CAPSTONE PROJECT PHASE-II REPORT

on

CROWD DENSITY ESTIMATION USING THERMAL CAMERA

Submitted By

Abhishek Shelke (1032180694)

Atharva Joshi (1032180695)

Shubham Sharma (1032181026)

Under the Guidance of

Dr. Vinaya Gohokar

School of Electronics & Communication Engineering

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY, PUNE.
[2021-2022]

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

DECLARATION

I/We the undersigned, declare that the work carried under Capstone Project Phase-II entitled **"_Crowd Density Estimation Using Thermal Camera_"** represents my/our idea in my/our own words. I/We have adequately cited and referenced the original sources where other ideas or words have been included. I/We also declare that I/We have adhered to all principles of academic honesty and integrity and have not misprinted or fabricated or falsified any ideas/data/fact/source in my/our submission. I/We understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or whom proper permission has not been taken when needed.
Date: 28.0522
Place: Pune

PRN Number	Name of student	Signature with date
1032180694	Abhishek Shelke	
1032180695	Atharva Joshi	
1032181026	Shubham Sharma	

Project Guide	Batch Coordinator	Head of School
Dr. Vinaya Gohokar	Prof . Manisha Ingle	Dr. Vinaya Gohokar

Date: 28.05.22
Place: Pune

Table of Contents

ABSTRACT

The Rapid spread of Sars Cov-2 from the past two years has made human civilization consider their Disease control methods. Masks, Social Distancing and even vaccines are solutions to prevent the spread of these air-borne viruses. But at the first stage of defense comes the thermal screening of people that takes place when they enter any public area. This screening is done to prevent any person from entering the area with an elevated body temperature, which is indeed a symptom of this disease.

Also, the gathering of people in an area defeats the sole purpose of social distancing rule. In this project, we have explored the implications of using thermal imaging techniques to get crowd analytics in public areas to make the outdoors safe for everyone. We are utilizing cutting-edge object detection algorithms to get an estimation of the number of people in an area regardless of the time of the day. In this project we have deployed object detection algorithm on Edge computing devices to get real time inferencing and detection.

ABBREVIATIONS

YOLO	You Only Look Once
CNN	Convolution Neural Network
GPU	Graphical Processing Unit
HOG	Histogram of Oriented Gradients
SSD	Single Shot Detection
IR	Infrared
SIFT	Scale-invariant feature transform
LBP	Local binary patterns
GLCM	Gray-Level Co-Occurrence Matrix

CHAPTER 1
INTRODUCTION

1.1 Introduction

The constant acceleration of urbanization and the rapid spread of communicable diseases, it is high time we monitor the density of people present in each area. Recently, video analysis has been finding its application for problems of crowd behavior monitoring and understanding. Focus of this work is to find a solution to accurately estimate the amount of people present in each facility.

Having means to acquire such information would allow us to better understand how various offices, meeting, common and workspaces are being used across the campus. Consequently, it can help to optimize crowd flow while making premises more attractive for both campus employees and visitors, and drive down maintenance costs.

Our project works on this problem statement to get a feasible and an universal solution. Crowd Density Estimation is the ratio of number of people in an area to the numerical value of the area itself. If the number of people exceed in the given area it can be said that the area is densely crowded. We have considered detecting Free-Flowing crowds for our project. Free flowing crowds can be seen at metro stations , malls etc.

The Video input that has to be passed to the object detection model is to be taken by a thermal camera; which has been chosen as for its exquisite performance in abundance and deprivation of light.

To undertake the arduous task of detection and estimation we have implemented an object detection algorithm which efficiently detects Humans (objects) and further gives out the required information. This topic is discussed in detail under Algorithm implementation of our report. The object detection model was also custom trained on Bespoke dataset collected by the team using the Thermal Camera Megger TC3231. This is an universal dataset which can be utilized to train any model for human detection and work flawlessly to achieve great accuracy.

The model was further implemented on Edge computing device for fast inference and the data generated by the model is displayed on Local Host Server.

The Report also discusses our system implementation and the components used; Jetson Nano and MLX90640 thermal camera.

CHAPTER 2
REVIEW OF LITERATURE

2.1 LITERATURE REVIEW

The various approaches for crowd counting are mainly divided into four categories: detection-based, regression-based, density estimation, and more recently CNN-based density estimation approaches. We focus on the CNN-based density estimation and crowd counting[1], [2] model in this survey. For the sake of completeness, it is necessary to review some other related works in this subsection.

Early works on crowd counting use detection- based approaches. These approaches usually apply a person or head detector via a sliding window on an image. Recently many extraordinary object detectors such as R-CNN, YOLO[3], and SSD have been presented, which may perform dramatic detection accuracy in the sparse scenes. However, they will present unsatisfactory results when encountered in the situation of occlusion and background clutter in extremely dense crowds.

To reduce the above problems, some works introduce regression-based methods which directly learn the mapping from an image patch to the count. They usually first extract global features (texture, gradient, edge features), or local features (SIFT[4], LBP[5], HOG[6], GLCM[7]). Then some regression techniques such as linear regression[8] and Gaussian mixture regression[9] are used to learn a mapping function to the crowd counting.

There are various techniques in literature which studied the characteristics and counting the size of the crowd in visible band and thermal band.

Literature research have proved that thermal bands can outperform many of the visible bands' problems, the fact that attracted researchers to use thermal bands in the field of crowd counting and control[10]

Through this survey, we expect to make reasonable inference and prediction for the future development of crowd density estimation, and meanwhile, it can also provide feasible solutions and make guidance for the problem of object counting in other domains.[11]

2.2 AIM AND OBJECTIVES OF PROJECT

The main objective of this work is to investigate possible deep learning-based solutions for an accurate indoor crowd estimation using thermal cameras. A range of potentially suitable approaches is tested in practice and documented. Solution which showed the best performance is implemented in production.

Creation of Dataset containing images taken from thermal camera to train the ML model. Training and deploying an object detection / tracking algorithm to work with the input given by the thermal camera. Interfacing of thermal camera with an embedded System on Module which can run machine learning workloads. Designing and researching effective and logical methods for crowd density [12]–[17]calculation.

CHAPTER 3
SYSTEM DEVELOPMENT

3.1 SYSTEM BLOCK DIAGRAM

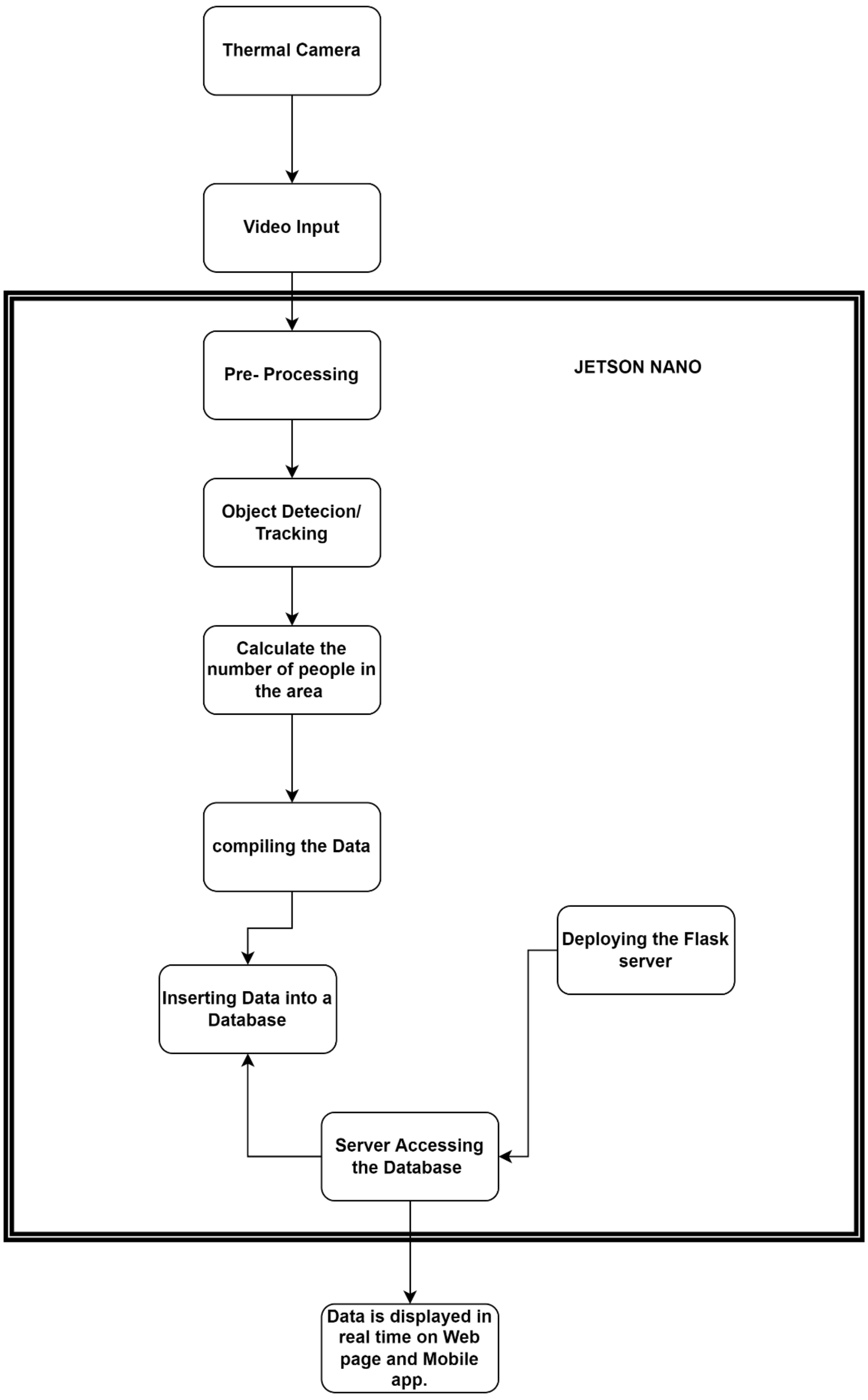

Thermal Camera
Video Input
Pre- Processing
JETSON NANO
Object Detecion/ Tracking
Calculate the number of people in the area
compiling the Data
Deploying the Flask server
Inserting Data into a Database
Server Accessing the Database
Data is displayed in real time on Web page and Mobile app.

Figure 1: System Block Diagram

3.2 SYSTEM SPECIFICATION

Hardware requirements -

1. **Edge Computing Device -**

- To run the model in a public place we will require a computing device which is compact and delivers the required performance.
- A device with dedicated GPU will be preferred over others as real time tracking with high FPS is required.
- Some edge computing devices are - NVIDIA Jetson Nano, Raspberry Pi[18], Beagle Board[19].
- **Raspberry Pi 3** –

  Raspberry Pi[20] is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom.

  This microcomputer is useful for small businesses that run on a lower budget to use their product or to invent new technology that embeds the product. Small business owners can use it to automate any small task, i.e., such as using the Pi to run a website or use it as a small database and media server.

  The product does not require users to have extensive programming experience since it is aimed for the younger generation to learn about programming. Python, the programming language i.e., Pi uses, is a smaller amount complex than other languages available. It has better code readability and allows the user to type concepts using fewer lines. Python also has an automatic memory management function.

  The Raspberry Pi is perfect for adaptive technology, and it is able to display images or play videos i.e., at high-definition resolution to building systems such as prototyping embedded systems. In our project we have tested 2 Raspberry Pi boards , the PI 3 and PI 4.

- Raspberry Pi 3 –

  Specifications –
  Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
  1GB RAM
  BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
  100 Base Ethernet
  40-pin extended GPIO
  4 USB 2 ports
  4 Pole stereo output and composite video port
  Full size HDMI
  CSI camera port for connecting a Raspberry Pi camera
  DSI display port for connecting a Raspberry Pi touchscreen display
  Micro SD port for loading your operating system and storing data
  Upgraded switched Micro USB power source up to 2.5A

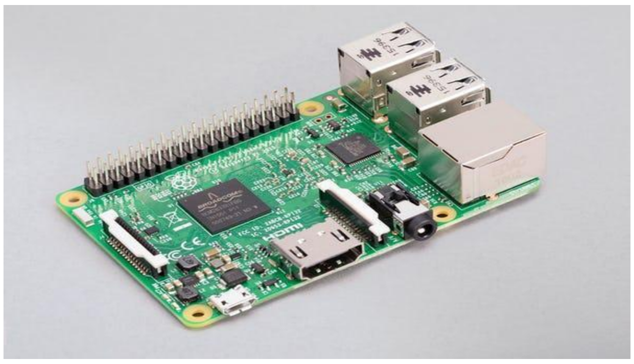


Figure 2: Raspberry Pi 3

- Raspberry Pi 4 –

  Specifications –
  Broadcom 2711, 64-bit quad-core Cortex-A72 processor
  4 GB RAM
  BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
  100 Base Ethernet
  40-pin extended GPIO
  4 USB 2 ports
  4 Pole stereo output and composite video port
  Dual micro-HDMI ports
  4K UHD video H.265 decode (4kp60)H.264 decode (1080p60)
  CSI camera port for connecting a Raspberry Pi camera
  DSI display port for connecting a Raspberry Pi touchscreen display
  Micro SD port for loading your operating system and storing data
  Upgraded switched Micro USB power source up to 2.5A

Figure 3: Raspberry Pi 4

- **NVIDIA Jetson Nano –**

  The small but powerful CUDA-X™ AI computer delivers 472 GFLOPS of compute performance for running modern AI workloads and is highly power-efficient, consuming as little as 5 watts.

  Jetson Nano supports high-resolution sensors, can process many sensors in parallel and can run multiple modern neural networks on each sensor stream. It also supports many popular AI frameworks, making it easy for developers to integrate their preferred models and frameworks into the product

  Key features of Jetson Nano include:

- GPU: 128-core NVIDIA Maxwell™ architecture-based GPU
- CPU: Quad-core ARM® A57
- Video: 4K @ 30 fps (H.264/H.265) / 4K @ 60 fps (H.264/H.265) encode and decode
- Camera: MIPI CSI-2 DPHY lanes, 12x (Module) and 1x (Developer Kit)
- Memory: 4 GB 64-bit LPDDR4; 25.6 gigabytes/second
- Connectivity: Gigabit Ethernet
- OS Support: Linux for Tegra®
- Module Size: 70mm x 45mm
- Developer Kit Size: 100mm x 80mm

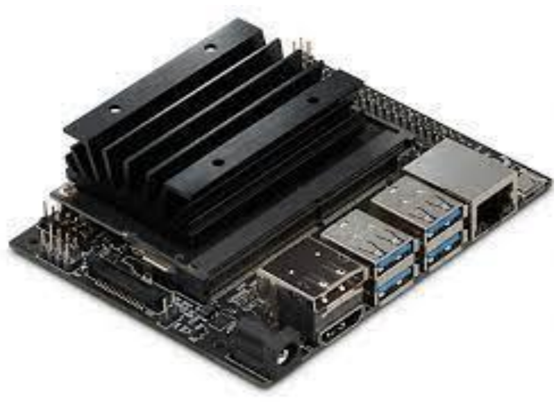


Figure 4: NVIDIA Jetson Nano Developer Kit

2. **Thermal Camera**[21]–[27] **-**

- Conventional surveillance cameras require light to produce images.
  However, optimal lighting isn't always available for security applications, especially in outdoor, remote, or rural locations.
- Traditional Cameras utilize visible light (450-700 nm) to create detailed color images for optimal target detection and object detection.
- But when scenarios arise where lighting is obscure and details that are to be captured from the image are critical Near infrared and thermal cameras are of most significance.
- These cameras are equipped in seeing through dust and rugged environments hence availability of pristine environments in not of concern for detection by these cameras.

- Thus for our application in this project, we need thermal cameras to estimate the crowd density not only at low lighting areas and at night but also during harsh outdoor conditions low visibility conditions.
- To familiarize ourselves with the concept of thermal cameras and their working we experimented with Megger TC 3231 Thermal camera , which is also used to collect data for our Custom Dataset.
- **Megger TC3231 -**

  The Megger TC3231 offers a professional infrared 32 x 31 pixels image thermometer with a 2.2 inch (55.88 mm) color TFT LCD display.

  Quick, accurate readings are possible; covering a wide range of surface temperature measurements. The product combines the convenience of an infrared thermometer with the visual advantage of a thermal imager providing a troubleshooting camera with infrared heat map. The TC3231 features a range of selectable thermal image color palette display options

  Some of its features are :

  - 2.2 inch (55.88 mm) 320*240 TFT LCD display
  - 32 x 31 pixel resolution IR temperature measurement
  - Measurement range -20 ° to 300 °C / -4 °F to 572 °F
  - Adjustable emissivity
  - Micro SD memory for storing up to 6000 images
  - High & Low Alarm (enable & disable)
  - Selectable color palette
  - Image blending with selectable distance of 0.5 m, 1 m, 2 m or 3 m

Figure 5 : Megger TC3231 Thermal Camera

- Even though the Megger TC 3231 thermal camera is great for scientific use and monitoring purposes but its main drawbacks come to light when we consider interfacing it with an edge computing device, it has no facility to enable streaming of video to give as an input to the edge computing device for processing purposes hence a need to find a thermal camera which meets the above said expectations was presented.

- Some thermal cameras which meet the above expectations and exceed are - FLIR Lepton 3, FLIR BOSON
- **FLIR Boson 320, 92° (HFOV) 2, 3mm -**

  The Boson longwave infrared (LWIR) [28]OEM thermal camera module sets the standard for size, weight, power, and performance (SWaP).

  The 12 µm pitch Vanadium Oxide (VOx) uncooled detector comes in two resolutions – 640 × 512 or 320 × 256.

  Both resolutions are available with multiple lens configurations, adding flexibility to integration programs. Radiometric models are also available with absolute temperature measurement in select configurations

  Thermal Imaging Detector - Uncooled VOx microbolometer

  Pixel Size - 12 µm

  Frame Rate Options - 60Hz baseline; 30 Hz runtime selectable

  Thermal Spectral Range - Longwave infrared; 8 µm – 14 µm

  Scene Temperature Range - to 140 °C (high gain) to 500 °C (low gain)

Figure 6: FLIR Boson

3. **IR Camera -**

- An infrared camera (also known as a thermal imager) detects and measures the infrared energy of objects. The camera converts that infrared data into an electronic image that shows the apparent surface temperature of the object being measured.
- Each pixel in the sensor array reacts to the infrared energy focused on it and produces an electronic signal. The camera processor takes the signal from each pixel and applies a mathematical calculation to it to create a color map of the apparent temperature of the object.

- Some IR Cameras are - Raspberry PI Infrared IR Night Vision Camera -

  Resolution – 5MP

  Sensors - Omnivision 5647 fixed-focus

  Interface type - CSI (Camera Serial Interface)

  Lens focus – Fixed focus

  Image Size (Pixels) – 2592 X 1944

  Supported Video Formats: 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 video

Figure 7: Raspberry Pi IR Cut Camera

4. **MLX90640 Thermal Camera -**

- The unavailability of the FLIR Cameras in the market due to Supply Chain Disruption led to implementation of this thermal camera.
- This sensor contains a 24x32 array of IR thermal sensors.
- When connected to your microcontroller (or Raspberry Pi) it will return an array of 768 individual infrared temperature readings over I2C.
- Programmable refresh rate 0.5Hz…64Hz (0.25 ~ 32 FPS)
- 3.3V-5V supply voltage, regulated to 3.3V on breakout
- Current consumption less than 23mA
- Field of view: 55°x35°
- Operating temperature -40°C ÷ 85°C
- Target temperature -40°C ÷ 300°C

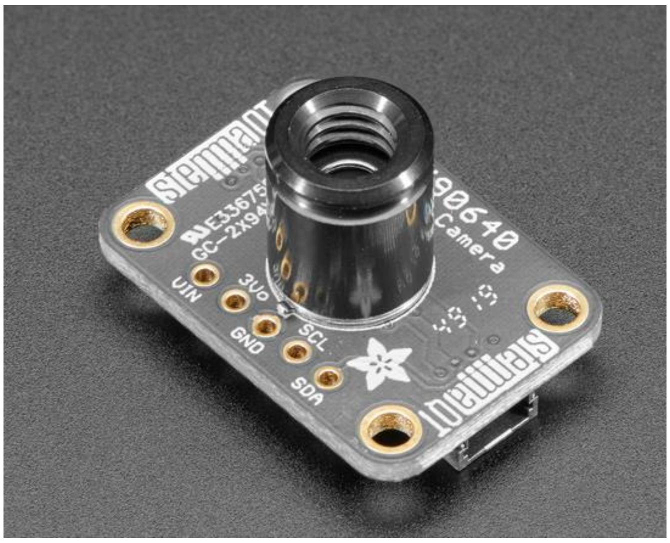


Figure 8: MLX90640 Camera

Software requirements -

1. **Python -**

- To implement the ML algorithms and train the model we need Python. Also, the edge devices which we have use Python to install their dependencies.
- The library support for Python is vast and some of which will be used are Scikit Learn, Pandas, TensorFlow, Matplotlib

2. **OpenCV -**

- We rely on OpenCV as it is a huge open-source library for computer vision, machine learning, and image processing.
- To identify whether the thermal image captured by the camera has a human in it we can use it and then estimate the density with respect to the frame of reference.

3. **TensorFlow -**

- Build and train models without sacrificing speed or performance.
- It is one of the best image recognition libraries with high accuracy results.
- TensorFlow also has their downscaled library such as TF-Lite which is recommended to use on edge computing devices due to lower computing capabilities as compared to traditional PC.

4. **Python SQLite3 –**

- Module is used to integrate the SQLite database with Python.
- It is a standardized Python DBI API 2.0 and provides a straightforward and simple-to-use interface for interacting with SQLite databases.

- There is no need to install this module separately as it comes along with Python after the 2.5x version.

5. **Flask -**

- It is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it.
- Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine.

6. **MIT App Inventor Platform -**

- MIT App Inventor is an intuitive, visual programming environment that is used to build fully functional apps for Android phones, iPhones, and Android/iOS tablets.
- It is used for implementing quick prototypes of applications to test out proof of concepts, before huge resources are put into developing the app.

3.3 CHALLENGES FACED

1. Definition of use case

While implementing the system we came across two use cases, free flowing crowd, and static crowd.

Free Flowing [29], [30]- A crowd, which is mainly seen in open markets, roadside markets, railway stations etc.

Static crowd - Something like a concert, stadium

<u>Solution</u> –

If we see a thermal image of any crowd in a stadium, we find that they are shown as small dots as compared to full body boxes in a free-flowing crowd. Therefore, a free-flowing crowd is much more suited for our application of density calculation.

2. Thermal Camera

The current thermal camera Megger TC3231 which we used for dataset collection, but this camera is not compatible with any edge device.

Also, the field of view of this camera is around 37° which itself is narrow for the application.

<u>Solution</u> –

FLIR Lepton 3, FLIR BOSON are some of the cameras we shortlisted. They are Jetson Nano compatible as well as have a wider field of view at 92°.

3. Database

The data generated during thermal detection needs to be saved and called upon for performing analytics. Deploying databases is a bit difficult and tedious on edge devices and hence getting a reliable solution is of the need.

Solution - SQLite is the key for this problem. It being available with python 3 packages, there is no need to install any other database service if not needed so.

4. Jetson Setup

Our code is code developed on a laptop with a different set of software versions and different packages. The setup of Jetson Nano with correct version installations is quite critical. Some of the installations which we had to do separately from the Jetpack 4.5 were - lxml, tqdm, absl-py, matplotlib, easydict, pillow.

Solution - Installation of dependencies using pip3 command on terminal. Such as - *pip3 install lxml* and more.

5. Tensorflow on edge

The Jetpack 4.5 which we are using does not come with Tensorflow installed for deep learning. The code on our laptop uses Tensorflow 2.3 for all the deep learning work. So to make the codes working on Jetson Nano we had to install Tensorflow 2.x on it.

Solution - Install Tensorflow 2.x on Jetson Nano using - *sudo pip3 install --pre --extra-index-url https://developer.download.nvidia.com/compute/redist/jp/v45 TensorFlow*

This successfully installs Tensorflow for deep learning.

6. Model Inference and Setup on Jetson Nano

The Yolo model in order to be implemented on the Jetson Nano, needs to be saved locally by using Frozen model and Trained weights. This process is ram intensive and fails if a ram bottleneck happens.

Solution - As the Jetson nano has only 4 gb available on board Ram and it is not upgradable, we increased the swap memory of the device from 2 gb to 4gb, after multiple tries the process was completed.

CHAPTER 4
SYSTEM IMPLEMENTATION

4.1 ALGORITHM IMPLEMENTATION

YOLO v4

YOLO v4[3], [31], [32] is an improvement on the YOLO v3[21] algorithm architecture, by having a significant improvement in the mean average precision (mAP) by as much as 10% and the number of frames per second by 12%. The architecture of Yolo v4 consists of 4 distinct blocks as shown.

The backbone, the neck, the dense prediction, sparse prediction.

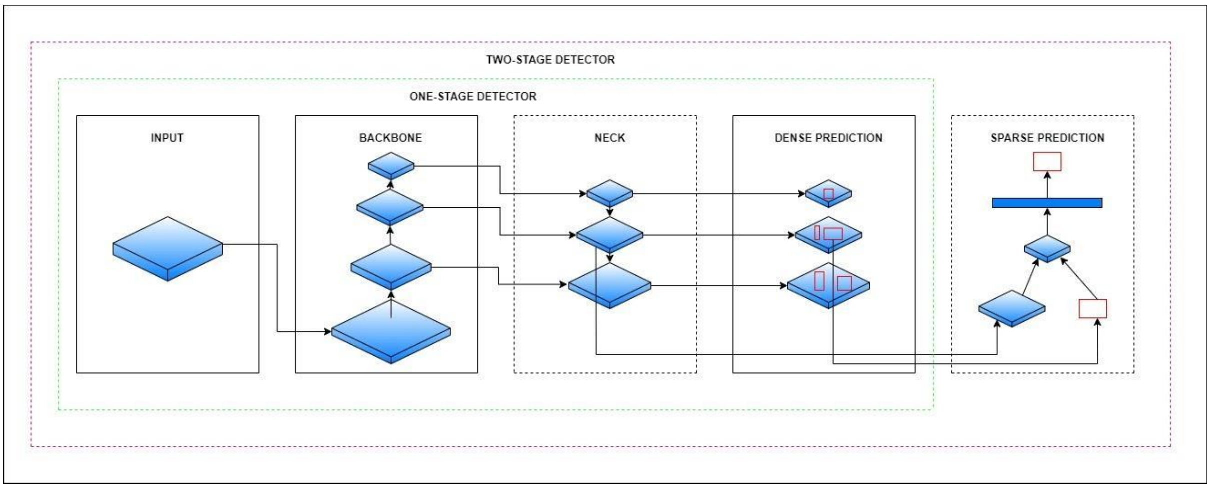


Figure 9: YOLO v4 Distinct Blocks

The backbone is the feature extraction architecture which is the CSPDarknet53. This CSPDarknet53 stands for Cross-Spatial -Partial connections, which is used to split the current layer into two parts, one to pass through convolution layers and the other that would not pass-through convolutions, after which the results are aggregated. Below is an example with DenseNet[33].

The neck helps to add layers between the backbone and the dense prediction block(head), which is a bit like what the ResNet[33] architecture does. The yolov4 architecture uses a modified Path aggregation network, a modified spatial attention module, and a modified spatial pyramid pooling, which are all used to aggregate the information to improve accuracy. The image below shows spatial pyramid pooling.

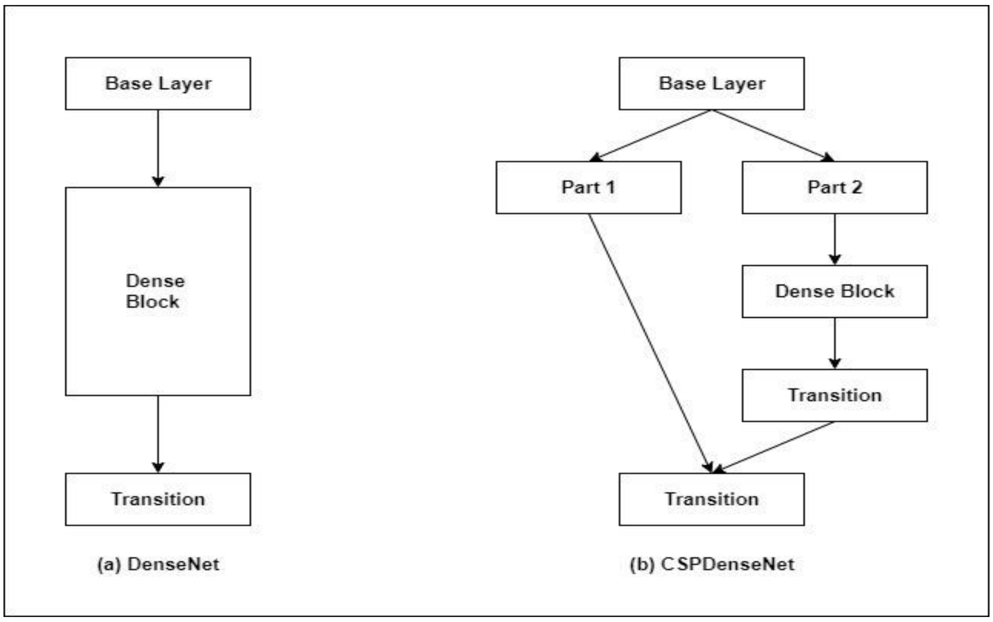


Figure 10: DenseNet vs CSPDenseNet

The head (Dense prediction) is used for locating bounding boxes and for classification. The process is the same as the one described for Yolo v3, the bounding box coordinates (x,y, height, and width) are detected as well as the score. Remember, the main goal of the Yolo algorithm is to divide an input image into several grid cells and predict the probability that a cell contains an object using anchor boxes. The output is then a vector with the bounding box coordinates and the probabilities of the classes.

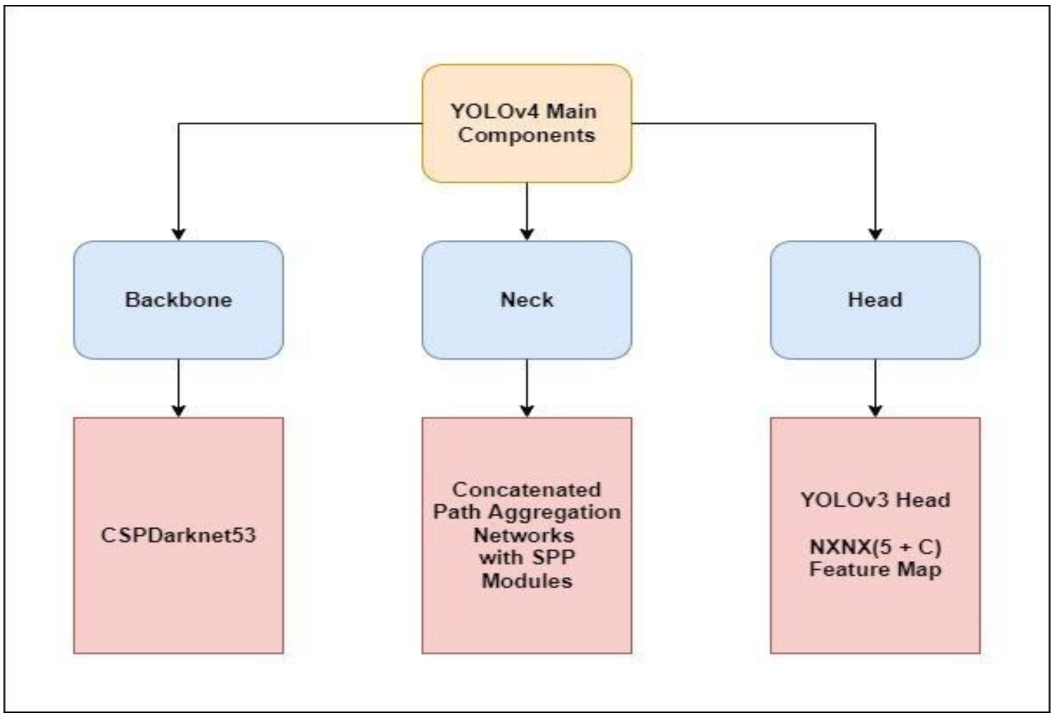


Figure 11: Basic outline of YOLO v4

4.2 ALGORITHM TRAINING

YOLO v4 is needed to be trained on a custom dataset in order to achieve better accuracy in human detection .

Custom Dataset

The Algorithm has been trained on a custom dataset which has been specifically curated to showcase crowded environments along with thermal images and Infrared images. This would give the algorithm a wide array edge cases and scenarios. The images were collected for numerous sources, namely Megger TC 3231 Thermal Camera, Thermal Videos available on the internet and Thermal people and dogs Dataset. The Dataset consists of around 500 images.

The annotation of the Dataset was performed on Roboflow tool, and the images were annotated for Dog and Human class.

The following images are taken by Megger TC3231 for our custom dataset.

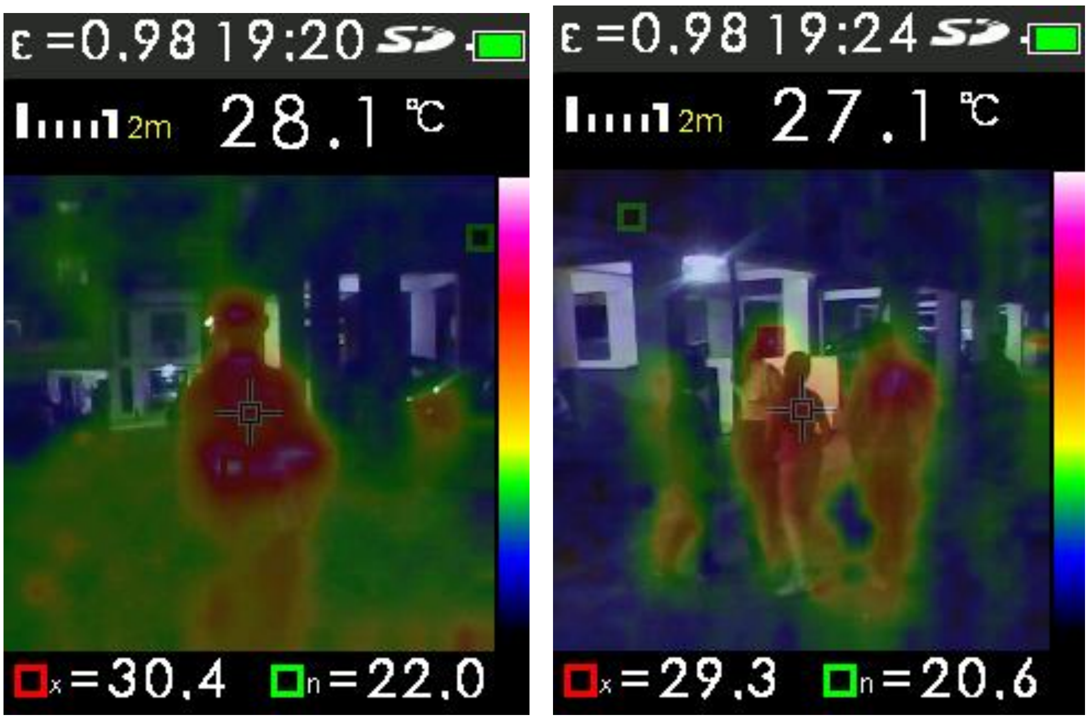


Figure 12: Megger TC3231 Dataset Images

The following images are sample images from our dataset which covers the IR spectrum and the Visible light spectrum.

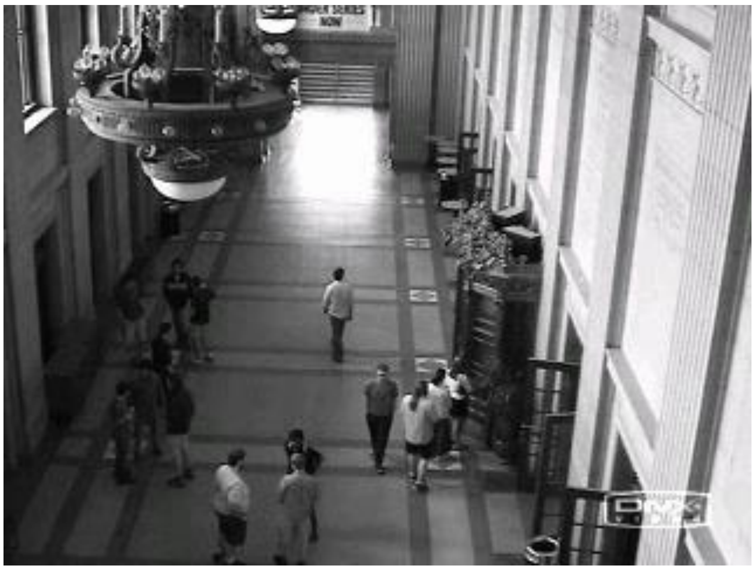


Figure 13: Sample images from our custom dataset covering IR Spectrum.

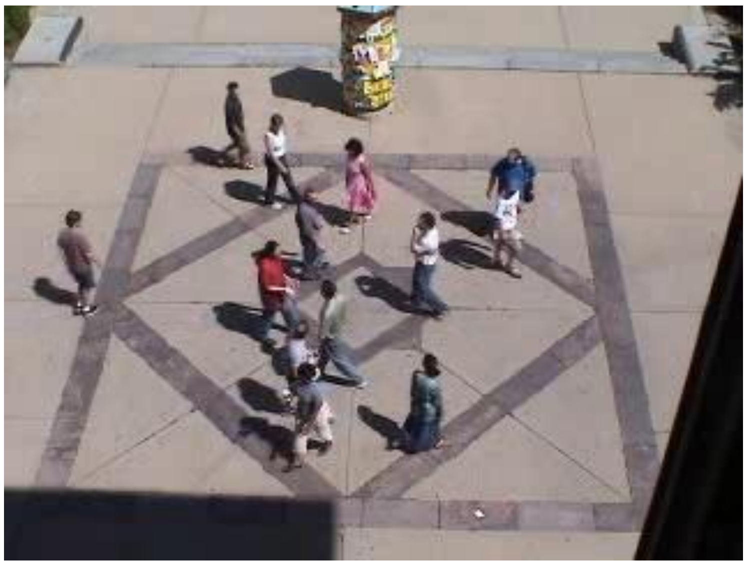


Figure 14: Sample images from our custom dataset covering Visible Spectrum.

The following images are sample images form our annotated dataset . The dataset spans over 450 images which are annotated for class Humans and Dogs and contain a varied images ranging from thermal , IR and from the visible spectrum to Train a versatile model.

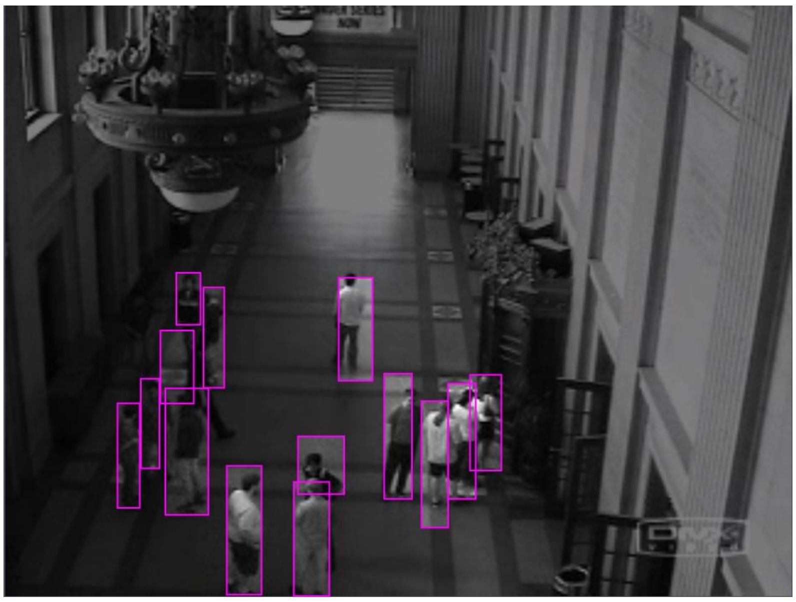


Figure 15: Sample annotated images from our custom dataset for IR imaging.

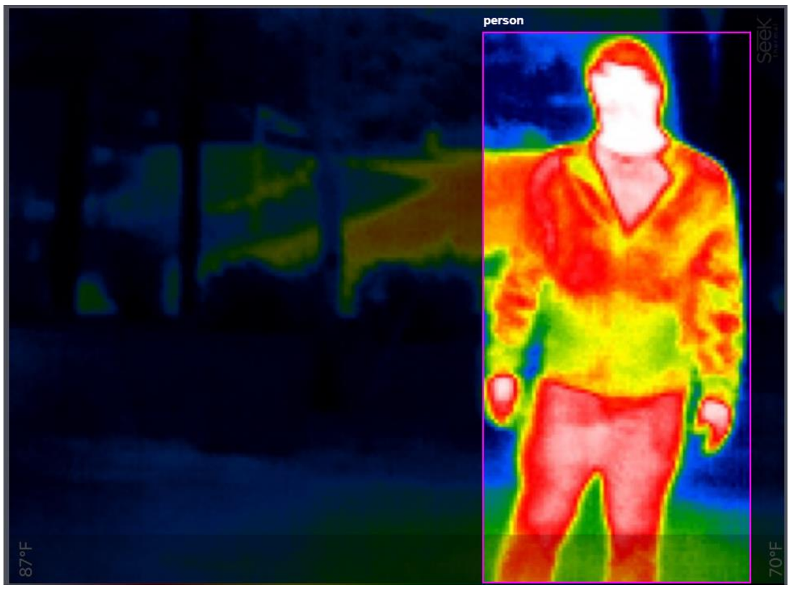


Figure 16: Sample annotated images from our custom dataset for Thermal Imaging.

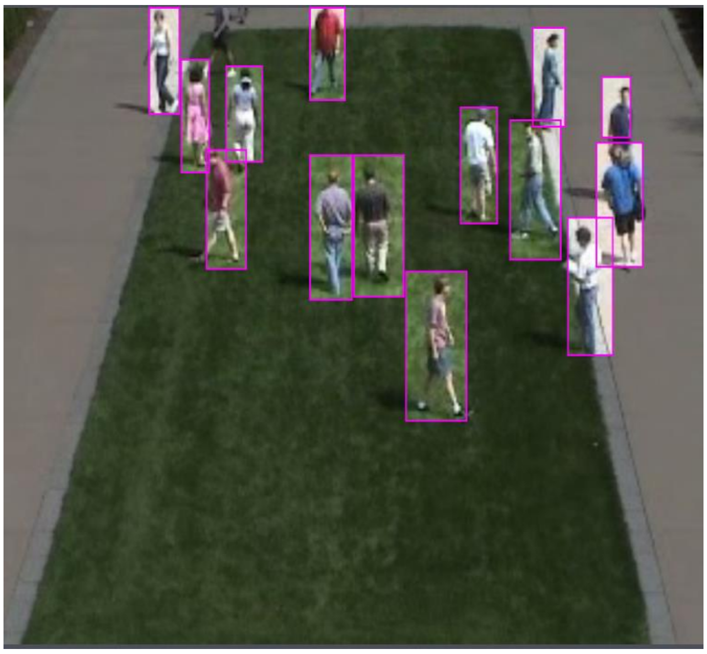


Figure 17: Sample annotated images from our custom dataset Visible spectrum Imaging.

Custom Training

The Yolo v4 algorithm was custom trained on the dataset using the transfer learning technique .For the YOLO v4 config file, we used specific values aligning with our classes. Batches are set to 64 and subdivisions set to 16. If the runtime goes into an issue, then we can give subdivisions set to 32. Width and height are set to 416 each as these values can be a multiple of 32, we can improve the results by making the value larger like 608 but in turn, the training will slow down. Max_batches are always given a – number of classes * 2000, therefore for our case we 8 classes so max batches will be 16000.Steps are 80% and 90% of max batches, so for max batches, 16000 steps will be 12800, 14400.Setting the filters in the config file – (number of classes + 5) * 3, so for 8 classes in this case the filters will be 39. We set the model to run for 6000 iterations. Once the average loss is under 2.0, we stop the training to avoid overfitting and overtraining of the model.

4.3 HARDWARE IMPLEMENTATION

Jetson Nano setup and Object detection model Inferencing

The custom trained model needs to be implemented on an edge computing device, the device chosen for this purpose is Jetson Nano. The process of implementing the algorithm is given below.

Step 1 - Download the Jetson Nano Developer Kit SD Card Image

Step 2 - Use Etcher to write the Jetson Nano Developer Kit SD Card Image to your microSD card

Step 3 - Installing TensorFlow For Jetson Platform

Step 3.1 - Install system packages required by TensorFlow:

```
$ sudo apt-get update
```

```
$ sudo apt-get install libhdf5-serial-dev hdf5-tools libhdf5-dev zlib1g-dev zip libjpeg8-dev liblapack-dev libblas-dev gfortran
```

Step 3.2 - Install and upgrade pip3:

```
$ sudo apt-get install python3-pip
$ sudo pip3 install -U pip testresources setuptools==49.6.0
```

Step 3.3 - Install Python package dependencies

```
$ sudo pip3 install -U --no-deps numpy==1.19.4 future==0.18.2 mock==3.0.5 keras_preprocessing==1.1.2 keras_applications==1.0.8 gast==0.4.0 protobuf pybind11 cython pkgconfig
```

```
$ sudo env H5PY_SETUP_REQUIRES=0 pip3 install -U h5py==3.1.0
```

Step 3.4 - Install TensorFlow using the following command

```
$sudo pip3 install --pre --extra-index-url https://developer.download.nvidia.com/compute/redist/jp/v45 TensorFlow
```

Step 4 - Port our object detection files from Laptop to Jetson

Step 5 - Install dependencies for running the code

- pip3 install absl-py
- pip3 install matplotlib
- pip3 install easydict
- pip 3 install pillow
- pip3 install lxml
- pip3 install tqdm

Step 6 - Downloading Official YOLOv4 Pre-trained Weights

Step 7 - Build and run the code.

Save yolov4-tiny model command -

```
python3 save_model.py --weights ./data/yolov4-tiny.weights --output ./checkpoints/yolov4-tiny-416 --model yolov4 –tiny
```

Step 8 - always run this command on the terminal before running the final object tracking codes.

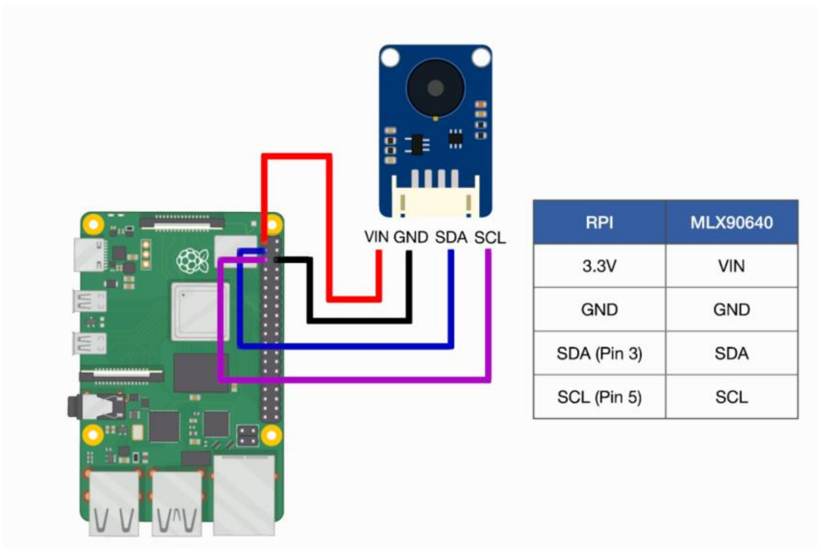
Command - export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1

Step 9 - Run yolov4-tiny object tracker.

```
python3 object_tracker.py --weights ./checkpoints/yolov4-tiny-416 --model yolov4 --video ./data/video/test.mp4 --output ./outputs/tiny.avi –tiny
```

Interfacing Raspberry Pi 4 with MLX 90640

The MLX90640 thermal module needs to be interfaced with the Raspberry Pi 4 Module according to the interfacing Diagram shown below.



RPI	MLX90640
3.3V	VIN
GND	GND
SDA (Pin 3)	SDA
SCL (Pin 5)	SCL

Figure 18: MLX90640 Interfacing with Raspberry PI.

After the connections are done , dependencies like Scipy , numpy etc. are needed to be installed. After that the I2C interface on the Raspberry PI needs to be turned on.

Using the command ”sudo i2cdetect -y -1” it is checked that Raspberry Pi correctly registers MLX90640 and the I2C address of the MLX90640 is seen on the terminal.

The the Adafruit libraries like Adafruit blinka and Circuitpython- MLX90640 are installed. After which a python script is used to access the camera and get real time thermal image on the screen. The following image is of actual interfacing of our Raspberry pi 4 and thermal camera.

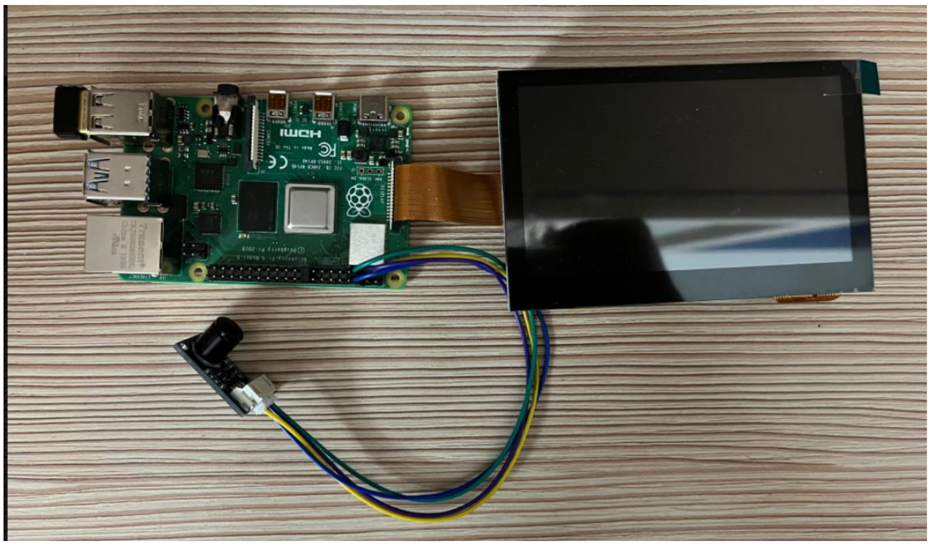


Figure 19: Actual Interfacing of Raspberry PI 4 with MLX90640.

CHAPTER 5
RESULT AND ANALYSIS

5.1 RESULTS ANALYSIS AND IMPLEMENTATION

After the implementation of the pre-trained model on local machine for testing purposes it is tested on a sample video. This testing has provided us on insights on whether to use the pre-trained model for implementation or using a custom trained model.

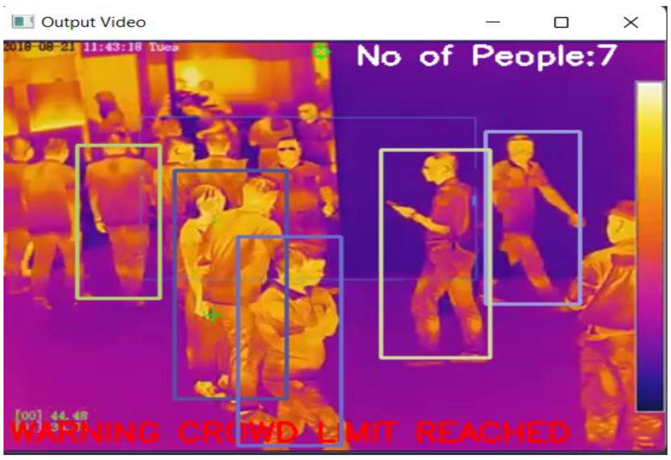


Figure 20: Crowd Limit Reached Output

The result screenshots show that the pre- trained model can detect solitary humans in the video but when a group is seen it is not able to give concrete detections. This hints us that custom training of the model is of utmost need, and it needs to be done.

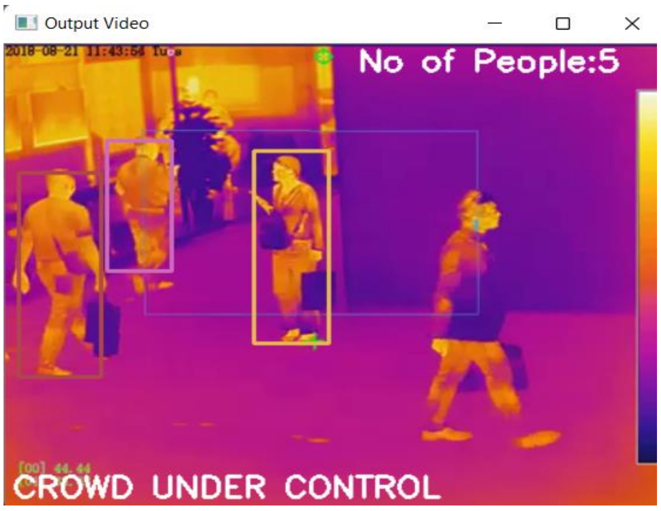


Figure 21: Crowd Under Control Output

The custom trained model when implemented on a local machine has given the following output. The drawbacks that were seen on the pre-trained model have been resolved to an extent by utilizing the Custom training approach.

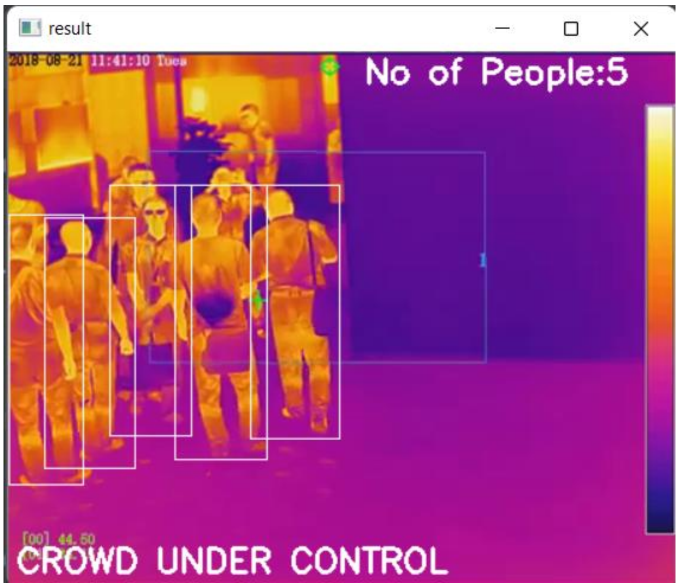


Figure 22: Crowd Under Control Output

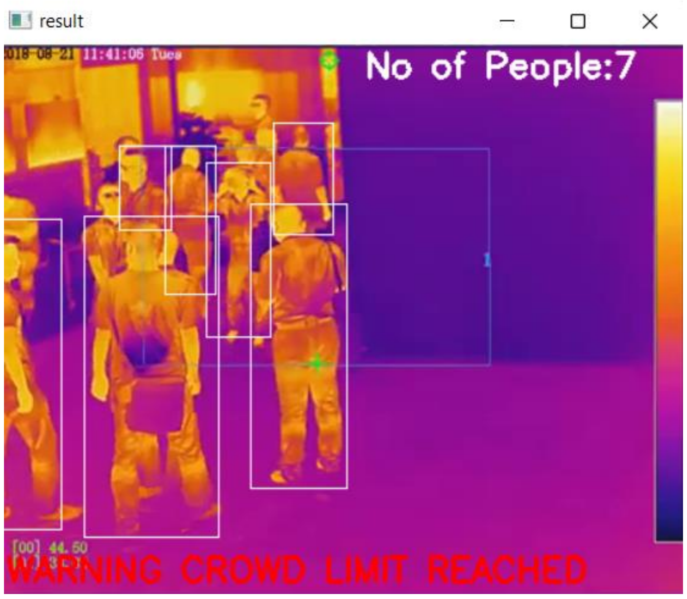
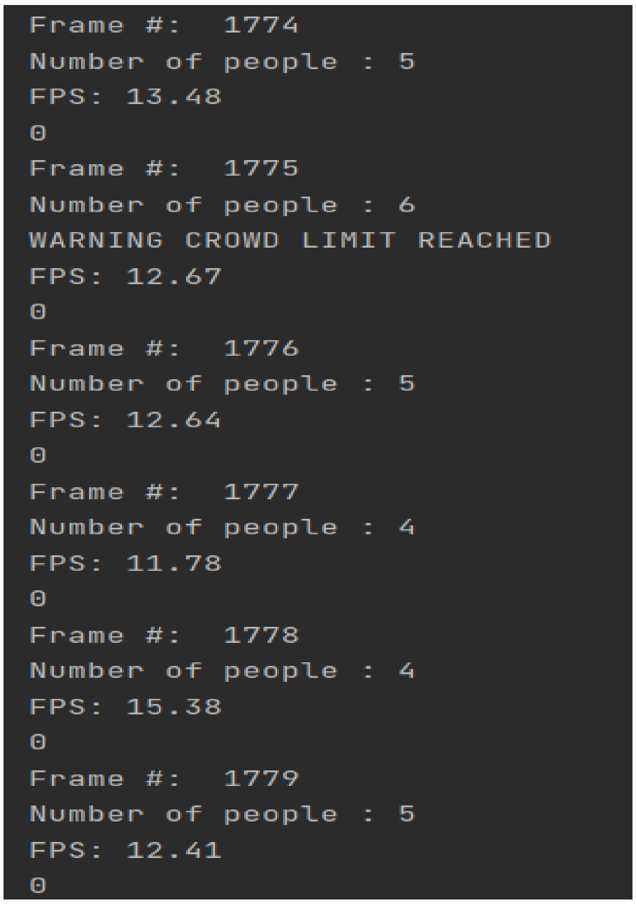


Figure 23: Crowd Limit Reached Output

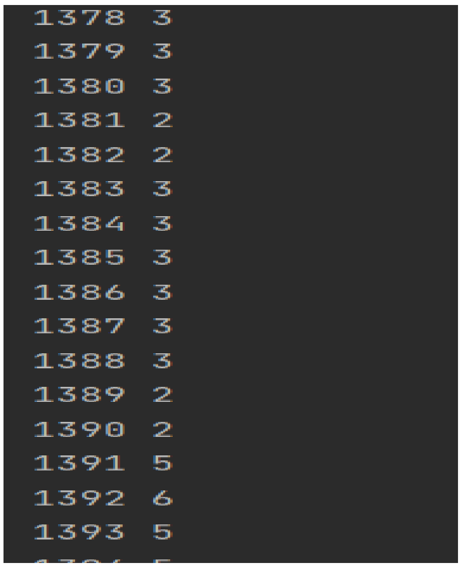
The Console screen shot below shows the FPS at which model is giving the output along with the number of people in each frame. The output will further be needed to be stored on a Database to perform analytics



```
Frame #:  1774
Number of people : 5
FPS: 13.48
0
Frame #:  1775
Number of people : 6
WARNING CROWD LIMIT REACHED
FPS: 12.67
0
Frame #:  1776
Number of people : 5
FPS: 12.64
0
Frame #:  1777
Number of people : 4
FPS: 11.78
0
Frame #:  1778
Number of people : 4
FPS: 15.38
0
Frame #:  1779
Number of people : 5
FPS: 12.41
0
```

Figure 24: Console Log Output

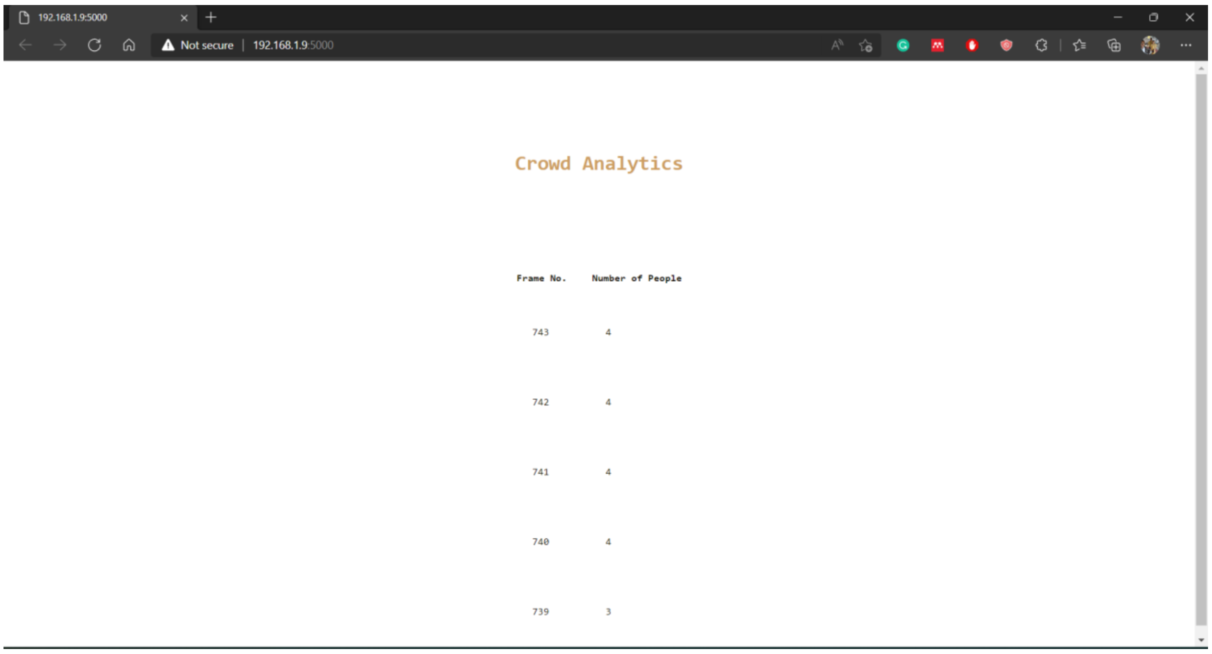
The Console screenshot below represents how the data generated during the running of the object detection code is stored and retrieved.



```
1378 3
1379 3
1380 3
1381 2
1382 2
1383 3
1384 3
1385 3
1386 3
1387 3
1388 3
1389 2
1390 2
1391 5
1392 6
1393 5
```

Figure 25: Database Output

The flask server that was implemented for showcasing the data generated is shown below. The page loads up on the Localhost and shows the previous 5 frames of Data



192.168.1.9:5000

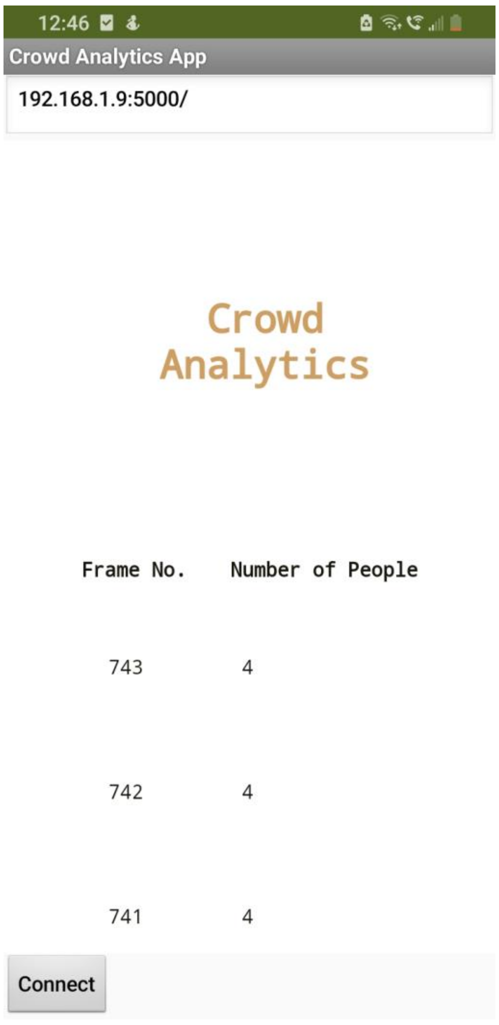
Not secure | 192.168.1.9:5000

Crowd Analytics

Frame No.	Number of People
743	4
742	4
741	4
740	4
739	3

Figure 26: Webpage Output

The Flask server is also accessed by the prototype mobile app and displays the analytics.



192.168.1.9:5000/

Crowd Analytics

Frame No.	Number of People
743	4
742	4
741	4

Figure 27: App Output

The object detection model is being inferenced on jetson nano on the input video which was streamed USB Camera. From the result given below it is seen that the model is successfully able to identify humans and also give out clear indication of the Crowd Density of that particular area.

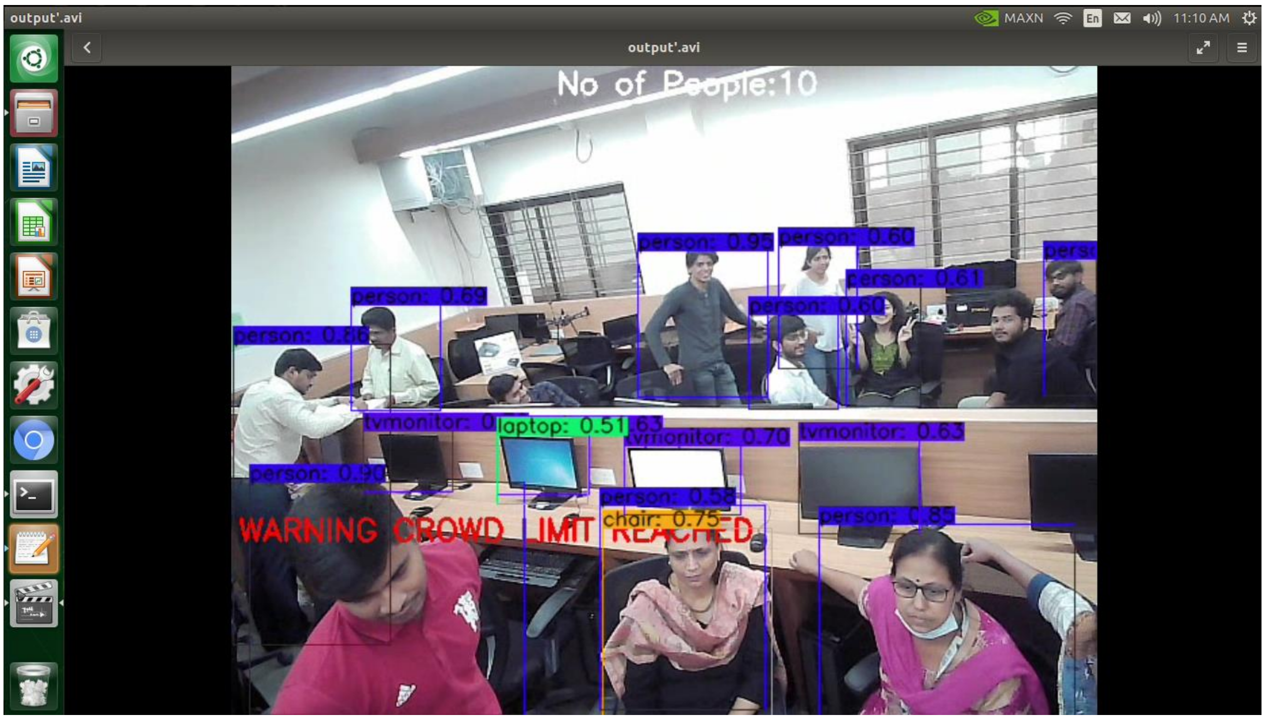


Figure 28: Jetson Nano Inferencing output.

The MLX90640 was interfaced with raspberry pi and the output achieved is shown below.

The output is a thermal image generated by the 32 x32 array of thermal sensors on the MLX90640.

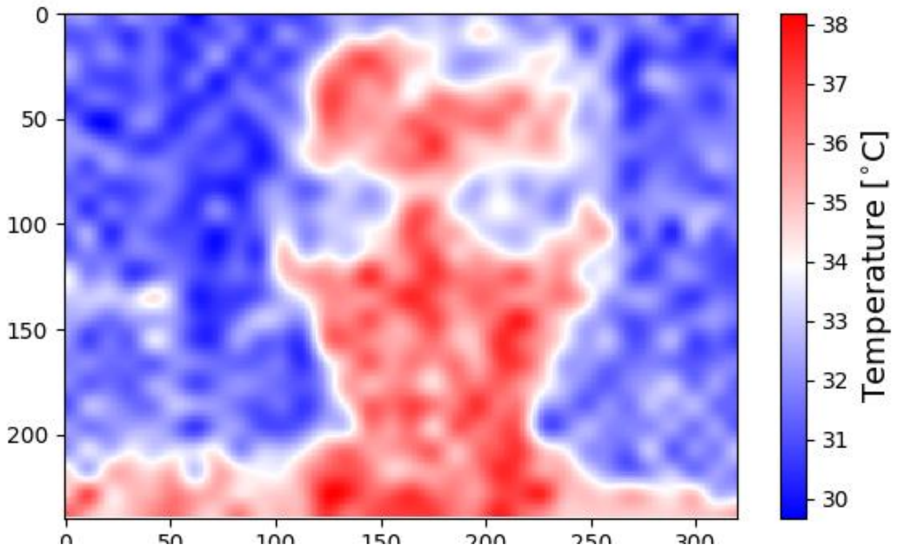


Figure 29: MLX90640 Output 1

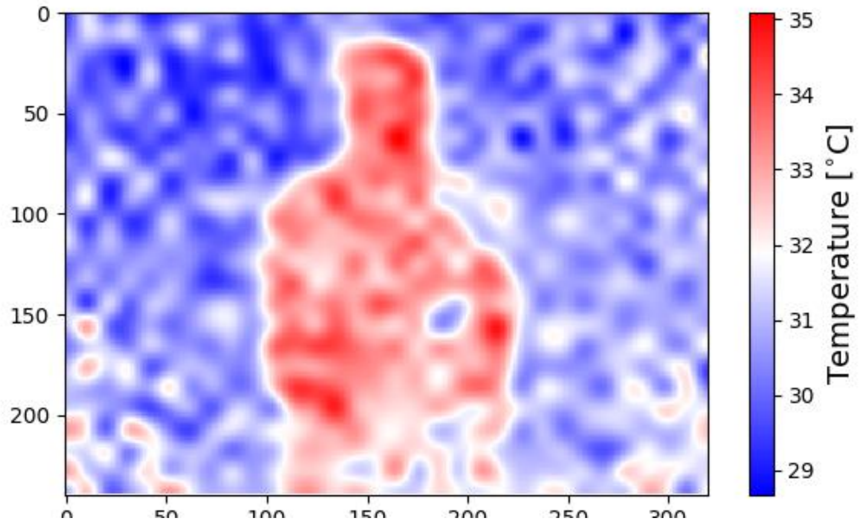


Figure 30: MLX90640 Output 2

CHAPTER 6
DISCUSSION AND CONCLUSION

6.1 DISCUSSION

Work done till date –

- Selection of Object Detection Algorithm
- Testing object detection algorithm for crowd analytics
- Testing and Evaluating object detection algorithm for thermal images and videos.
- Testing thermal camera Megger TC 3231.
- Evaluating the Thermal Camera.
- Research For a New Thermal Camera.
- Collecting Training Data
- Annotation of Training Data (in progress)
- Researching on methods to dynamically store detection data.
- Retrieving detection data and deploying a Flask server on a local host.
- Development of HTML page to serve the data on local host.
- Setting up of Raspberry Pi for IR Camera testing
- Setting up Jetson Nano
- Deployment of Yolo v4 on Jetson nano
- Development of Prototype app for Crowd Analytics
- Testing Tensorflow Lite for Jetson Nano Deployment.
- Collection of Custom Dataset.
- Annotation of Dataset Images.
- Custom Training of models.
- Interfacing MLX90640 Thermal camera

Key learnings till date –

- Working of Thermal Camera and Research on finding optimum camera for our use.
- Working of object detection algorithms.
- Research on various object detection algorithms.
- Dataset creation using the thermal camera.
- Annotation of thermal images.
- Implementation of Yolo algorithm on Local machine for testing purposes.

- Implementation of SQlite database in python.
- Setting up of Jetson Nano
- working of MIT App inventor
- annotation of Dataset images
- Working of Flask Library
- implementation of transfer learning for Machine Learning Algorithms.

6.2 FUTURE SCOPE AND APPLICATION

The future scope for this project lies in making a pre-production prototype, which would work flawlessly utilizing the fine-tuned object detection algorithm .

The app and the website proposed would be deployment ready, and would provide immense information on the data collected, i.e., crowd analytics.

The combination of temperature detection and alert generation can also be explored.

The main possible applications for our project are deployment in areas where the crowd is free-flowing, these areas include Railway Stations, Temples , malls , Schools.

This project can also be highly useful in Surveillance and intruder detection applications.

6.3 CONCLUSION

The use of thermal camera for crowd estimation and analytics was successfully looked upon, along with interfacing the said thermal camera with various SoC boards.

Object detection algorithms like YOLO were implemented to detect humans in RGB scale thermal images and a pre-trained YOLO model was tested for RGB scale thermal video. This resulted in identification in disparity between the ground truth and detection given by the model and hence custom training for the algorithm was done. Further model was implemented on Edge computing device (Jetson Nano) and results achieved were showcased using flask server and mobile application.

The MLX90640 Thermal Camera was successfully interfaced with Raspberry pi and outputs were noted.

REFERENCES

[1] B. Yılmaz, S. N. H. Sheikh Abdullah, and V. J. Kok, “Vanishing region loss for crowd density estimation,” *Pattern Recognition Letters*, vol. 138, 2020, doi: 10.1016/j.patrec.2020.08.001.

[2] L. Zhu, C. Li, Z. Yang, K. Yuan, and S. Wang, “Crowd density estimation based on classification activation map and patch density level,” *Neural Computing and Applications*, vol. 32, no. 9, 2020, doi: 10.1007/s00521-018-3954-7.

[3] M. Ivašić-Kos, M. Krišto, and M. Pobar, “Human detection in thermal imaging using YOLO,” in *ACM International Conference Proceeding Series*, 2019, vol. Part F148262. doi: 10.1145/3323933.3324076.

[4] L. Zheng, Y. Yang, and Q. Tian, “SIFT Meets CNN: A Decade Survey of Instance Retrieval,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5. 2018. doi: 10.1109/TPAMI.2017.2709749.

[5] B. Yang and S. Chen, “A comparative study on local binary pattern (LBP) based face recognition: LBP histogram versus LBP image,” *Neurocomputing*, vol. 120, 2013, doi: 10.1016/j.neucom.2012.10.032.

[6] Y. Zhong, L. Sun, C. Ge, and H. Fan, “Hog-esrs face emotion recognition algorithm based on hog feature and esrs method,” *Symmetry*, vol. 13, no. 2, 2021, doi: 10.3390/sym13020228.

[7] S. A. Alazawi, N. M. Shati, and A. H. Abbas, “Texture features extraction based on GLCM for face retrieval system,” *Periodicals of Engineering and Natural Sciences*, vol. 7, no. 3, 2019, doi: 10.21533/pen.v7i3.787.

[8] A. F. Schmidt and C. Finan, “Linear regression and the normality assumption,” *Journal of Clinical Epidemiology*, vol. 98. 2018. doi: 10.1016/j.jclinepi.2017.12.006.

[9] R. Frühwirth, “Regression with gaussian mixture models applied to track fitting,” *Instruments*, vol. 4, no. 3, 2020, doi: 10.3390/instruments4030025.

[10] I. Shopovska, L. Jovanov, and W. Philips, "Deep visible and thermal image fusion for enhanced pedestrian visibility," *Sensors (Switzerland)*, vol. 19, no. 17, 2019, doi: 10.3390/s19173727.

[11] E. S. Jeon *et al.*, "Human detection based on the generation of a background image by using a far-infrared light camera," *Sensors (Switzerland)*, vol. 15, no. 3, 2015, doi: 10.3390/s150306763.

[12] C. C. Lai and H. C. Chiu, "Crowd density estimation using Taylor expansion and local binary count descriptor," *ICIC Express Letters, Part B: Applications*, vol. 11, no. 5, 2020, doi: 10.24507/icicelb.11.05.479.

[13] M. S. Saleem, M. J. Khan, K. Khurshid, and M. S. Hanif, "Crowd density estimation in still images using multiple local features and boosting regression ensemble," *Neural Computing and Applications*, vol. 32, no. 21, 2020, doi: 10.1007/s00521-019-04021-2.

[14] Z. Fan, H. Zhang, Z. Zhang, G. Lu, Y. Zhang, and Y. Wang, "A survey of crowd counting and density estimation based on convolutional neural network," *Neurocomputing*, vol. 472, 2022, doi: 10.1016/j.neucom.2021.02.103.

[15] Y. B. Liu, R. S. Jia, J. T. Yu, R. N. Yin, and H. M. Sun, "Crowd density estimation via a multichannel dense grouping network," *Neurocomputing*, vol. 449, 2021, doi: 10.1016/j.neucom.2021.03.078.

[16] M. A. Kizrak and B. Bolat, "Crowd Density Estimation by Using Attention Based Capsule Network and Multi-Column CNN," *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3081529.

[17] A. Hafeezallah, A. Al-Dhamari, and S. A. R. Abu-Bakar, "U-ASD Net: Supervised Crowd Counting Based on Semantic Segmentation and Adaptive Scenario Discovery," *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3112174.

[18] A. A. Suzen, B. Duman, and B. Sen, "Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN," 2020. doi: 10.1109/HORA49412.2020.9152915.

[19] N. B and H. Manjunath, "A Design of Novel Algorithm for Image Steganography Using Discrete Wavelet Transformation on Beagle Board-XM," *IOSR Journal of Electronics and Communication Engineering Ver. I*, vol. 10, no. 3, 2015.

[20] Raspberry Pi, "Raspberry Pi 4 Model B specifications – Raspberry Pi," *Raspberry Pi Foundation*. 2020.

[21] R. Kalita, A. K. Talukdar, and K. Kumar Sarma, "Real-Time Human Detection with Thermal Camera Feed using YOLOv3," 2020. doi: 10.1109/INDICON49873.2020.9342089.

[22] M. G. Pavlović *et al.*, "Advanced thermal camera based system for object detection on rail tracks," *Thermal Science*, vol. 22, 2018, doi: 10.2298/TSCI18S5551P.

[23] M. Younsi, M. Diaf, and P. Siarry, "Automatic multiple moving humans detection and tracking in image sequences taken from a stationary thermal infrared camera," *Expert Systems with Applications*, vol. 146, 2020, doi: 10.1016/j.eswa.2019.113171.

[24] E. S. Jeon, J. H. Kim, H. G. Hong, G. Batchuluun, and K. R. Park, "Human detection based on the generation of a background image and fuzzy system by using a thermal camera," *Sensors (Switzerland)*, vol. 16, no. 4, 2016, doi: 10.3390/s16040453.

[25] R. Gade and T. B. Moeslund, “Thermal cameras and applications: A survey,” *Machine Vision and Applications*, vol. 25, no. 1, 2014, doi: 10.1007/s00138-013-0570-5.

[26] V. Guerra, J. R. Ticay-Rivas, V. Alonso-Eugenio, and R. Perez-Jimenez, “Characterization and performance of a thermal camera communication system,” *Sensors (Switzerland)*, vol. 20, no. 11, 2020, doi: 10.3390/s20113288.

[27] G. Batchuluun, J. K. Kang, D. T. Nguyen, T. D. Pham, M. Arsalan, and K. R. Park, “Deep Learning-Based Thermal Image Reconstruction and Object Detection,” *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2020.3048437.

[28] K. M. Judd, M. Thornton, and A. Richards, “Automotive sensing: assessing the impact of fog on LWIR, MWIR, SWIR, visible, and lidar performance,” 2019. doi: 10.1117/12.2519423.

[29] W. Liu, M. Salzmann, and P. Fua, “Estimating People Flows to Better Count Them in Crowded Scenes,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020, vol. 12360 LNCS. doi: 10.1007/978-3-030-58555-6_43.

[30] W. Liu, M. Salzmann, and P. Fua, “Counting People by Estimating People Flows,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021, doi: 10.1109/TPAMI.2021.3102690.

[31] M. Kristo, M. Ivasic-Kos, and M. Pobar, “Thermal Object Detection in Difficult Weather Conditions Using YOLO,” *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3007481.

[32] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, “YOLOv4,” *CVPR Workshop on The Future of Datasets in Vision*, 2020.

[33] C. Zhang *et al.*, “ResNet or DenseNet? Introducing dense shortcuts to ResNet,” 2021. doi: 10.1109/WACV48630.2021.00359.