently makes use of such secondary features.

2. **Matching the Algorithm**: A ML algorithm intelligently matches the new images feature vector with the predefined feature vectors in the corpus

Several python deep learning libraries are used which assist us in implementing the ML functionality in our program.

6.2.3 Libraries Utilized:

- Face_recognition
- Numpy
- OpenCV
- Python Imaging Library (PIL)
- Image and ImageDraw (from PIL)

6.2.4 Face_recognition

The main aim of using the *face_recogntion* library is *matching* i.e to check the similarity of the input face to the stored pictures list. It also yields a numerical measurement that corresponds to the similarity amidst the input face and all known faces list. Criteria for selection set is that the lower the number obtained, the more similar the faces. A threshold level is also set known as the tolerance level.

6.2.5 OpenCV and its Prerequisites:

OpenCV is an open-source library of python whose main focus is real-time computer vision. It includes attributes like image processing, object detection, face detection, etc. The functionality that OpenCV performs is to facilitate the face_recognition library with a deep learning network.

To use OpenCV, CUDA and cuDNN support are required. CUDA stands for *Compute Unified Device Architecture* and it refers to 2 things:

1. **The CUDA Architecture:** A parallel GPU architecture.
2. **The CUDA Software Platform And Programming Model**: A type of API used to program GPUs for general purpose processing.

CUDA is a vital part as it enables Its user to speed up different intensive calculations by utilizing the GPU for parallel processing.

cuDNN refers to CUDA Deep Neural Network. It represents a deep learning library used for GPU acceleration and is based upon CUDA.

The main difference between CUDA and cuDNN is that CUDA is similar to a workbench which has many tools inside it whereas cuDNN is one of its tools. Hence, to run a deep neural network based on CuDNN, we must first install CUDA. The JetPack already has in-built support for CUDA and cuDNN.

Other prerequisites for OpenCV include:

- Dlib
- Cmake

Additionally, the JetPack in-built support for OpenCV too as well which just needs to be activated. The main purpose of using openCV is to utilize the GPU for the deep learning.

OpenCV by default comes with a pre-trained Haar cascades but currently we are not using this in our model. We are using the more accurate, deep learning-based face detector which can be found in the official release of OpenCV in GitHub.

6.3 Results:

Figure 10 Wajiha Jawwad

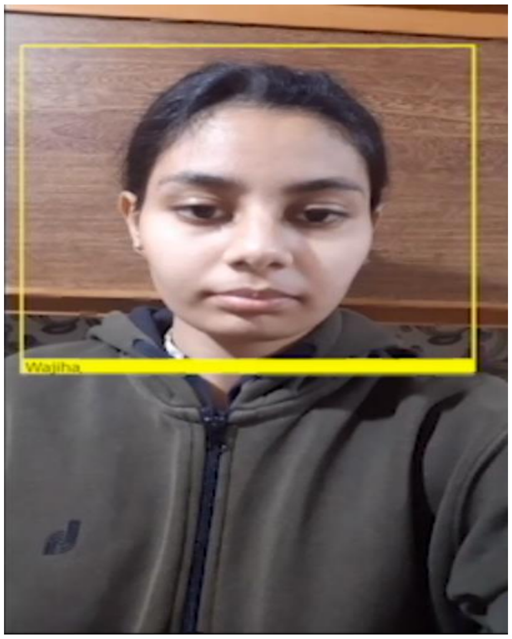


Figure 11 Pervaz Alam

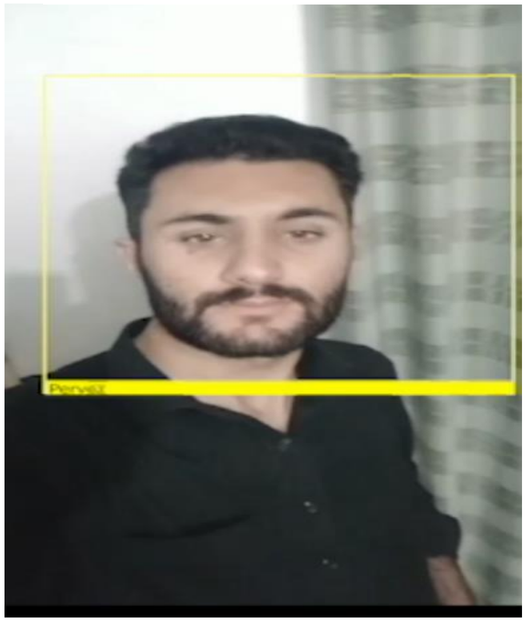
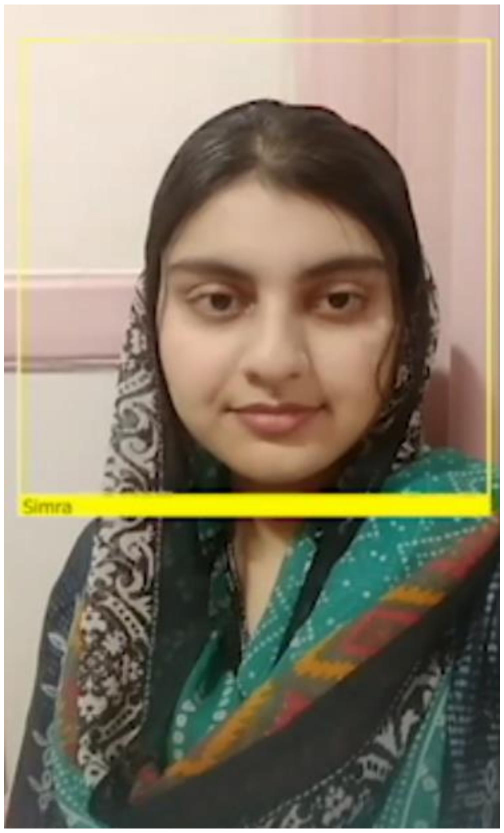
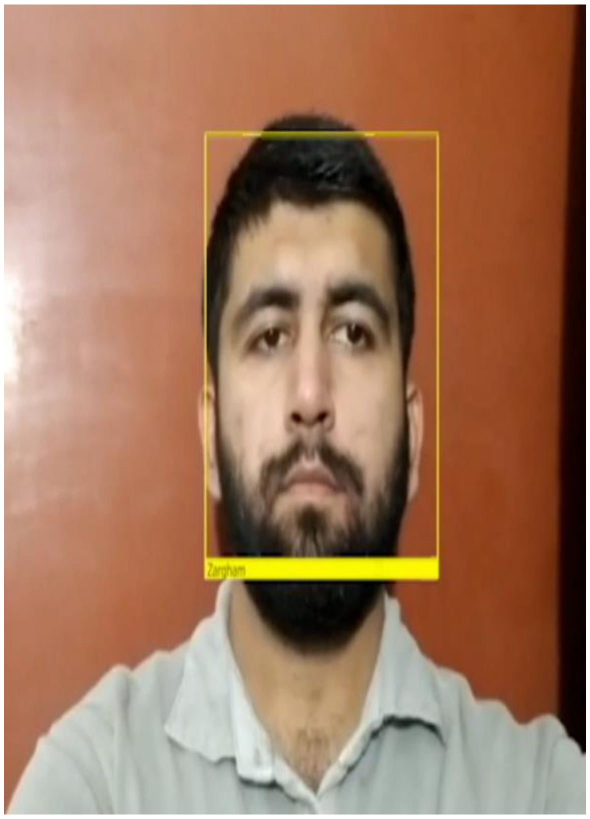


Figure 12 Simra Kausar Raja

Figure 13 M. Zargham Baig





CHAPTER 7: IMAGE CAPTIONING

7.1 Introduction

Image Captioning refers to generating a textual description for a given image. Basically, its purpose is to take an image as input and produces a relevant caption as output against it. In our project, we have given a video input and the program takes its frames into account. With the emergence of Deep Learning, along with numerous datasets also available, this technique has evolved significantly. The motivation to understand image captioning lies in its real-world scenario applications. Aid to blind people is the most important application because a product can be made for such people that guides them while navigating with their surrounding environment without the assistance of someone else. This can be accomplished by first converting the real-time scene around them into a textual caption and then convert it into audio.

7.2 Algorithm

There are 2 major parts of our algorithm. First is while looking at an image we are forming the description as we are seeing the image, secondly, at the same time, we are looking to create a meaningful sequence of words. The first part is handled by ***Convolutional Neural Networks*** and the second is handled by ***Recurrent Neural Networks***.

7.2.1 Convolutional Neural Networks (CNN)

CNN are a type of neural network (already discussed in section 6.2.1) that analyzes visual imagery in a grid-like topology. It performs both generative and descriptive tasks when used in combination with recommender systems and natural language processing (NLP). There are multiple layers present in a CNN. These layers are generally input, output and hidden layers. These include multiple types of layers as discussed below.

- ***Layer:1 Convolutional Layer,*** this layer performs the convolution of information with a filter/kernels so that specific features can be detected on an image.

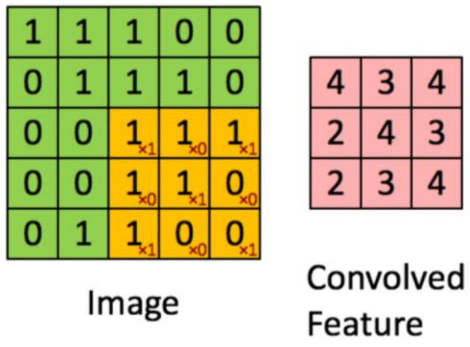


Figure 14 Example of convolved feature [13]

- ***Layer: Activation Layer,*** it harnesses the Rectified Linear Unit, also known as ReLu which refers to rectifier function. It is multiplied with the convoluted input with the intent to extend the non-linearity among the network.
- ***Layer:3 Pooling Layer,*** this layer reduces the size of the input so that a reduced number of parameters to be computed. It basically down-samples the features. It utilizes a non-overlapping filter of 2x2 accompanying a stride of 2 that returns the max value of features. Purpose of max filter is to give the max value inside the features inside the region.

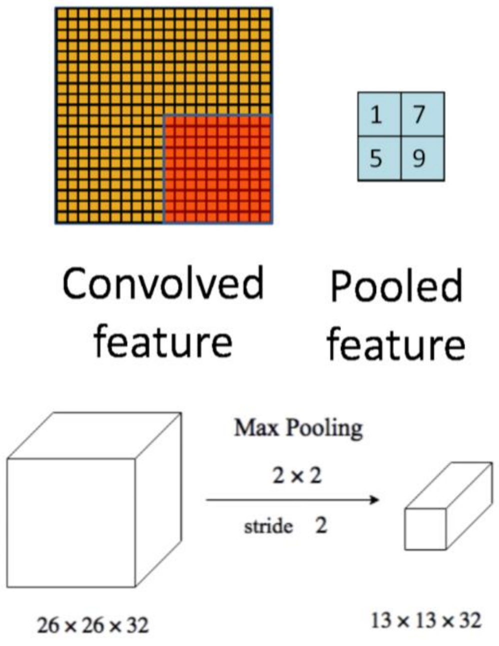


Figure 15 Example of pooling layers working [13]

- ***Layer:4 Fully Connected Layer,*** this layer connects all inputs from one layer to the next layer activation units. This layer includes flattening which enables the user to learn new nonlinear combinations of features and combine these features together for building a model.

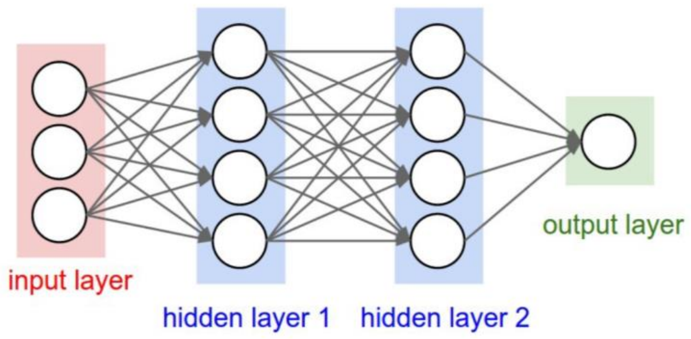


Figure 16 Example of connecting layers working [13]

Ultimately, the result achieved is that of an activation function that will help classify the output.

7.2.2 Recurrent Neural Networks (RNN)

RNN is of neural network that saves the output of a particular layer and feeds that output as input in the next step to predict the output of the next layer while using sequential data or time-series data. They have a Hidden Layer or ***Internal Memory*** which retains information about the sequence and previous inputs.

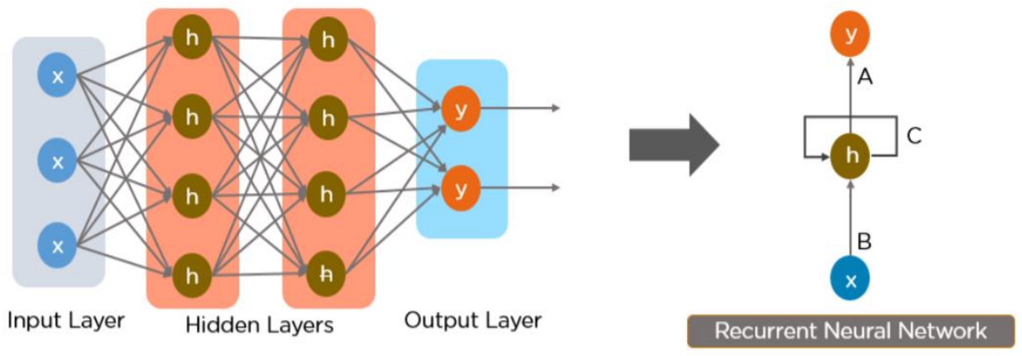


Figure 17 Converting Feed-Forward NN to RNN [15]

7.3 Algorithm Concept:

First, we will import the necessary libraries. Using libraries is necessary as they are collections of prewritten code that users can use to optimize a task.

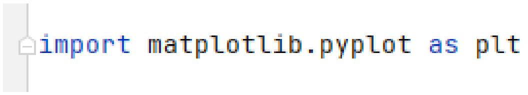
7.3.1 Importing Libraries

The libraries used in our code are:

- Tensorflow
- Matplot
- Numpy
- JSON
- pyttsx3
- cv2

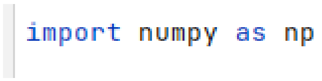
```python
import tensorflow as tf
```

TensorFlow is a foundation library used to create Deep Learning models directly.



```python
import matplotlib.pyplot as plt
```

Used to generate plots and graphs of attention for visualizing on what features our model is focusing on while captioning them.



```python
import numpy as np
```

NumPy is used to perform a wide variety of mathematical operations on arrays in a fast and efficient way by manipulating numerical data inside them.

```python
import json
```

JSON is used for storing and exchanging data between Python objects and JavaScript object notation strings.

```python
import pyttsx3
```

This library provides an engine that gives text-to-speech conversion functionality. Its detailed functionality is already discussed in section

```
import cv2
```

OpenCV is used to provide various functions for image and video processing. Its prerequires are already discussed in section 6.2.5.

7.3.2 GPU Configuration

Used to configure the GPU (Graphic Processing Unit) memory for TensorFlow otherwise TensorFlow will allocate all of the available GPU memory when it is started. So, we have allocated GPU memory.

```
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth=True
sess = tf.compat.v1.Session(config=config)
```

7.3.3 Capturing Live Camera Feed

```
cap = cv2.VideoCapture(0)
```

Used to video capture live objects from the camera. The ‘0’ specifies that we are taking a feed from the webcam.

```
    ret, frame = cap.read()
```

we can retrieve frame by frame from the video feed.

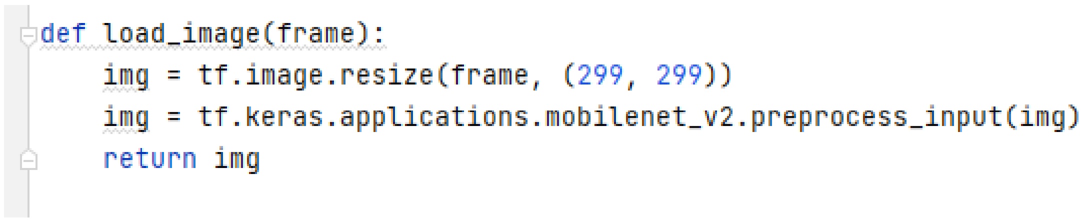
```
     img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

OpenCV actually reads colors as BGR (Blue Green Red), where most models read as RGB (Red Green Blue), so we convert the frames as such.

7.3.4 Loading the frames

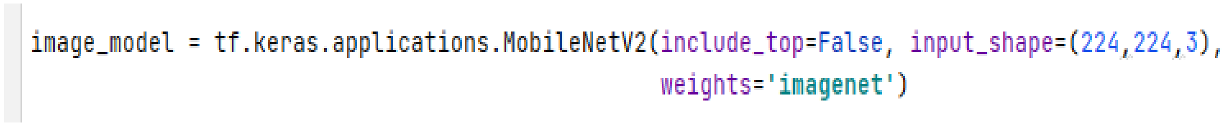
First, we will convert the frames into Mobile Net V2's expected format by resizing the image to 299px by 299px

Next, preprocess the images using the ***preprocess_input*** method to normalize the image so that it contains pixels in the range of -1 to 1, which is meant to adequate your image to the format that our model requires.



```
def load_image(frame):
    img = tf.image.resize(frame, (299, 299))
    img = tf.keras.applications.mobilenet_v2.preprocess_input(img)
    return img
```

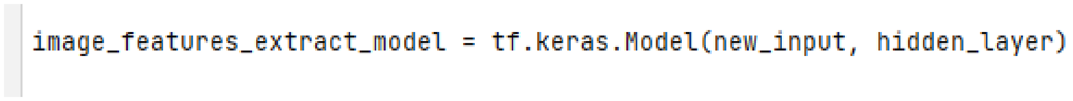
7.3.5 Initialize MobileNetV2 and load the pretrained ImageNet weights



```
image_model = tf.keras.applications.MobileNetV2(include_top=False, input_shape=(224,224,3),
                                                weights='imagenet')
```

This function returns a Keras image classification model, which is our base CNN, loaded with weights pre-trained on ImageNet i.e., an image database.

7.3.6 Feature extraction

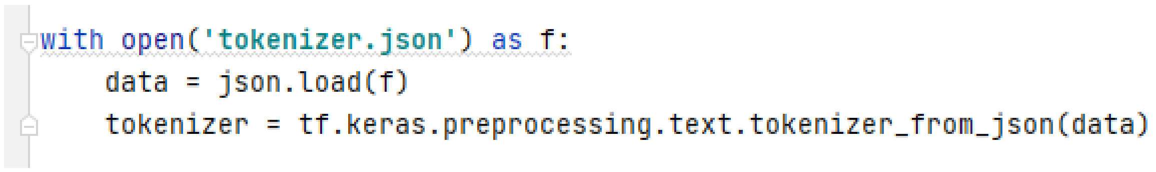


```
image_features_extract_model = tf.keras.Model(new_input, hidden_layer)
```

The CNN is performing feature extraction whose purpose is to compress the information in the original image frames into smaller dimensions. Since it is encoding the information of the image here the CNN acts as an encoder. The ***image_features_extract_model*** is our deep CNN encoder, its purpose is to learn the features from the input image.

7.3.7 Tokenize the captions

The captions of a pre-trained model are already tokenized. This gives us a vocabulary of all of the unique words in the available dataset.
We have limited the vocabulary size of words to the top 5,000 words in order to save memory.
The mapping file is used with 5000 unique word-to-index mappings and vice versa.



```
with open('tokenizer.json') as f:
    data = json.load(f)
    tokenizer = tf.keras.preprocessing.text.tokenizer_from_json(data)
```

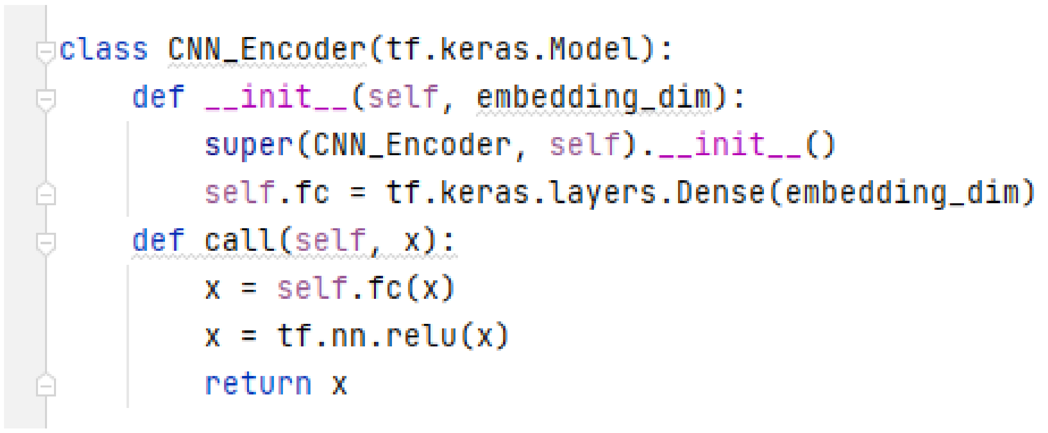
7.4 Model: MobileNetV2

We are using a pre-trained model: ***MobileNetV2*** or *transfer learning* to customize this model according to our needs. MobileNet model is TensorFlow's mobile computer vision model which employs ***depthwise separable convolutions***. MobileNet refers to a class of CNN which has been open-sourced. It notably lowers the amount of parameters resulting in a more lightweight, low-latency deep neural networks which are ideal for our edge device. The major contrast among MobileNet architecture and a typical CNN is that in place of a single 3x3 convolution layer ensued by batch normalization and ReLU, MobileNets splits the convolution resulting in a 3x3 depth-wise convolution with the addition of 1x1 pointwise convolution. MobileNetV2 is similar to MobileNet with the exception that it has inverted residual blocks with bottlenecking features resulting in a more reduced parameter count.

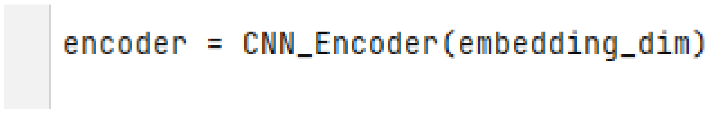
It consists of 2 major parts: Encoder and Decoder. The parts from which we have to extract the features, out of the lowermost convolutional layer present in MobileNet yielding an initial vector of 8×8×2048 dimension. We have to diminish it to a shape of 64x2048. This diminished vector is then transited through the CNN Encoder (constitutes of a single fully connected layer). Finally, RNN (here GRU), the decoder, is responsible to forecast the next word of the image and connect them together.

7.4.1 Encoder

Previously, we had extracted the features and then passed them to our encoder. The job of the encoder is to pass those features over a fully connected layer and we have added a linear dense layer after it so that our model can handle tedious and complex features. The encoder will return the generated string that goes into the Decoder afterward to process.



```
class CNN_Encoder(tf.keras.Model):
    def __init__(self, embedding_dim):
        super(CNN_Encoder, self).__init__()
        self.fc = tf.keras.layers.Dense(embedding_dim)
    def call(self, x):
        x = self.fc(x)
        x = tf.nn.relu(x)
        return x
```

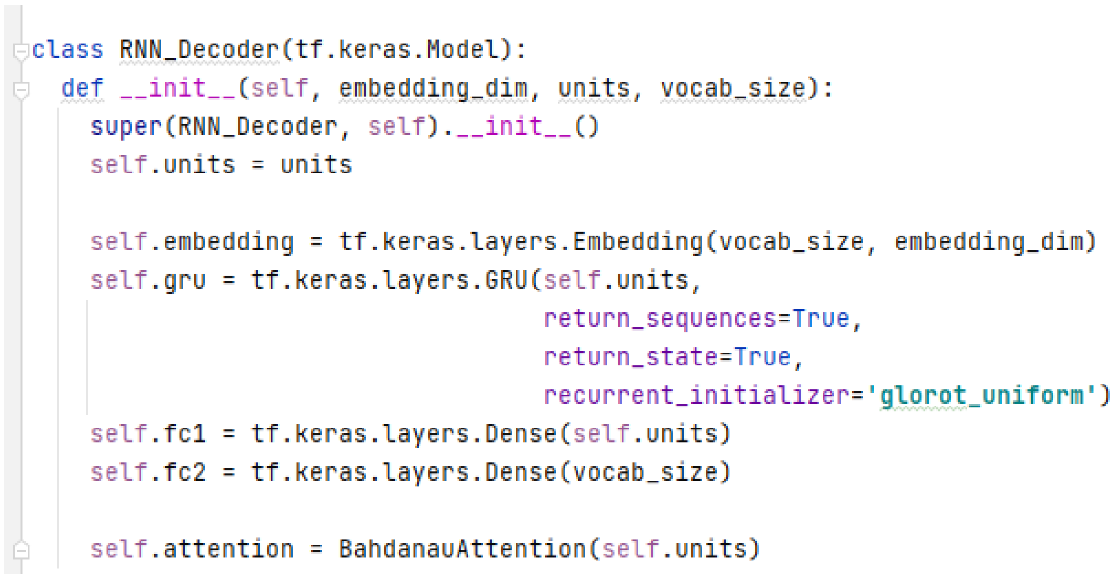


```
encoder = CNN_Encoder(embedding_dim)
```

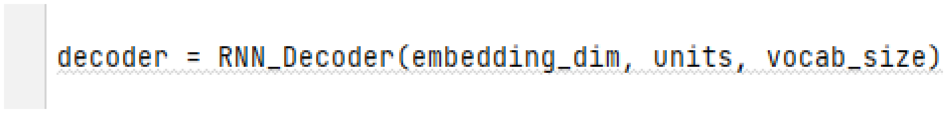
7.4.2 Decoder

The Decoder is our ***Natural Language Processing*** (NLP) Model. For the captions to be generated we have used GRU (Gated Recurrent Units) because it has slightly less complexity in structure as compared to LSTM (Long Short-Term Memory). LSTM layer in this case would have been complex and it would not be deployable on the processor. GRU is used here because we want quick and decent accuracy, also scenarios where infrastructure is an issue just like our processor.

A Single layer of GRU is used followed by two dense layers. Since this is a time series problem where we need to predict the next string based on its connection with the previous string, the addition of dense layers itself is not so efficient so we have added a forced learning attention model.



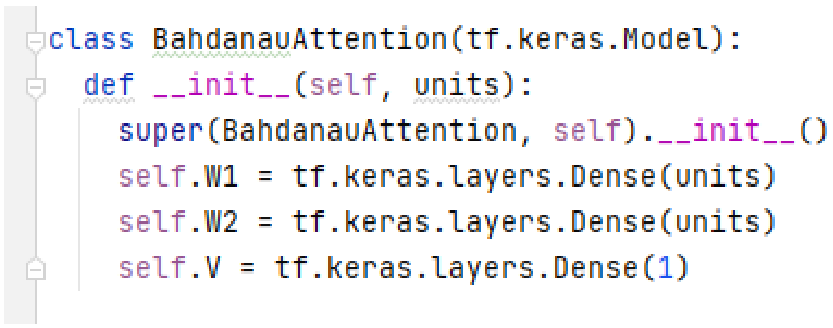
```
class RNN_Decoder(tf.keras.Model):
  def __init__(self, embedding_dim, units, vocab_size):
    super(RNN_Decoder, self).__init__()
    self.units = units

    self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
    self.gru = tf.keras.layers.GRU(self.units,
                                   return_sequences=True,
                                   return_state=True,
                                   recurrent_initializer='glorot_uniform')
    self.fc1 = tf.keras.layers.Dense(self.units)
    self.fc2 = tf.keras.layers.Dense(vocab_size)

    self.attention = BahdanauAttention(self.units)
```



```
decoder = RNN_Decoder(embedding_dim, units, vocab_size)
```

7.4.3 Attention Mechanism

In every sequence of text words, outputs from previous words are used as inputs, in combination with new sequence words. This gives the RNN networks a feature of memory that makes captions more context-aware. RNNs become computationally complex if more layers are simply added on, so generally, in limited memory scenarios, attention models come to help in selecting the most relevant words.



```
class BahdanauAttention(tf.keras.Model):
  def __init__(self, units):
    super(BahdanauAttention, self).__init__()
    self.W1 = tf.keras.layers.Dense(units)
    self.W2 = tf.keras.layers.Dense(units)
    self.V = tf.keras.layers.Dense(1)
```

The ***Bahdanau*** Attention or Additive Attention model refers to taking the floating-point value from the dense layers and is predicting the correct token. This mechanism learns to align and translate jointly, therefore, performing a linear combination of encoder states and decoder states. With the help of few dense layers, it is training such a network that is converting the floating-point input to an integer token which is associated with the previously generated target words. Thus, acts as a ***forced teacher model***.

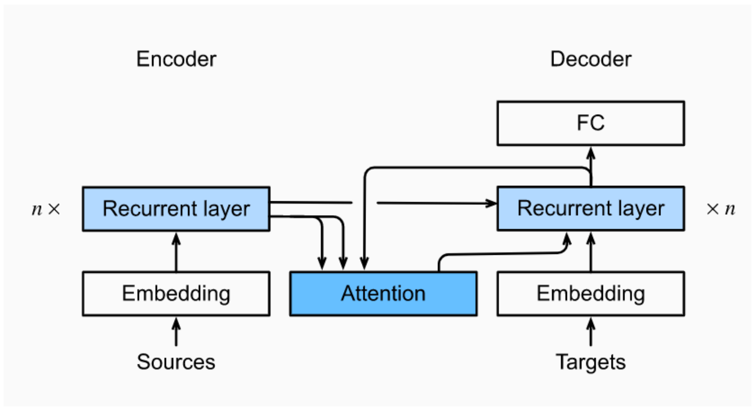


Figure 18 Layers in RNN encoder-decoder model with Bahdanau attention [14]

7.4.4 Evaluate

Now for the part when we run our model with input frames coming from the live feed, we fundamentally run the evaluate function whose jobs are as follows

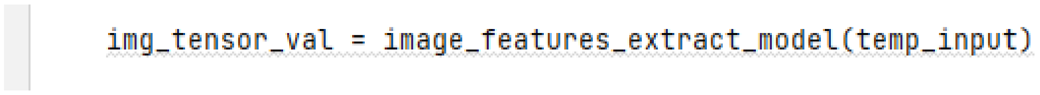
The decoder is called with batch size =1 because we are providing one frame at a time.

```
hidden = decoder.reset_state(batch_size=1)
```

The preprocessed image that is given as an input here has a 3-dimensional matrix i.e. [height, width, channels (RGB)], but to give it as an input to the model we need to increase another dimension that is done by wrapping around it another dimension by expand_dim.

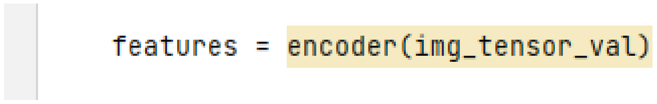
```
temp_input = tf.expand_dims(load_image(image), 0)
```

The expanded dimensional image is passed through the image_features_extract_model and its features are extracted in the variable ***img_tensor_val***



```
img_tensor_val = image_features_extract_model(temp_input)
```

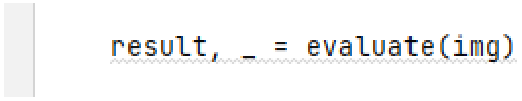
The encoder features are called and saved in ***features*** variable



```
features = encoder(img_tensor_val)
```

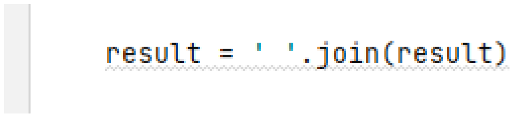
Then the ***features, hidden and img_tensor_val*** variables are passed through a loop to predict the exact token against the string words

The tokenizer returns the string values to the evaluate function, and we are saving the function output in a variable named ‘result’



```
result, _ = evaluate(img)
```

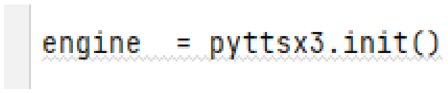
Then the strings are joined together to form the caption



```
result = ' '.join(result)
```

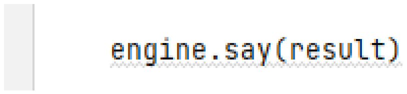
7.5 Text to speech

Lastly, the generated caption has to be converted into audio, which is done first by initializing an engine using the pyttsx3 library. The pyttsx3 has been discussed in detail in section 3.5



```
engine  = pyttsx3.init()
```

Lastly, we pass the result variable that contains our caption to the engine and use the ‘say’ function so it generates audio that can be listened through the headphones



```
engine.say(result)
```

7.6 Results

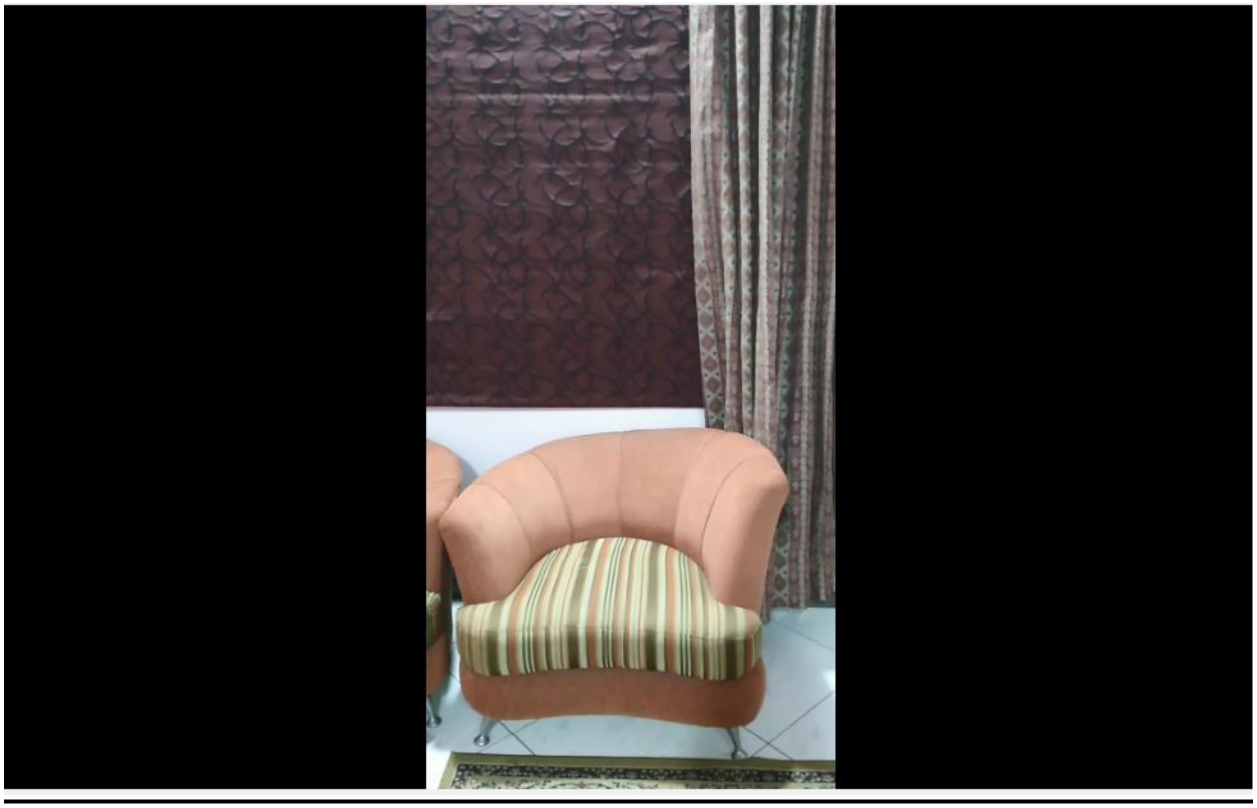
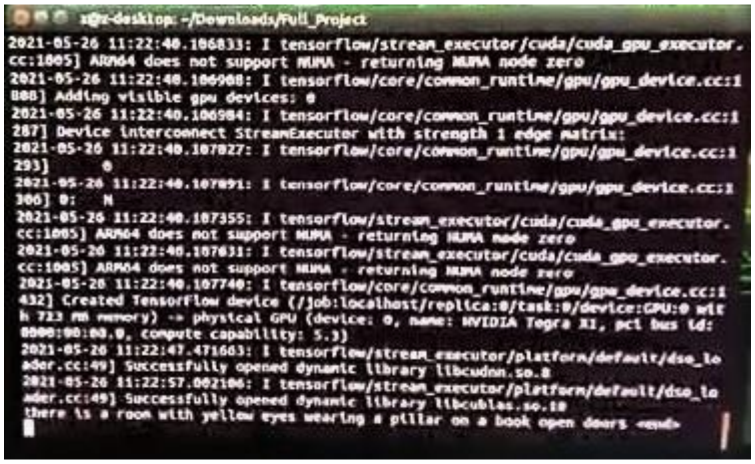


Figure 19 Input



```
x@x-desktop: ~/Downloads/Full_Project
2021-05-26 11:22:40.106833: I tensorflow/stream_executor/cuda/cuda_gpu_executor.
cc:1005] ARM64 does not support NUMA - returning NUMA node zero
2021-05-26 11:22:40.106908: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1
808] Adding visible gpu devices: 0
2021-05-26 11:22:40.106984: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1
287] Device interconnect StreamExecutor with strength 1 edge matrix:
2021-05-26 11:22:40.107027: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1
293]      0
2021-05-26 11:22:40.107091: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1
306] 0:   N
2021-05-26 11:22:40.107355: I tensorflow/stream_executor/cuda/cuda_gpu_executor.
cc:1005] ARM64 does not support NUMA - returning NUMA node zero
2021-05-26 11:22:40.107631: I tensorflow/stream_executor/cuda/cuda_gpu_executor.
cc:1005] ARM64 does not support NUMA - returning NUMA node zero
2021-05-26 11:22:40.107740: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1
432] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 wit
h 723 MB memory) -> physical GPU (device: 0, name: NVIDIA Tegra X1, pci bus id:
0000:00:00.0, compute capability: 5.3)
2021-05-26 11:22:47.471663: I tensorflow/stream_executor/platform/default/dso_lo
ader.cc:49] Successfully opened dynamic library libcudnn.so.8
2021-05-26 11:22:57.002106: I tensorflow/stream_executor/platform/default/dso_lo
ader.cc:49] Successfully opened dynamic library libcublas.so.10
there is a room with yellow eyes wearing a pillar on a book open doors <end>
```

Figure 20 Running Image Captioning Model and Output

The caption displayed is given in the form of audio output to the user. The displayed image is a backend display for the developer.

CHAPTER 8:
CURRENCY RECOGNITION

8.1 Processing Flow:

Pakistani currency notes of PKR.10, 20, 50, 100, 500, 1000, 5000 are identified using the machine learning approach. The steps involved are as follows:

8.2 Using Machine Learning Algorithm (YOLO V5)

It is an ML procedure, which utilizes convolutional neural organizations for the discovery of articles. YOLO, an acronym for You Only Look Once, is one of the most efficient object detection methods available. Despite the fact that it isn't the most precise object detection technique but is one of the robust algorithms available which is very suitable for the purpose of real-time recognition on an edge device.

In contrast with recognition procedures, the detection algorithm's goal isn't just to forecast class labels but also to detect the position of objects. As a result, it can detect several items within an image in addition to classifying them into a category. The architecture of YOLO is like FCNN (fully convolutional neural network) where an image (nxn) is given as input and a prediction (mxm) is given as output.

This Algorithm has modeled detection as a regression problem which is further explained in Section 8.1.1.2. It makes use of a single neural network to a given image; it implies that this network separates the image into regions, or an SxS grid and predicts each region's bounding boxes and probability. To anticipate the bounding box confidence for the boxes and class c probabilities, the algorithm uses features from the full image.,. These bounding boxes are weighted by the predicted probabilities. These predictions are processed as

S x S x (B * 5 + C) tensor

If the object's centre is within a grid cell, then detecting the object is the job of the grid cell.

Every grid cell makes a prediction regarding two things:

- The B bounding boxes.
- Those boxes' Confidence scores

The model's confidence score indicates how certain it is that this box contains an object. and also how accurately the box is predicted by YOLO. Formally, it is defined as

$$\boldsymbol{Prob(Obj) * lOU}$$

If that cell has no objects in it, then its confidence score will be zero. If Not, we would like the confidence value to be equivalent to the (I.O.U) between the predicted box and the. ground truth

Every bounding box comprises of five predictions, which are:

- X and Y coordinates, width w, height h
- Confidence

The (x, y) coordinates refer to the middle of the box in relation to the boundaries of the grid cell. The width and height are calculated using the complete image as a reference. Lastly, the confidence prediction characterizes the intersection over the union between the predicted box and the true box. Each grid cell also predicts the C conditional probabilities for a class:

$$\boldsymbol{Prob(Class\ |Object)}$$

These probabilities are written on the grid cell comprising the object Per grid cell, Only one set of class probabilities is predicted.no matter the amount of boxes B.

The class-specific confidence scores for each box are calculated by multiplying conditional class probabilities and individual box confidence predictions during testing.:

$$\boldsymbol{Prob\ (\ Class\ |\ Object\ ) * Prob\ (\ Object\ ) * IOU = Prob\ (\ Class\ ) * IOU}$$

This score is the final confidence score written.

8.2.1 Network Architecture and Training

There are 24 convolutional layers of the YOLO network are followed by two fully linked layers. We will use 1x1 reduction layers followed by 3x3 convolutional layers instead of employing inception modules.

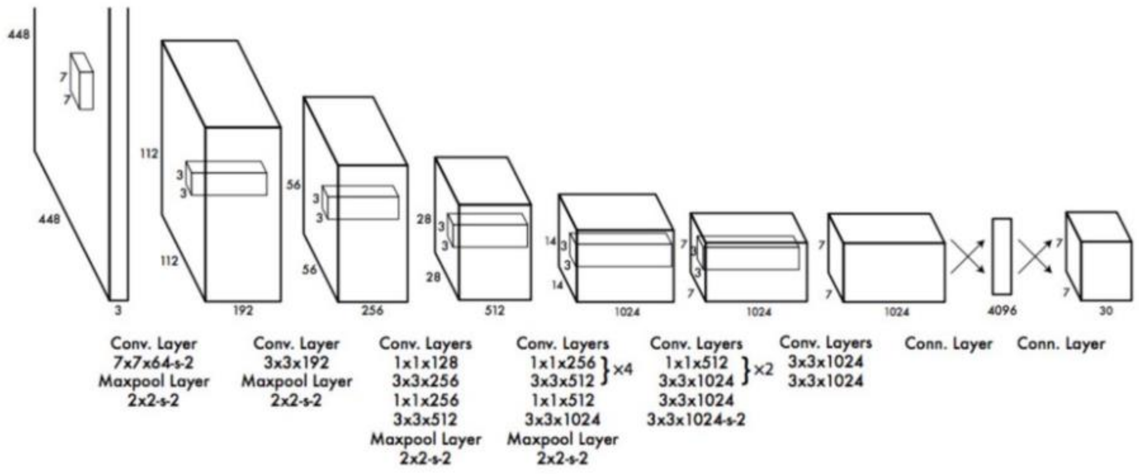


Figure 21 YOLO Architecture [10]

Since we are using fast YOLO in our project a convolutional neural network with fewer layers i.e 9 instead of 24 and fewer filters in those layers is going to be implemented. Except for the size of the network, all The parameters for both training and testing are the same. For Fast YOLO and YOLO.

Our model's output is optimized for the sum-squared error. To reduce model instability caused by weighing localization error equal with the classification error, we have increased the loss obtained via bounding box coordinate predictions and decreased the losing confidence predictions for boxes without any objects. We have employed 2 parameters:

λcoord = 5
λnoobj = 5

Multiple bounding boxes per grid cell can be actively predicted by YOLO but an issue arises that during training just one bounding box predictor is required for every object hence we have assigned one predictor liable for predicting an object, whose prediction contains the highest current IOU with the ground truth. This results in specialization among the bounding box predictors. Every predictor improves its ability to anticipate specific sizes, aspect ratios, or object classifications., therefore enhancing our overall recall.

8.2.2 Linear Regression:

The linear regression allows the building of a model that predicts the value of new data, based on the training data used to train our model.

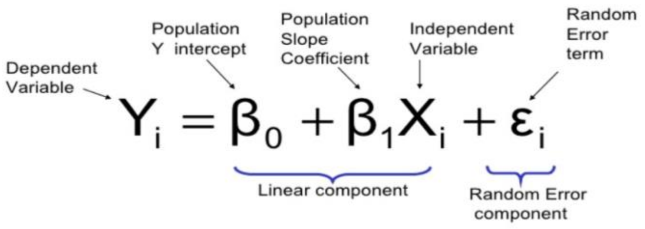


Figure 22 Eq of Linear Regression [11]

The **Y**-variable is the dependent variable or our response which we intend to predict. This enables us to create a model with many features i.e X variables to predict values in Y.

8.3 Data Set Gathering

We gathered images by manually clicking pictures from a 12-megapixel digital camera. Approximately 600 to 1000 pictures were taken for each banknote with different angles and diverse backgrounds to make our dataset more versatile.

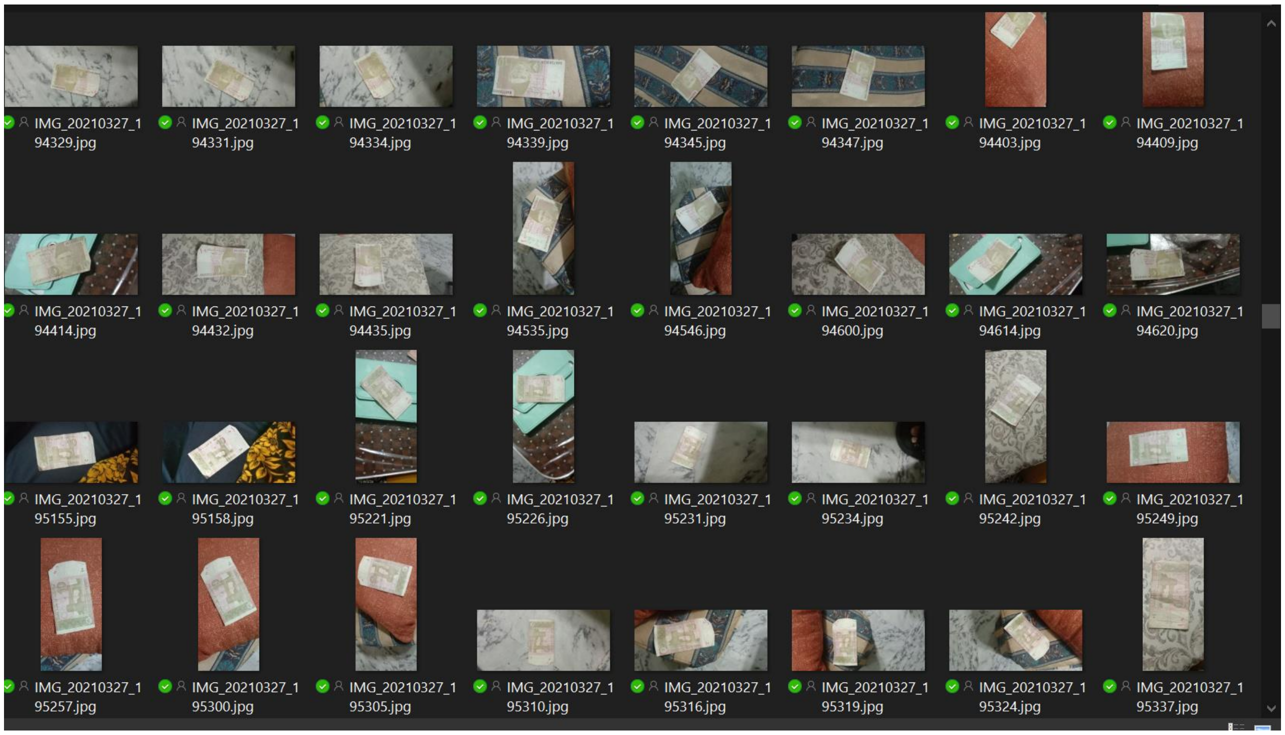


Figure 23 Sample of Data set of Rs.10

8.4 Data Pre-processing

Data pre-processing is that part of our project where we filter out the data which is either blurry or irrelevant. This step must be done manually as each image has to be analyzed and afterwards a decision is made on whether that image must be kept in the data set or not. Some of the few points which were kept in mind during Data Pre-processing were:

- Clear picture with avoiding motion blur
- Currency notes should aptly fit the screen; They should not go beyond the screen.
- Acquire pictures from different angles and distances

8.5 Data Annotation

Afterwards, all the images were annotated by using an open-source software "***Label.Img***". it refers to marking or labeling the data. We have used the parallel rectangles approach to mark our data. When the picture is manually labeled and saved in the respective folder then a text(txt) file of YOLO format appears within the same folder which contains the image having the same name. A file referred to as ***"classes.txt"*** is also saved within that folder. ***"classes.txt"*** contains the list of class names that our YOLO label refers to.

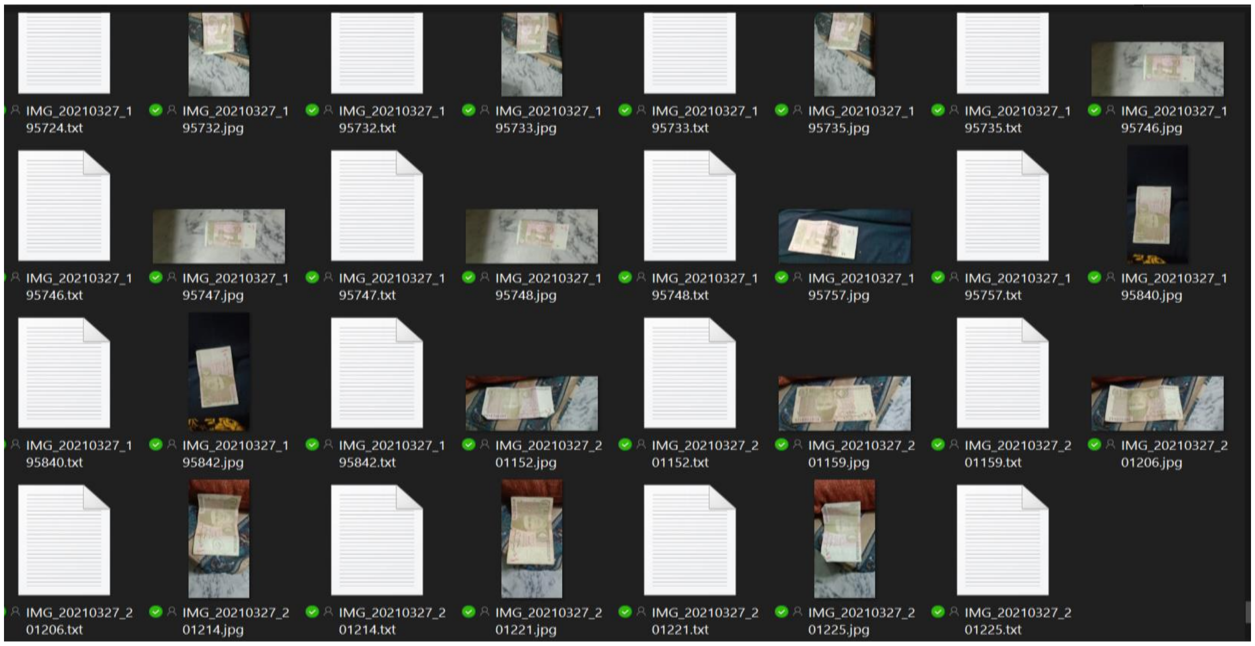


Figure 24 Data set with Classes Txt

8.6 Data Augmentation

The current dataset which we have acquired has 600 to 1000 pictures but these are not sufficient for our model training. In order to achieve a dataset that bears better results a technique known as ***Data Augmentation*** must be performed.

In this step, our dataset is expanded in size using ***morphological operations***. Augmentation is such a strategy that allows the user to notably expand the diversity of training data, without the need of gathering new data. Several data augmentation techniques include cropping, padding, horizontal flipping, etc to train large neural networks. In order to train a deep learning model, we require many images per class i.e. minimum of 600 images per class for decent accuracy.

Afterwards, the dataset is divided into two parts namely training set and test set. In this project, we divided the dataset with a ratio of about 70:30. i.e. 70% of images were placed in the training set while 30% were placed into the test set. The 5 steps done in data augmentation were:

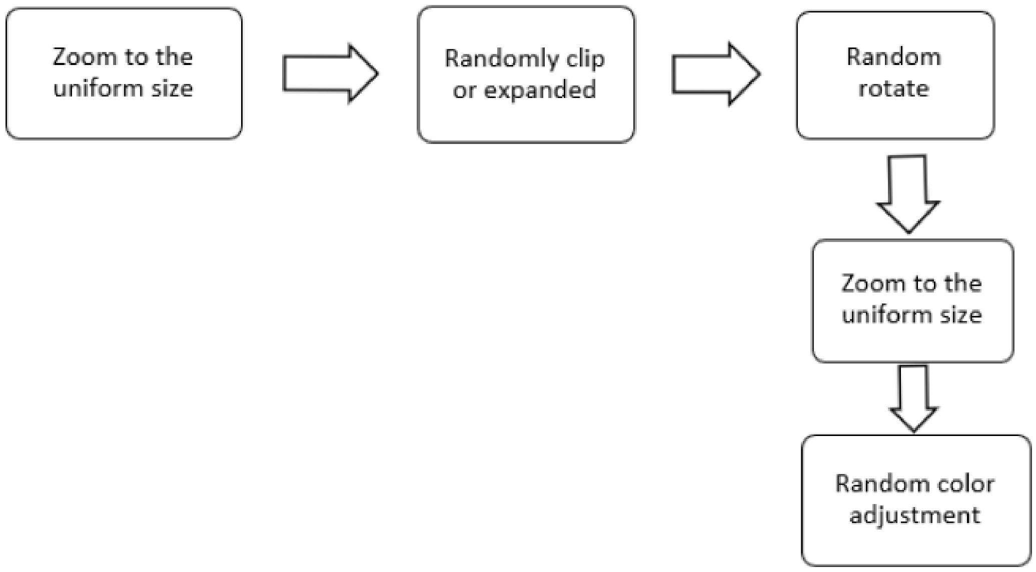


Figure 25 Steps of Augmentation

8.7 Software Prerequisites

The software required for the implementation of the currency recognition includes:

8.7.1 Python 3.9.0

The major software requirement for this project is a python 3.9 installation on your operating system. It can be done by visiting the official website of python.

8.7.2 Anaconda Distribution

The first step of working with image processing on Python is to install the Anaconda distribution from their official website given as https://www.anaconda.com/

8.7.3 Visual Studio code

After installing the anaconda distribution, launch visual studio code, and wait for its installation. Visual Studio Code’s main use is as a source-code editor. It is an essential source-code editor used for various functionalities such as debugging, optimizing code, embedded Git and other version controlling, code refactoring and much more.

8.8 Preparation of Machine Learning Model

8.8.1 TensorFlow

TensorFlow refers to a programming library available in open-source. It is mainly for AI and can be employed across various undertakings but its main focus is on preparing and deduction of neural networks. It may also be referred to as a representative numerical library hooked into differentiable programming and dataflow.

8.8.2 YAML

YAML is a comprehensible information serialization standard that can be utilized related to all programming dialects and is frequently used to compose design records.
To start training a Yolo V5 model we would need two YAML files.
The first YAML is to specify:

- The location of your training
- Location of your validation data
- The number of classes (sorts of items) that you need to distinguish
- The names of the classes they belong to

The second YAML is to indicate the entire model setup. We can change the organization design in this progression on the basis that we need it; yet we will go with the default one.

We are equipped to train our data with the help of these 2 YAML files i.e data.yaml and custom_yolov5s.yaml (given in Appendix D)

To begin training, we must execute the training command with these subsequent options:

- **img**: characterize input picture size
- **batch**: decide group(batch) size
- **epochs**: characterize the quantity of preparing ages. (Note: default=3000)
- **data**: set the way or the path to our yaml record/file
- **cfg**: indicate our model setup
- **weights**: indicate a custom way to loads.
- **name**: result names
- **nosave**: only save the last designated spot
- **cache**: store pictures for quicker preparing

8.9 Using The Model To Make Predictions

This deep learning model thus produced will be used to make predictions on new data. An example of this is shown below.

Figure 27 Rs. 10

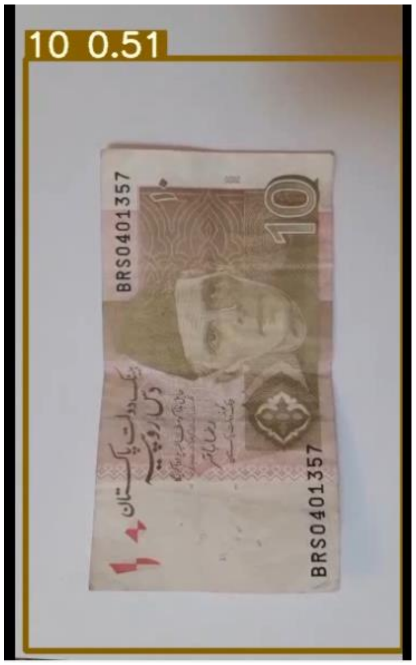


Figure 26 Rs.20

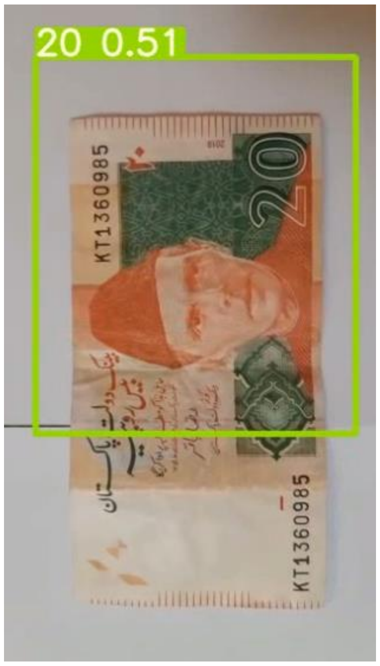


Figure 28 Rs. 50

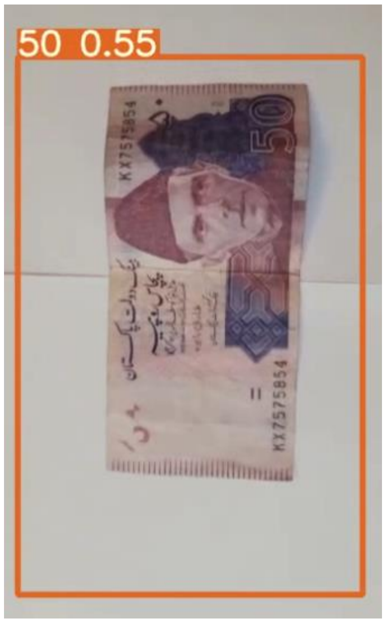


Figure 29 Rs. 500

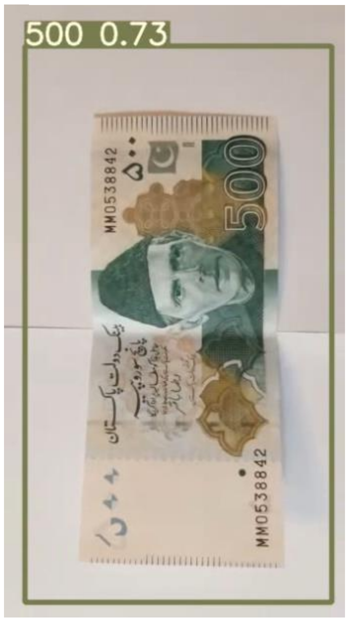


Figure 30 Rs. 5000

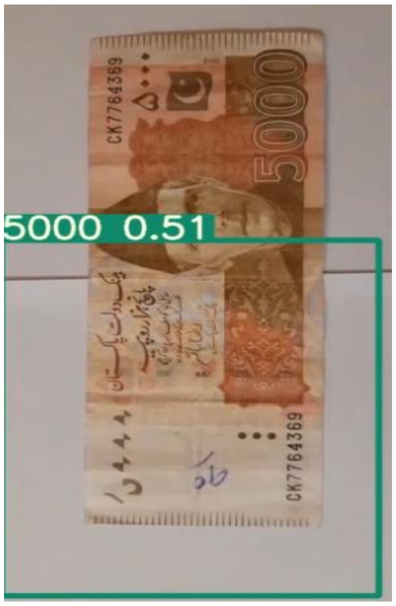


Figure 31 Rs.100

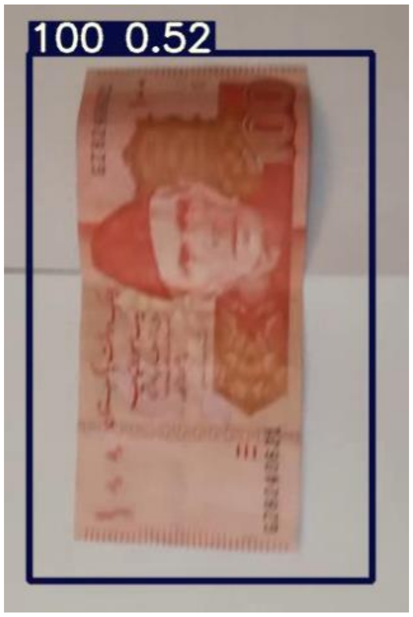
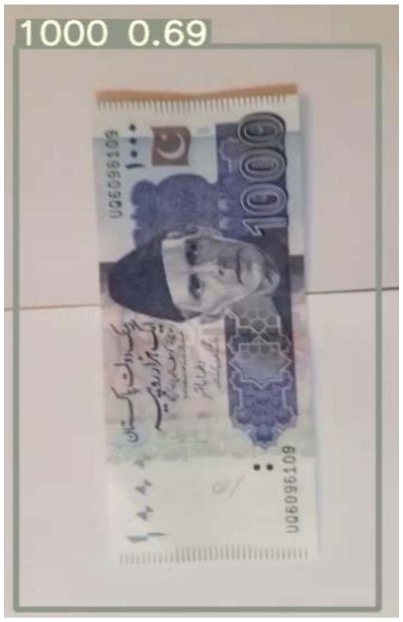


Figure 32 Rs.1000



This feature of our model will be used in our final product while taking frames from a video.

8.10 Accuracy of Deep Learning Algorithm

After the development of the deep learning algorithm, the accuracy on the test set came out to be 80% i.e., about 80% of banknotes from the test set were recognized correctly. Afterwards, the algorithm was also tested manually on new unseen data.

CHAPTER 9:
RECOMMENDATIONS AND CONCLUSIONS

9.1 Recommendations

In future work following features can be accommodated:

- Instead of a portable stick, the device can be made so it can be simply attached to the clothes.
- Further features and functionalities can be added in order to make it more efficient e.g. detection of various things like stairs, doors, windows, car detection etc.
- Instead of the Jetson Nano kit, a more advanced development board i.e., ***Jetson Xavier NX*** and ***Jetson Xavier AGX*** can be used for faster processing.
- GSM system can be mounted in order to send a message about the location of the user to his relatives.
- Voice recognition can be added as another feature that can be done.

9.2 Conclusion

We have been able to make this device portable, wearable, unique, low power, cost effective and easy to use. It is capable to help blind people perform their daily chores more efficiently and independently. We have successfully implemented the concepts of digital image processing, circuit designing, coding/programming in python language and prototyping. It is a multi-functional system that successfully performs the function of obstacle detection in the path of the user using an array of ultrasonic sensors, face recognition, determines the number of people in front of the user by face detection scheme, identifies paper currency using a camera and it also monitors user's health conditions using pulse and temperature sensors. The outputs of the respective functions are given to the user through an audio signal via headphones/speakers.

Therefore, through this project we have integrated our theoretical knowledge with practicality and gained further insight and refined our skills in all the fields mentioned above. The major part of this project consists of concepts involving artificial intelligence, deep learning, machine learning and sensors.

APPENDICES

APPENDIX A

APPROVED SYNOPSIS

VISION-Z

Extended Title: Artificial Intelligence Based Aiding Device for Visually Impaired People
Brief Description of The Project / Thesis with Salient Specifications: A multifunctional aiding device for visually impaired people which detects obstacles, recognize various objects and monitors healthcare
Scope of Work: Design, implementation, and development of a visual aiding system
Academic Objectives: This project will provide hands-on experience in the following courses: • Deep Learning Algorithms • Image Processing • Embedded Systems • Programming Techniques • Structure Designing
Application / End Goal Objectives: AI based standalone device aiding daily functionality of blind people
Previous Work Done on The Subject: • Vision Pi • Laser Canes, Braille printers, Infrared based obstacle detection and Text readers
Material Resources Required: We will require Jetson Nano in concurrence with Arduino interfaced with different sensors, incorporation of deep learning-based object detection and recognition. Programming Languages I-e Python, Arduino
No of Students Required: 4 **Group Members:** NC SIMRA KAUSAR RAJA, NC WAJIHA JAWWAD, ASC PERVAZ ALAM, NC ZARGHAM BAIG
Special Skills Required: • Circuit design and analysis skills • Programming skills • Deep learning, Computer Vision • Technical documentation writing skills

Table 2 Approved Synopsis

APPENDIX B

PROJECT TIMELINE

Month	**Work Done**
Sep	Project Synopsis and Approval
Oct	Literature Review
Nov	Nano Installation and Setup, Installation and Setup of OpenCv
Dec	Camera Integration with Nano, Integrating various sensors with Arduino.eg ultra-sonic sensors, temperature sensor etc.
Jan	Gathering Data Set, managing the data set and training on yolo model. Integration of Arduino with Nano.
Feb	Currency Recognition, Facial Recognition and testing all models
Mar	Image captioning, Testing Sensors and Hardware interfacing
Apr	Arduino output, Audio output, Final assembly and Testing
May	Thesis write up and finalization

Table 3 Project Timeline

APPENDIX C
COST BRAKDOWN

Name of Equipment	**Cost-estimate**
Jetson Nano	25000PKR
MicroSD Card	1350 PKR
Arduino	1250 PKR
Pulse Sensor	1350 PKR
Temperature Sensor	550 PKR
Ultrasonic Sensor	350 PKR
Keypad	150 PKR
Camera	1000 PKR
Headphones	750 PKR
Blind cane	800 PKR
Power Bank	3000 PKR
Total	35,550 PKR

Table 4 Cost Breakdown

APPENDIX D:
CODES

Code for Facial Recognition

```
import face_recognition
from PIL import Image, ImageDraw
import numpy as np
import cv2

image_of_pervaz = face_recognition.load_image_file('known/Pervaz.jpg')
pervaz_face_encoding = face_recognition.face_encodings(image_of_pervaz)[0]

image_of_simra = face_recognition.load_image_file('known/Simra.jpg')
simra_face_encoding = face_recognition.face_encodings(image_of_simra)[0]

image_of_wajiha = face_recognition.load_image_file('known/Wajiha.jpg')
wajiha_face_encoding = face_recognition.face_encodings(image_of_wajiha)[0]

image_of_zargham = face_recognition.load_image_file('known/Zargham.jpg')
zargham_face_encoding = face_recognition.face_encodings(image_of_zargham)[0]
```

```
#  Create arrays of encodings and names
known_face_encodings = [
  pervaz_face_encoding,
  simra_face_encoding,
  wajiha_face_encoding,
  zargham_face_encoding
]

known_face_names = [
  "pervaz",
  "Simra",
  "Wajiha",
  "zargham"
]

# Load test image to find faces in

test_image = face_recognition.load_image_file('./unknown/a (30).jpg')

# Find faces in test image
face_locations = face_recognition.face_locations(test_image)
face_encodings = face_recognition.face_encodings(test_image, face_locations)

# Convert to PIL format
pil_image = Image.fromarray(test_image)

# Create a ImageDraw instance
draw = ImageDraw.Draw(pil_image)

# Loop through faces in test image
for(top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
  matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
  name = "Unknown Person"

  # If match
  if True in matches:
    first_match_index = matches.index(True)
    name = known_face_names[first_match_index]

  # Draw box
  draw.rectangle(((left, top), (right, bottom)), outline=(255,255,0))

  # Draw label
  text_width, text_height = draw.textsize(name)
  draw.rectangle(((left,bottom - text_height - 10), (right, bottom)), fill=(255,255,0),
 outline=(255,255,0))
  draw.text((left + 6, bottom - text_height - 5), name, fill=(0,0,0))
```

```
name = "Unknown Person"

  # If match
  if True in matches:
    first_match_index = matches.index(True)
    name = known_face_names[first_match_index]

  # Draw box
  draw.rectangle(((left, top), (right, bottom)), outline=(255,255,0))

  # Draw label
  text_width, text_height = draw.textsize(name)
  draw.rectangle(((left,bottom - text_height - 10), (right, bottom)), fill=(255,255,0),
outline=(255,255,0))
  draw.text((left + 6, bottom - text_height - 5), name, fill=(0,0,0))
```

```
del draw

# Display image
pil_image.show()
# image = np.array(pil_image)
# cv2.imshow("Results", image)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
# Save image
pil_image.save('identify.jpg')
```

Code for Image Captioning

```
import tensorflow as tf

# You'll generate plots of attention in order to see which parts of an image
# our model focuses on during captioning
import matplotlib.pyplot as plt

import collections
import random
import numpy as np
import os
import time
import json
from PIL import Image
import pyttsx3
import cv2
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth=True
```

```
sess = tf.compat.v1.Session(config=config)

def load_image(frame):
    img = tf.image.resize(frame, (299, 299))
    img = tf.keras.applications.inception_v3.preprocess_input(img)
    return img

image_model = tf.keras.applications.InceptionV3(include_top=False,
                                                weights='imagenet')
new_input = image_model.input
hidden_layer = image_model.layers[-1].output

image_features_extract_model = tf.keras.Model(new_input, hidden_layer)

with open('tokenizer.json') as f:
    data = json.load(f)
    tokenizer = tf.keras.preprocessing.text.tokenizer_from_json(data)
tokenizer.word_index['<pad>'] = 0
tokenizer.index_word[0] = '<pad>'

top_k = 30000
BATCH_SIZE = 128
BUFFER_SIZE = 1000
embedding_dim = 256
units = 512
vocab_size = top_k + 1
# num_steps = len(img_name_train) // BATCH_SIZE
# Shape of the vector extracted from InceptionV3 is (64, 2048)
# These two variables represent that vector shape
features_shape = 2048
attention_features_shape = 64
```

```
class BahdanauAttention(tf.keras.Model):
  def __init__(self, units):
    super(BahdanauAttention, self).__init__()
    self.W1 = tf.keras.layers.Dense(units)
    self.W2 = tf.keras.layers.Dense(units)
    self.V = tf.keras.layers.Dense(1)

  def call(self, features, hidden):
    # features(CNN_encoder output) shape == (batch_size, 64, embedding_dim)

    # hidden shape == (batch_size, hidden_size)
    # hidden_with_time_axis shape == (batch_size, 1, hidden_size)
    hidden_with_time_axis = tf.expand_dims(hidden, 1)

    # attention_hidden_layer shape == (batch_size, 64, units)
    attention_hidden_layer = (tf.nn.tanh(self.W1(features) +
                                         self.W2(hidden_with_time_axis)))
    # score shape == (batch_size, 64, 1)
    # This gives you an unnormalized score for each image feature.
    score = self.V(attention_hidden_layer)

    # attention_weights shape == (batch_size, 64, 1)
    attention_weights = tf.nn.softmax(score, axis=1)

    # context_vector shape after sum == (batch_size, hidden_size)
    context_vector = attention_weights * features
    context_vector = tf.reduce_sum(context_vector, axis=1)

    return context_vector, attention_weights


# In[ ]:
```

```
class CNN_Encoder(tf.keras.Model):
    # Since you have already extracted the features and dumped it
    # This encoder passes those features through a Fully connected layer
    def __init__(self, embedding_dim):
        super(CNN_Encoder, self).__init__()
        # shape after fc == (batch_size, 64, embedding_dim)
        self.fc = tf.keras.layers.Dense(embedding_dim)

    def call(self, x):
        x = self.fc(x)
        x = tf.nn.relu(x)
        return x

# In[ ]:

class RNN_Decoder(tf.keras.Model):
  def __init__(self, embedding_dim, units, vocab_size):
def __init__(self, embedding_dim, units, vocab_size):
super(RNN_Decoder, self).__init__()
self.units = units

self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
self.gru = tf.keras.layers.GRU(self.units,
                               return_sequences=True,
                               return_state=True,
                               recurrent_initializer='glorot_uniform')
self.fc1 = tf.keras.layers.Dense(self.units)
self.fc2 = tf.keras.layers.Dense(vocab_size)

self.attention = BahdanauAttention(self.units)

def call(self, x, features, hidden):
# defining attention as a separate model
```

```
context_vector, attention_weights = self.attention(features, hidden)

# x shape after passing through embedding == (batch_size, 1, embedding_dim)
x = self.embedding(x)

# x shape after concatenation == (batch_size, 1, embedding_dim + hidden_size)
x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

# passing the concatenated vector to the GRU
output, state = self.gru(x)

# shape == (batch_size, max_length, hidden_size)
x = self.fc1(output)

# x shape == (batch_size * max_length, hidden_size)
x = tf.reshape(x, (-1, x.shape[2]))
    # output shape == (batch_size * max_length, vocab)
    x = self.fc2(x)

    return x, state, attention_weights

   def reset_state(self, batch_size):
    return tf.zeros((batch_size, self.units))

  # In[ ]:

  encoder = CNN_Encoder(embedding_dim)
  decoder = RNN_Decoder(embedding_dim, units, vocab_size)

  # In[ ]:

  optimizer = tf.keras.optimizers.Adam()
```

```
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True, reduction='none')



def loss_function(real, pred):
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  loss_ = loss_object(real, pred)

  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask

  return tf.reduce_mean(loss_)



# ## Checkpoint

# In[ ]:

checkpoint_path = "./checkpoints/train"
ckpt = tf.train.Checkpoint(encoder=encoder,
                           decoder=decoder,
                           optimizer=optimizer)
ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path, max_to_keep=5)
ckpt.restore(ckpt_manager.latest_checkpoint)

max_length = 52

def evaluate(image):
    attention_plot = np.zeros((max_length, attention_features_shape))
  hidden = decoder.reset_state(batch_size=1)

  temp_input = tf.expand_dims(load_image(image), 0)
  img_tensor_val = image_features_extract_model(temp_input)
  img_tensor_val = tf.reshape(img_tensor_val, (img_tensor_val.shape[0],
                                               -1,
                                               img_tensor_val.shape[3]))

  features = encoder(img_tensor_val)

  dec_input = tf.expand_dims([tokenizer.word_index['<start>']], 0)
  result = []

  for i in range(max_length):
      predictions, hidden, attention_weights = decoder(dec_input,
                                                       features,
                                                       hidden)
      attention_plot[i] = tf.reshape(attention_weights, (-1, )).numpy()

      predicted_id = tf.random.categorical(predictions, 1)[0][0].numpy()
      result.append(tokenizer.index_word[predicted_id])

      if tokenizer.index_word[predicted_id] == '<end>':
          return result, attention_plot

      dec_input = tf.expand_dims([predicted_id], 0)

  attention_plot = attention_plot[:len(result), :]
  return result, attention_plot

video_name = 'img.mp4'
```

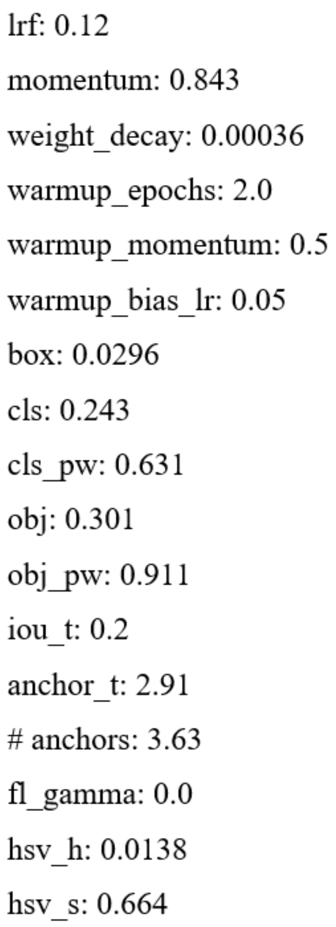
```
cap = cv2.VideoCapture(video_name)
engine  = pyttsx3.init()
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result, _ = evaluate(img)
    result = ' '.join(result)
    print(result)



    # Display the resulting frame
    cv2.imshow('frame',frame)
    engine.say(result)
    engine.runAndWait()
      cv2.imshow('frame',frame)
      engine.say(result)
      engine.runAndWait()
if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

YAML file for YOLO for Currency Recognition

```
# Hyperparameters for COCO training from scratch
# python train.py --batch 40 --cfg yolov5m.yaml --weights '' --data coco.yaml --img 640 --epochs
300
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.2  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.5  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 60.0  # image rotation (+/- deg)
translate: 0.3  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.2  # image shear (+/- deg)
perspective: 0.001  # image perspective (+/- fraction), range 0-0.001
flipud: 0.5  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
```

YAML File for Fine Tuning

```
# Hyperparameters for VOC finetuning
# python train.py --batch 64 --weights yolov5m.pt --data voc.yaml --img 512 --epochs 50
# See tutorials for hyperparameter evolution https://github.com/ultralytics/yolov5#tutorials


# Hyperparameter Evolution Results
# Generations: 306
#                    P         R     mAP.5 mAP.5:.95       box       obj       cls
# Metrics:        0.6     0.936     0.896     0.684    0.0115   0.00805   0.00146

lr0: 0.0032
```



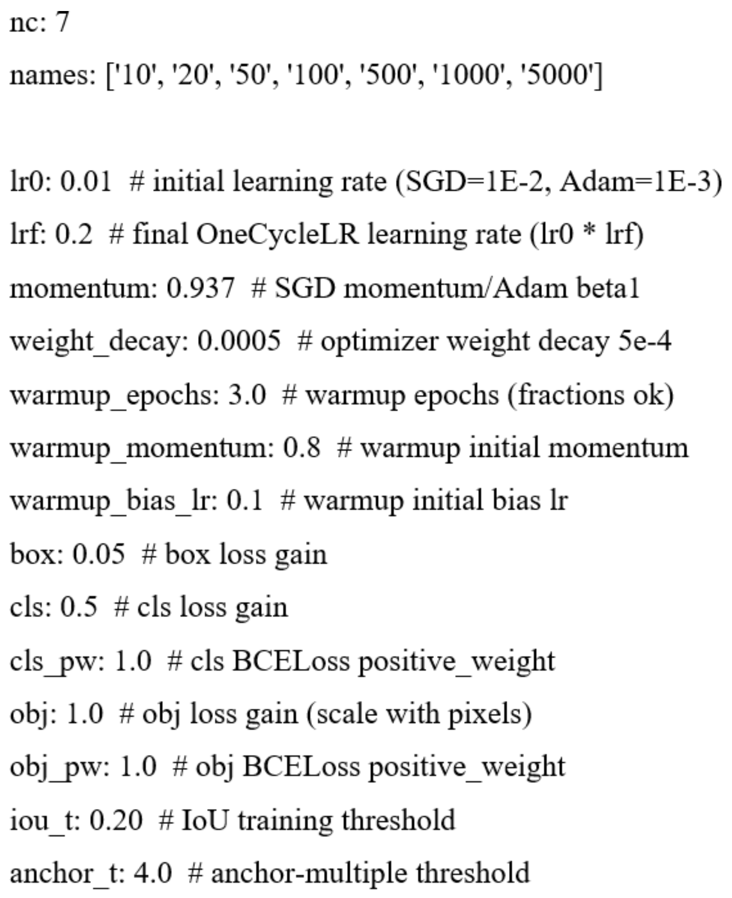
```
lrf: 0.12
momentum: 0.843
weight_decay: 0.00036
warmup_epochs: 2.0
warmup_momentum: 0.5
warmup_bias_lr: 0.05
box: 0.0296
cls: 0.243
cls_pw: 0.631
obj: 0.301
obj_pw: 0.911
iou_t: 0.2
anchor_t: 2.91
# anchors: 3.63
fl_gamma: 0.0
hsv_h: 0.0138
hsv_s: 0.664
```

```
hsv_v: 0.464
degrees: 0.373
```

```
translate: 0.245
scale: 0.898
shear: 0.602
perspective: 0.0
flipud: 0.00856
fliplr: 0.5
mosaic: 1.0
mixup: 0.243
```

YAML File for Data

```
train: yolodata/train
val: yolodata/val
```



```
nc: 7
names: ['10', '20', '50', '100', '500', '1000', '5000']

lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.2  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
```

```
# anchors: 3  # anchors per output layer (0 to ignore)
```

```
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.15  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 60.0  # image rotation (+/- deg)
translate: 0.4  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.3  # image shear (+/- deg)
perspective: 0.1  # image perspective (+/- fraction), range 0-0.001
flipud: 0.5  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.5  # image mixup (probability)
```

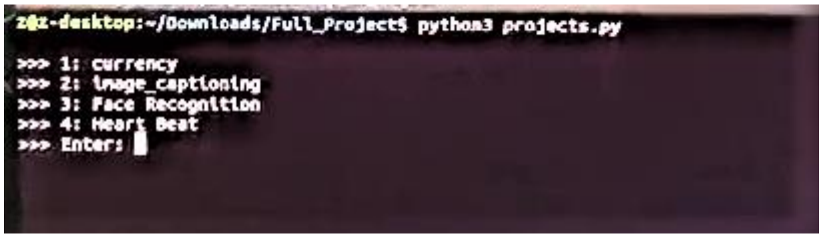
Code for Object Detection

```
#include <Servo.h>.
// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 10;
const int echoPin = 11;
// Variables for the duration and the distance
long duration;
int distance;
Servo myServo; // Creates a servo object for controlling the servo motor
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600);
  myServo.attach(12); // Defines on which pin is the servo motor attached
}
void loop() {
```

```
// rotates the servo motor from 15 to 165 degrees
  for(int i=15;i<=165;i++){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();// Calls a function for calculating the distance measured by the
Ultrasonic sensor for each degree

  Serial.print(i); // Sends the current degree into the Serial Port
  Serial.print(","); // Sends addition character right next to the previous value needed later in the
Processing IDE for indexing
  Serial.print(distance); // Sends the distance value into the Serial Port
  Serial.print("."); // Sends addition character right next to the previous value needed later in the
Processing IDE for indexing
  }
  // Repeats the previous lines from 165 to 15 degrees
  for(int i=165;i>15;i--){
  myServo.write(i);
 delay(30);
   distance = calculateDistance();
   Serial.print(i);
   Serial.print(",");
   Serial.print(distance);
   Serial.print(".");
   }
 }
 // Function for calculating the distance measured by the Ultrasonic sensor
 int calculateDistance(){

   digitalWrite(trigPin, LOW);
   delayMicroseconds(2);
   // Sets the trigPin on HIGH state for 10 micro seconds
   digitalWrite(trigPin, HIGH);
```

```
delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in
microseconds
  distance= duration*0.034/2;
  return distance;
}
```

Code for Jetson Nano



```
z@z-desktop:~/Downloads/Full_Project$ python3 projects.py

>>> 1: currency
>>> 2: image_captioning
>>> 3: Face Recognition
>>> 4: Heart Beat
>>> Enter:
```

Figure 33 Backend Display

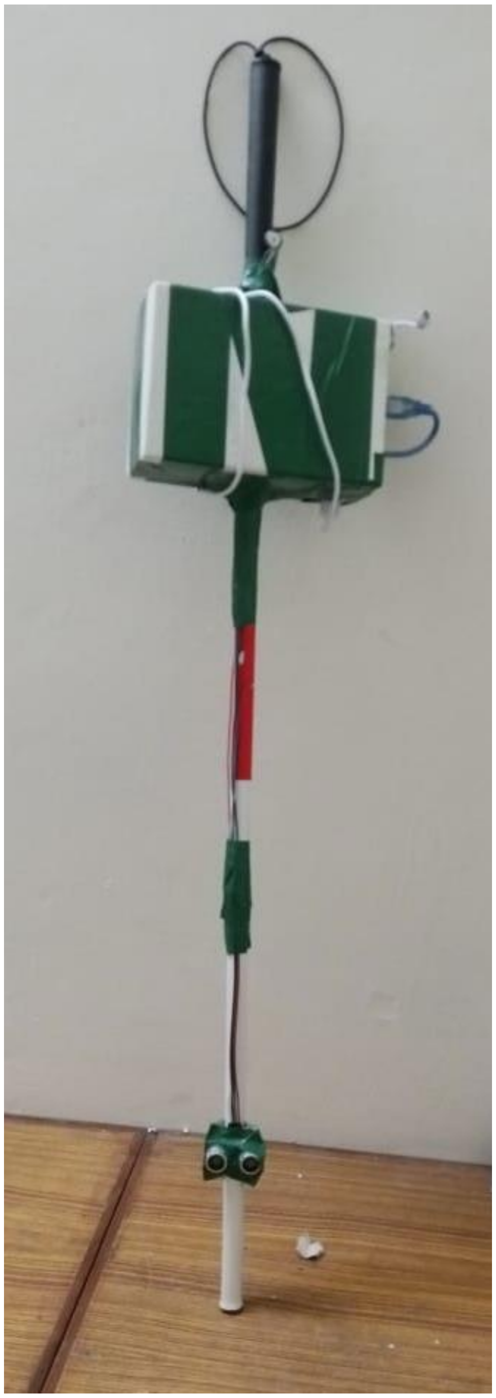
```
import subprocess
import os
import pyautogui
import pyttsx3

engine = pyttsx3.init()
project = input("\n>>> 1:  currency \n>>> 2:  image_captioning \n>>> 31: Face Recognition
(video) \n>>> 32: Face Recognition (webcam)  \n>>> 4:  Sensors \n>>> Enter: ")

if project == "1":
    os.chdir("yolo/")
    subprocess.run(["python3", "detect.py", "--source", "1.mp4", "--weights", "best.pt"])
```

```
if project == "2":
    os.chdir("ic_mobilenet/")
    subprocess.run(["python3", "evaluate_caption.py"])

if project == "31":
    os.chdir("face_rec/")
    subprocess.run(["python3", "video.py"])

if project == "32":
    os.chdir("face_rec/")
    subprocess.run(["python3", "webcam.py"])

if project == "4":
    print("PulseSensor")
    print(">>> S with A0 of arduino \n>>> vcc+ with 3.3V of arduino \n>>> vcc- with ground of
arduino \n")
    print("Temperature sensor")
    print(">>>Temperature sensor DS18B20 \n>>> D with 2 of arduino \n>>> vcc+ with 3.3V of
arduino \n>>> vcc- with ground of arduino \n")
    print("UltraSonic")
    print(">>>UltraSonic HC-SR04 \n>>> ECHO with ~10 of arduino \n>>> trg with ~9 of
aurdino \n>>> vcc+ with 3.3V of arduino \n>>> vcc- with ground of arduino \n")

    os.chdir("Sensors/")
    engine.say(subprocess.run(["arduino", "--upload", "sensors.ino", "--port", "/dev/ttyUSB0"]))
    engine.runAndWait()
    subprocess.run(["minicom", "-D", "/dev/ttyUSB0", "-b", "9600"])
    pyautogui.press(['ctrl','a','x','enter'])
    print("\n")

if project == "0":
```

```
pyautogui.hotkey('ctrl', 'z')
pyautogui.press(['ctrl','a','x','enter'])
print("\n")
```

APPENDIX E :

FINAL DELIVERABLE



BIBLOGRAPHY

[1]. Cardin, S., Thalmann, D., & Vexo, F. (2007). A wearable system for mobility improvement of visually impaired people. The Visual Computer, 23(2), 109-118.

[2]. MacNamara, S., & Lacey, G. (2000, April). A smart walker for the frail visually impaired. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065) (Vol. 2, pp. 1354-1359). IEEE.

[3]. Kim, C. G., & Song, B. S. (2007, April). Design of a wearable walking-guide system for the blind. In Proceedings of the 1st international convention on Rehabilitation engineering & assistive technology: in conjunction with 1st Tan Tock Seng Hospital Neurorehabilitation Meeting (pp. 118-122)

[4]. Zöllner, M., Huber, S., Jetter, H. C., & Reiterer, H. (2011, September). NAVI–a proof-of-concept of a mobile navigational aid for visually impaired based on the Microsoft Kinect. In IFIP Conference on Human-Computer Interaction (pp. 584-587). Springer, Berlin, Heidelberg.

[5]. Innet, S., & Ritnoom, N. (2009, February). An application of infrared sensors for electronic white stick. In 2008 International Symposium on Intelligent Signal Processing and Communications Systems (pp. 1-4). IEEE.

[6]. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[7]. Josh, H., Yong, B., & Kleeman, L. (2011, December). A real-time fpga-based vision system for a bionic eye. In Proceedings of Australasian Conference on Robotics and Automation (ACRA).

[8]. Batavia, P. H., & Singh, S. (2001, May). Obstacle detection using adaptive color segmentation and color stereo homography. In Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164) (Vol. 1, pp. 705-710). IEEE.

[9]. Jain, P., & Awasthy, M. (2014). Automatic Obstacle Detection using Image Segmentation. International Journal of Emerging Technology and Advanced Engineering, 4(3).

79

REFERENCES

[10]. YOLO — You only look once, real time object detection explained (2017). Available at: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006 (Accessed: 7 June 2021).

[11]. Linear Regression in Python (2017). Available at: https://towardsdatascience.com/linear-regression-in-python-9a1f5f000606 (Accessed: 7 June 2021).

[12]. Brownlee, J. (2019) A Gentle Introduction to Deep Learning for Face Recognition, Machine Learning Mastery. Available at: https://machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/ (Accessed: 7 June 2021).

[13].Introduction to how CNNs Work (2019). Available at: https://medium.datadriveninvestor.com/introduction-to-how-cnns-work-77e0e4cde99b (Accessed: 7 June 2021).

[14]. Bahdanau Attention — Dive into Deep Learning 0.16.5 documentation (2021). Available at: https://d2l.ai/chapter_attention-mechanisms/bahdanau-attention.html (Accessed: 7 June 2021).

[15]. Recurrent Neural Network (RNN) Tutorial for Beginners (2021). Available at: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn (Accessed: 7 June 2021).

thesis vision Z

ORIGINALITY REPORT

12%	9%	4%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	4%
2	Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016 Publication	1%
3	www.tensorflow.org Internet Source	1%
4	blog.csdn.net Internet Source	<1%
5	gilberttanner.com Internet Source	<1%
6	www.bbc.co.uk Internet Source	<1%
7	Submitted to British University in Egypt Student Paper	<1%

8	Ashwini B Yadav, Leena Bindal, V. U Namhakumar, K Namitha, H Harsha. "Design and development of smart assistive device for visually impaired people", 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2016 Publication	<1 %
9	www.elprocus.com Internet Source	<1 %
10	www.maxbotix.com Internet Source	<1 %
11	Submitted to University of Southampton Student Paper	<1 %
12	Submitted to La Trobe University Student Paper	<1 %
13	tensorflow.google.cn Internet Source	<1 %
14	Submitted to Nanyang Technological University Student Paper	<1 %
15	Submitted to University of Sydney Student Paper	<1 %
16	epdf.tips Internet Source	<1 %

17	speakerdeck.com Internet Source	<1%
18	Submitted to Ganpat University Student Paper	<1%
19	Bing Yu, Yang Li, Chao Ping Chen, Nizamuddin Maitlo, Jiaqi Chen, Wenbo Zhang, Lantian Mi. "30-4: Semantic Simultaneous Localization and Mapping for Augmented Reality", SID Symposium Digest of Technical Papers, 2018 Publication	<1%
20	Alexy Bhowmick, Saurabh Prakash, Rukmani Bhagat, Vijay Prasad, Shyamanta M. Hazarika. "Chapter 16 IntelliNavi: Navigation for Blind Based on Kinect and Machine Learning", Springer Science and Business Media LLC, 2014 Publication	<1%
21	yingrenn.blogspot.com Internet Source	<1%
22	Submitted to Institute of Research & Postgraduate Studies, Universiti Kuala Lumpur Student Paper	<1%
23	Suet-Peng Yong, Yoon-Chow Yeong. "Human Object Detection in Forest with Deep Learning based on Drone's Vision", 2018 4th	<1%

	International Conference on Computer and Information Sciences (ICCOINS), 2018 Publication	
24	Submitted to University of Central Lancashire Student Paper	<1%
25	pythonprogramming.net Internet Source	<1%
26	Submitted to Akdeniz University Student Paper	<1%
27	Submitted to Imperial College of Science, Technology and Medicine Student Paper	<1%
28	Submitted to Melbourne Institute of Technology Student Paper	<1%
29	Submitted to Softwarica College Of IT & E-Commerce Student Paper	<1%
30	Submitted to University of Northumbria at Newcastle Student Paper	<1%
31	Submitted to University of Sheffield Student Paper	<1%
32	www.dabapps.com Internet Source	<1%

33	C. Stamate, G.D. Magoulas, S. Kueppers, E. Nomikou et al. "The cloudUPDRS app: A medical device for the clinical assessment of Parkinson's Disease", Pervasive and Mobile Computing, 2018 Publication	<1 %
34	Submitted to Far Eastern University Student Paper	<1 %
35	Rohit Ghosh, Omar Smadi. "Automated Detection and Classification of Pavement Distresses using 3D Pavement Surface Images and Deep Learning", Transportation Research Record: Journal of the Transportation Research Board, 2021 Publication	<1 %
36	Tingling Xu, Bingsong Wang, Hua Liu, Haidong Wang et al. "Prevalence and causes of vision loss in China from 1990 to 2019: findings from the Global Burden of Disease Study 2019", The Lancet Public Health, 2020 Publication	<1 %
37	github.com Internet Source	<1 %
38	Yuchuan Du, Ning Pan, Zihao Xu, Fuwen Deng, Yu Shen, Hua Kang. "Pavement distress detection and classification based on YOLO	<1 %

network", International Journal of Pavement Engineering, 2020
Publication

39	dspace.auk.edu.kw Internet Source	<1%
40	eprints.nottingham.ac.uk Internet Source	<1%
41	medium.com Internet Source	<1%
42	developer.nvidia.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		