DIRECTORATE OF ONLINE EDUCATION

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

MCA IV SEM

PROJECT VIVA VOCE

December 2024

ENROLMENT NUMBER : EC2332251010137
NAME OF STUDENT: BHASKAR SARMA PATRAYADI

Optimizing Deep Learning Techniques for Enhanced Real-Time Object Detection

PROJECT GUIDE : Dr.G.Babu

CONTENTS

1. Abstract
2. Introduction
3. Project Scope
4. Product Functions
5. Features
6. Module Description
7. System Requirements
8. Existing System
9. Proposed System
10. Data Flow Diagram
11. Workflow
12. Sequence Diagram
13. Output Screenshots
14. Source Code
15. Conclusion
16. Feature Enhancements

ABSTRACT

- **Project Overview:** Optimize deep learning models using MATLAB, Python, and advanced libraries for efficient real-time object detection, focusing on deployment to devices like Raspberry Pi for social distancing monitoring.

- **Deep Learning Intro:** Deep learning transforms real-time object detection, essential for tasks like social distancing, with challenges in optimizing for resource-limited devices like Raspberry Pi.

- **Objective:** Create optimized models for real-time detection, targeting practical uses such as social distancing.

- **Methodology:**
  - Develop models with MATLAB and Python.
  - Integrate libraries: Intel MKL-DNN, NVIDIA cuDNN, ARM Compute Library.
  - Apply optimization techniques: pruning, quantization, and lightweight architectures.

ABSTRACT

Analysis:

- Assess model complexity and computational needs.
- Evaluate hardware capabilities: CPUs, GPUs, ARM devices.
- Investigate real-time processing solutions, including edge computing.

Workflow

.

1. **Data Collection:** Compiled a dataset from real-time video feeds for object detection, focusing on social distancing.

2. **Model Optimization:** Developed and refined deep learning models in MATLAB and Python using pruning and quantization for better performance.

3. **Library Integration:** Used Intel MKL-DNN, NVIDIA cuDNN, and ARM Compute Library to boost computational speed and deployment efficiency.

ABSTRACT

4. **Embedded Deployment:** Deployed optimized models on Raspberry Pi for efficient real-time processing in limited-resource environments.

5. **Edge Computing:** Applied strategies to minimize latency and boost system responsiveness for immediate data processing needs.

6. **Application Development:** Customized detection systems for specific uses, like social distancing, to ensure precise distance measurement.

7. **Performance Evaluation:** Assessed system performance to establish operational benchmarks for real-time applications.

8. **Data Visualization:** Used visualization tools to clearly present detection results, facilitating communication and analysis.

9. **Real-World Impact:** Showcased the benefits of optimized models in enhancing object detection capabilities for practical applications.

INTRODUCTION

Overview of the Project

Development of an Optimized Object Detection System for real-time applications.

Purpose and Significance

Enhance the efficiency and accuracy of object detection, especially for social distancing monitoring. Facilitate deployment on embedded devices like Raspberry Pi.

Project Scope

Optimize deep learning models using MATLAB, Python, and advanced libraries. Deploy solutions on resource-constrained platforms for real-time processing.

Expected Outcomes

Improved real-time object detection with minimal computational resources. Efficient deployment on embedded systems for practical applications.

PROJECT SCOPE

Create DL Network

- Create the Deep Learning network using dl network designer in MATLAB

Quantize Network

- Use dlquantizer in MATLAB to quantize the network

Take Advantage of ACL (ARM Compute Library)

- Use ACL to use optimized ARM intrinsics (NEON intrinsics) and ARM assembly instructions

Use Embedded Cooder and Raspi

- Use dlquantizer in MATLAB to quantize the network

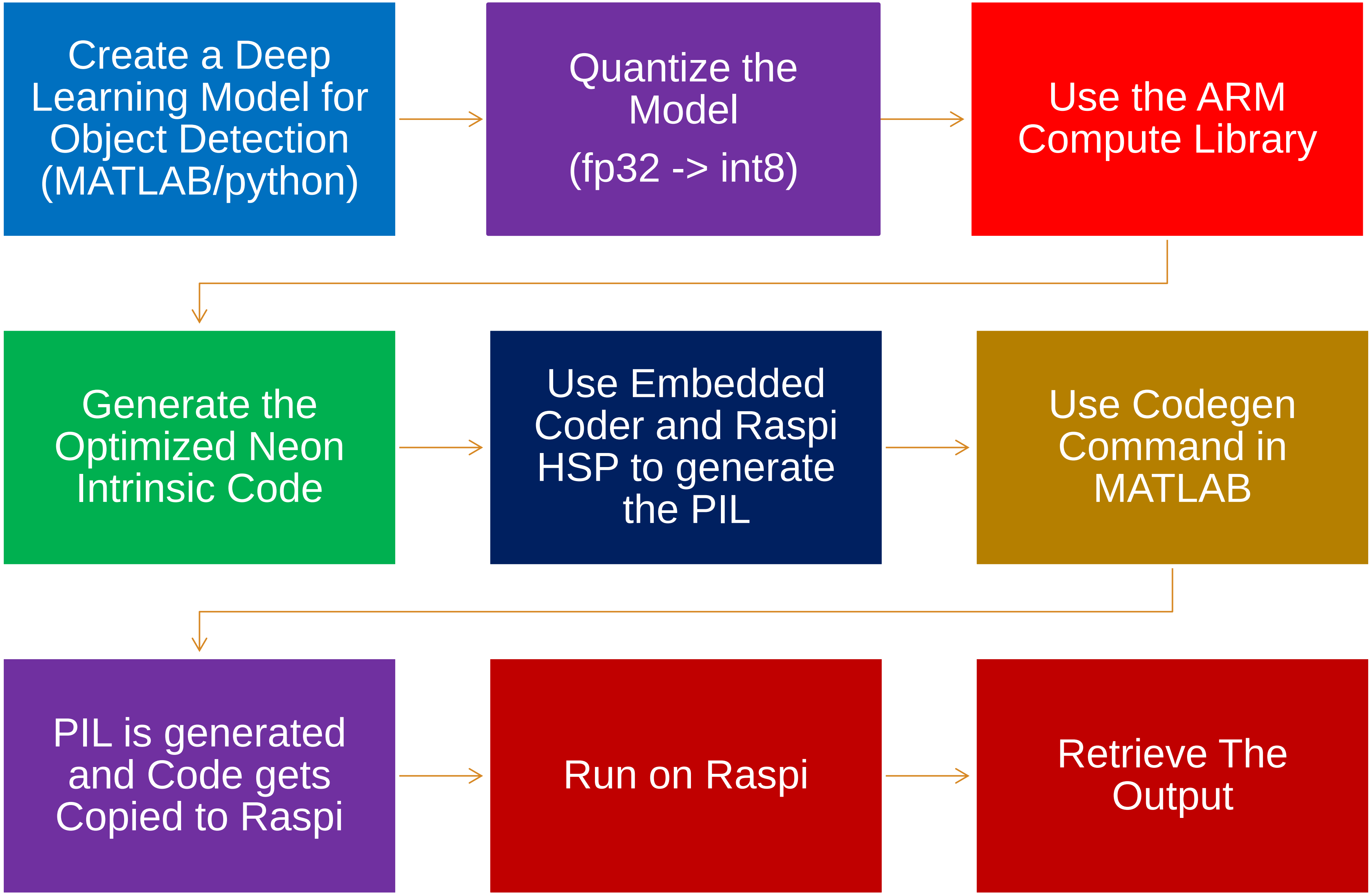
Use Codegen in MATLAB to generate optimized Code

- Use codegen in MATLAB to generate code for the inference of TFLite Model for pose-estimation network.
- This command generates PIL executable on MATLAB

Run the PIL and Retrieve the data

- Run the PIL on MATLAB which internally calls the executable generated on Raspi hardware which performs the pose-estimation.
- The output is retrieved back to MATLAB

PRODUCT FUNCTIONS



Create a Deep Learning Model for Object Detection (MATLAB/python)

Quantize the Model

(fp32 -> int8)

Use the ARM Compute Library

Generate the Optimized Neon Intrinsic Code

Use Embedded Coder and Raspi HSP to generate the PIL

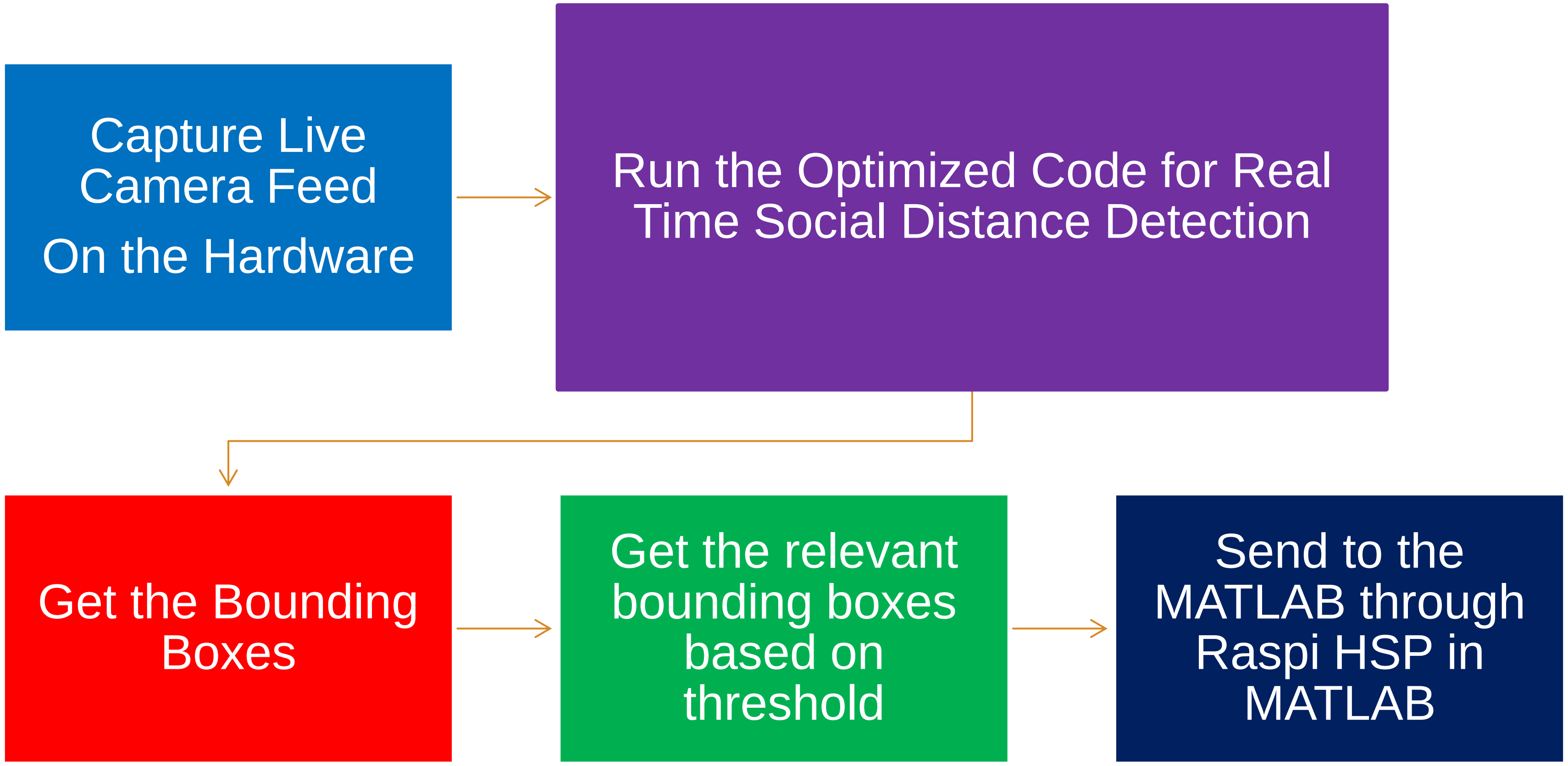
Use Codegen Command in MATLAB

PIL is generated and Code gets Copied to Raspi

Run on Raspi

Retrieve The Output

PRODUCT FUNCTIONS (Run on Raspi)



Capture Live Camera Feed

On the Hardware

Run the Optimized Code for Real Time Social Distance Detection

Get the Bounding Boxes

Get the relevant bounding boxes based on threshold

Send to the MATLAB through Raspi HSP in MATLAB

FEATURES

✓ Seamless process in creation and quantizing Network

Used dlquantizer (MATLAB) to convert the floating point model to quantized fixed point model more suited for embedded devices.

✓ Seamless Optimizations using ACL

Leverage ARM NEON intrinsics for inference of quantized TFLite Model

✓ Easy to generate Optimized code using Embedded Coder

With Optimized inference code along with pre and post processing, Any video can be used for pose estimation on embedded hardware

FEATURES

Reduced Manual Copying to hardware and retrieving the output

Using Raspi HSP to avoid manual copying to hardware and generates PIL which runs and retrieves output back to MATLAB.

Real Time Performance for Object Detection

.Real time performance (Better FPS for pose detection)

Seamless Deployment to Embedded Devices

Deploy Code to Raspi using HSP and Easy Run and Retrieve Data using PIL

Easy Visualize the Output Data

Use MATLAB tools to visualize the pose detection of live video/ existing video

MODULE DESCRIPTION

S.No	Module
1	Create DL Network
2	Quantize DL Network
3	Generate ACL optimizations
4	Use EC and Raspi HSP to Generate PIL
5	Use Codegen Command to Generate PIL
6	Visualize the Data

- Capture Video Feed.
- DL network is used to detect people in video feed.
- Use ACL optimizations to convert in bird eye view and identify distance
- Give bounding boxes of people not compying social media norms.

MODULE DESCRIPTION

Create DL Network:

- Create DL Network for Object detection using dlnetwork.

(or)

- Pretrained TFLite Models can be imported.

Quantize DL Network:

- Using dlquantizer, Models can be quantized from floating point to fixed point models..

Generate ACL optimizations:

- ARM provides ACL library which has set of ARM Compiler Assembly and Intrinsic SIMD NEON optimized instruction set.
- Using Codegen in MATLAB, we can leverage ACL optimizations
- Using TensorFlow Lite Library , we can also leverage Google TensorFlow Optimizations which can be targeted on Embedded Devices.

MODULE DESCRIPTION

Use EC and Raspi HSP to Generate PIL:

- HSP provides an easy way of communicating with hardware and easily retrieve the data back and forth.
- Embedded Coder (EC) generates optimized code for Embedded devices.
- Embedded Coder creates PIL interface which calls the executable on hardware and retrieves the output.

Use Codegen Command to Generate PIL:

- Codegen command generates optimized C code for MATLAB code targeted for hardware.
- It uses ACL library for NEON optimizations on ARM hardware.

Visualize the Data:

- Run the PIL to run executable
- Should allow the users to initiate an extraction process and view the progress

SYSTEM REQUIREMENTS

Processor	Intel Core i5 or more / AMD Ryzen 5 or more
RAM	16 GB Or more
Hard Disk	20 GB or more
Monitor	15” colour monitor or better
Keyboard	Any Keyboard
Mouse	Any mouse
Printer	In case of printing the graphs or filtered data
Network	Not needed
Software Needed	MATLAB, Embedded Coder Products Raspi HSP
Operating System	Windows / Linux / Mac
Licensing	MATLAB and Embedded Coder License and Deep Leaning Toolbox License

Host Requirements

Processor	Rasperry Pi 4
RAM	4 GB Or more
Hard Disk	SD card 32 GB
CPU	Quad-core ARM Cortex-A72 (64-bit) at 1.5 GHz
GPU	Broadcom VideoCore VI, OpenGL ES 3.0 support
Monitor	Optional
Camera	Raspberry Pi 5 MP camera module
Network	Needed while setup

Hardware Requirements

EXISTING SYSTEM

Manual Process

Manual Effort in Optimizing Code

- Manually Optimize using ARM Assembly Intrinsics
- Write a script for Quantizing Model

Manual Copying To Hardware

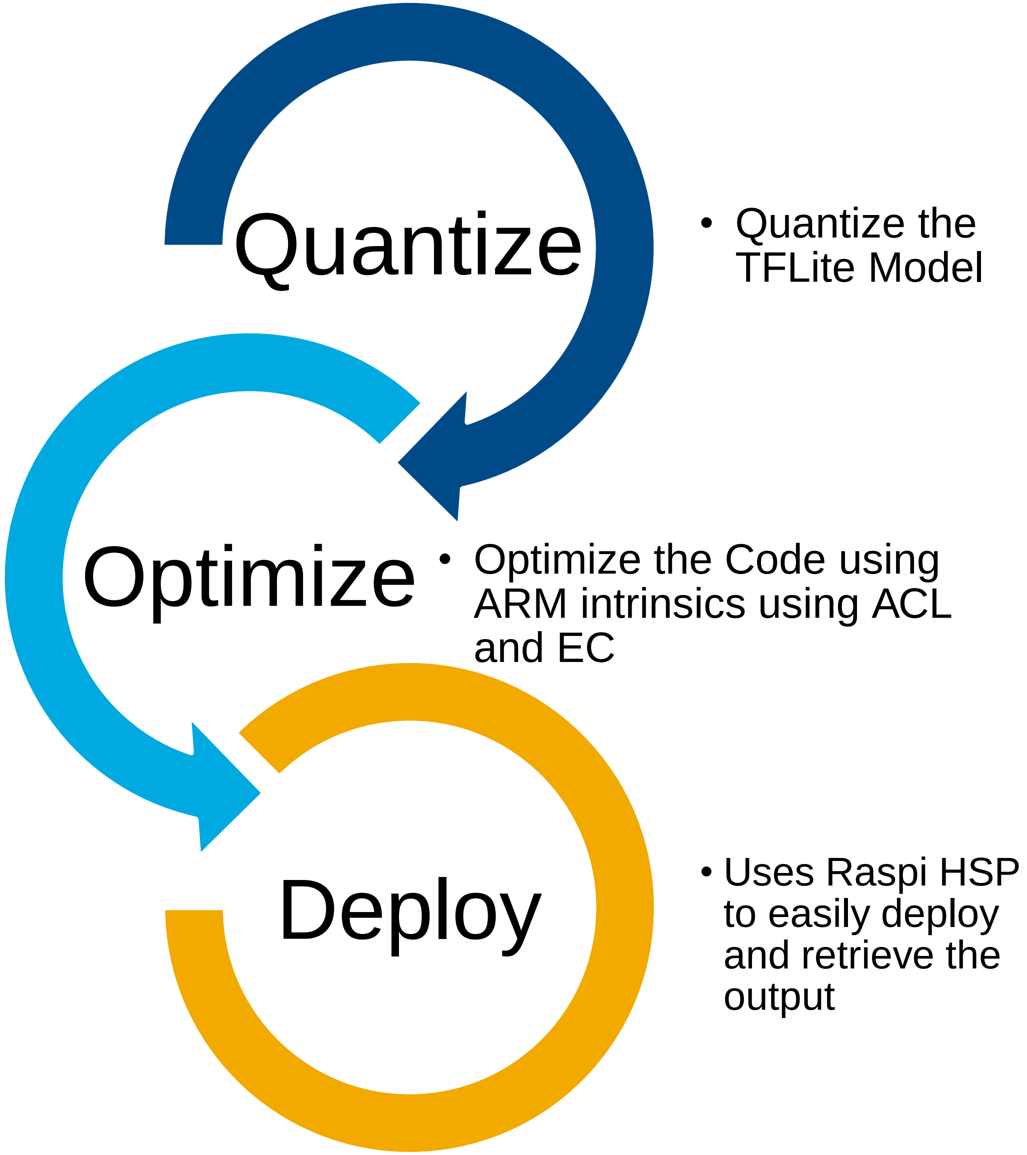
- Have to manual copy code to hardware
- Retrieve output is difficult and have to manually handle output store and copying

Difficult to Deploy and Achieve Realtime Performance

- Create a Makefile manually tp create EXE on the hardware

- Human error prone process
- Need to handle optimizations, deployment and codegen process.

PROPOSED SYSTEM

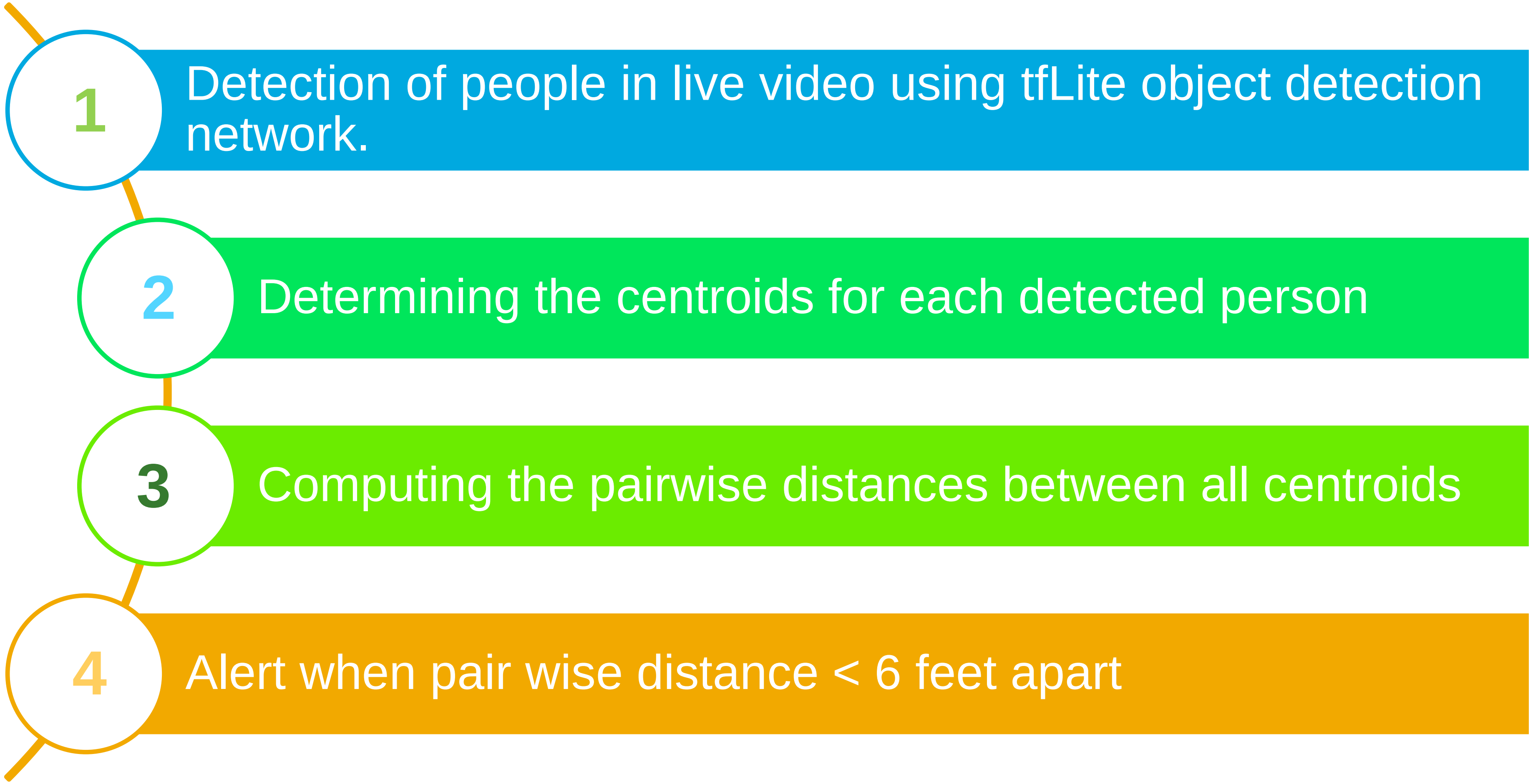


- Quantize the TFLite Model
- Optimize the Code using ARM intrinsics using ACL and EC
- Uses Raspi HSP to easily deploy and retrieve the output

Fully Automated Process

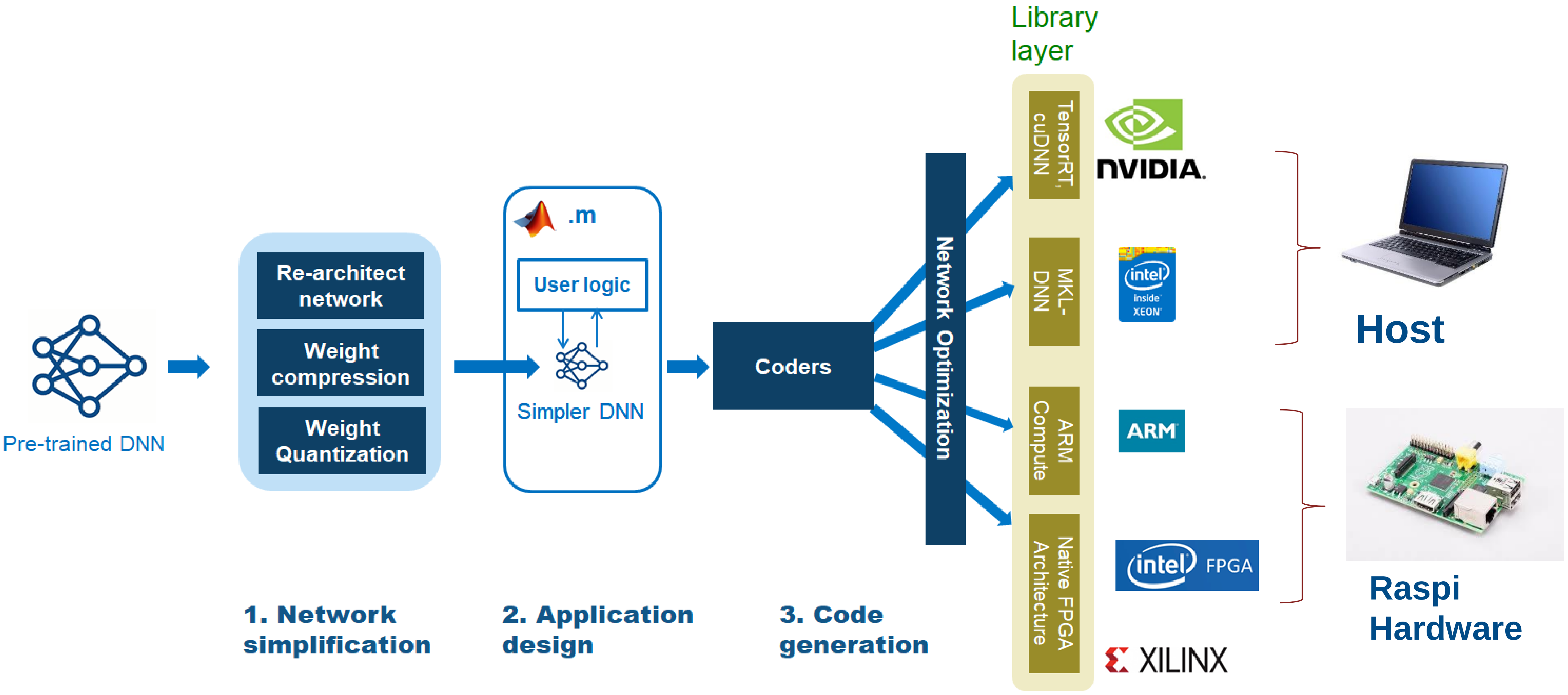
- A fully automated software process will Quantize the model
- Capture data or use pre captured data
- Use EC to Optimize the Code leveraging ACL library
- Use HSP to deploy the Network Inference along with pre and post processing.
- Easily retrieve the output along with post processing to host.

Social distance detection system workflow



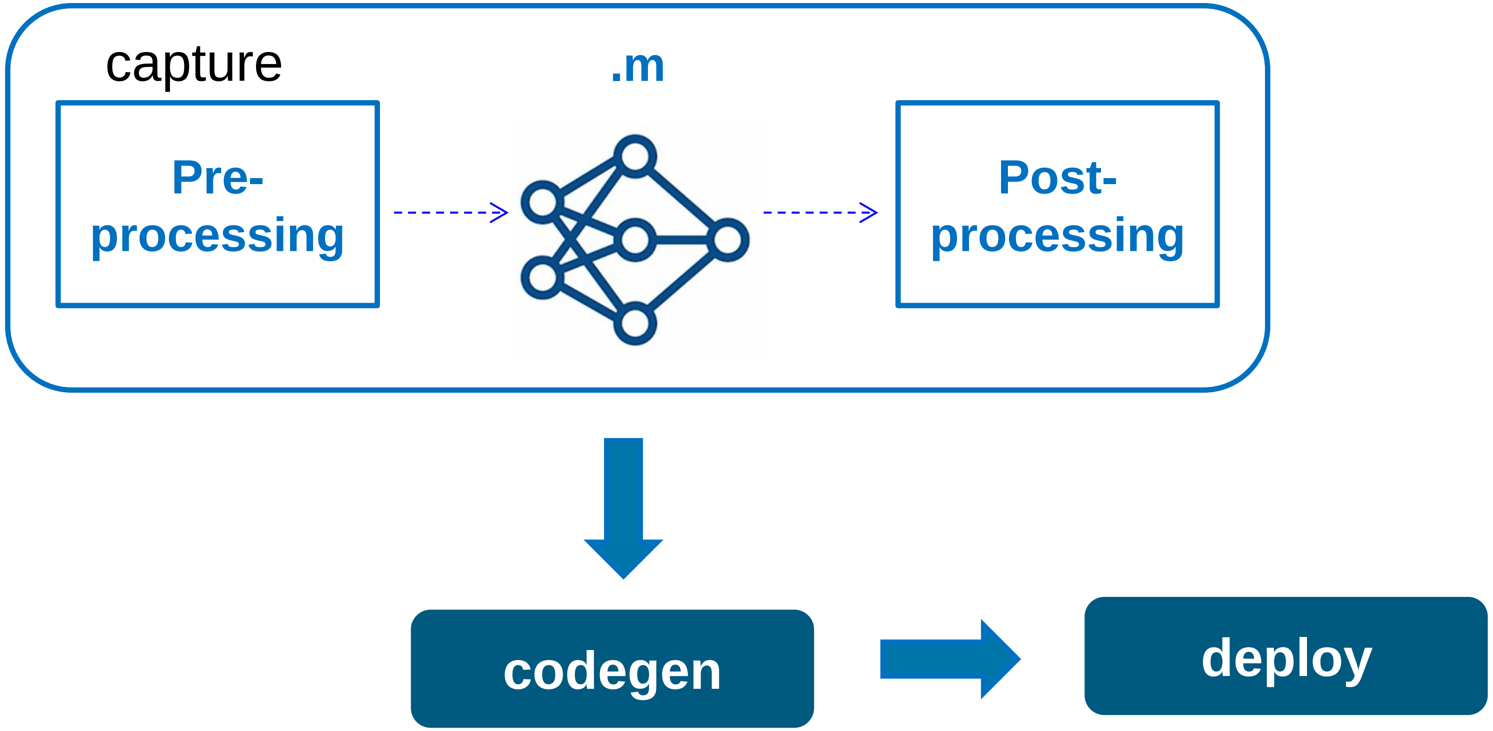
1. Detection of people in live video using tfLite object detection network.
2. Determining the centroids for each detected person
3. Computing the pairwise distances between all centroids
4. Alert when pair wise distance < 6 feet apart

DATA FLOW DIAGRAM (DFD)



Seamless Workflow

Whole application with
trained DL network



capture

Pre-processing

.m

Post-processing

codegen

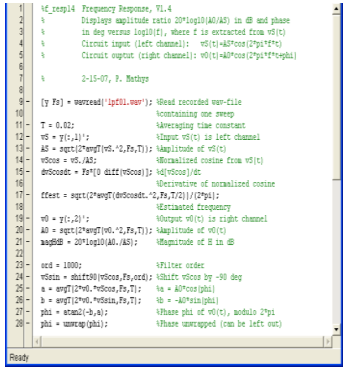
deploy

Raspi
Hardware

Sequence Diagram

MATLAB

Functional algorithm

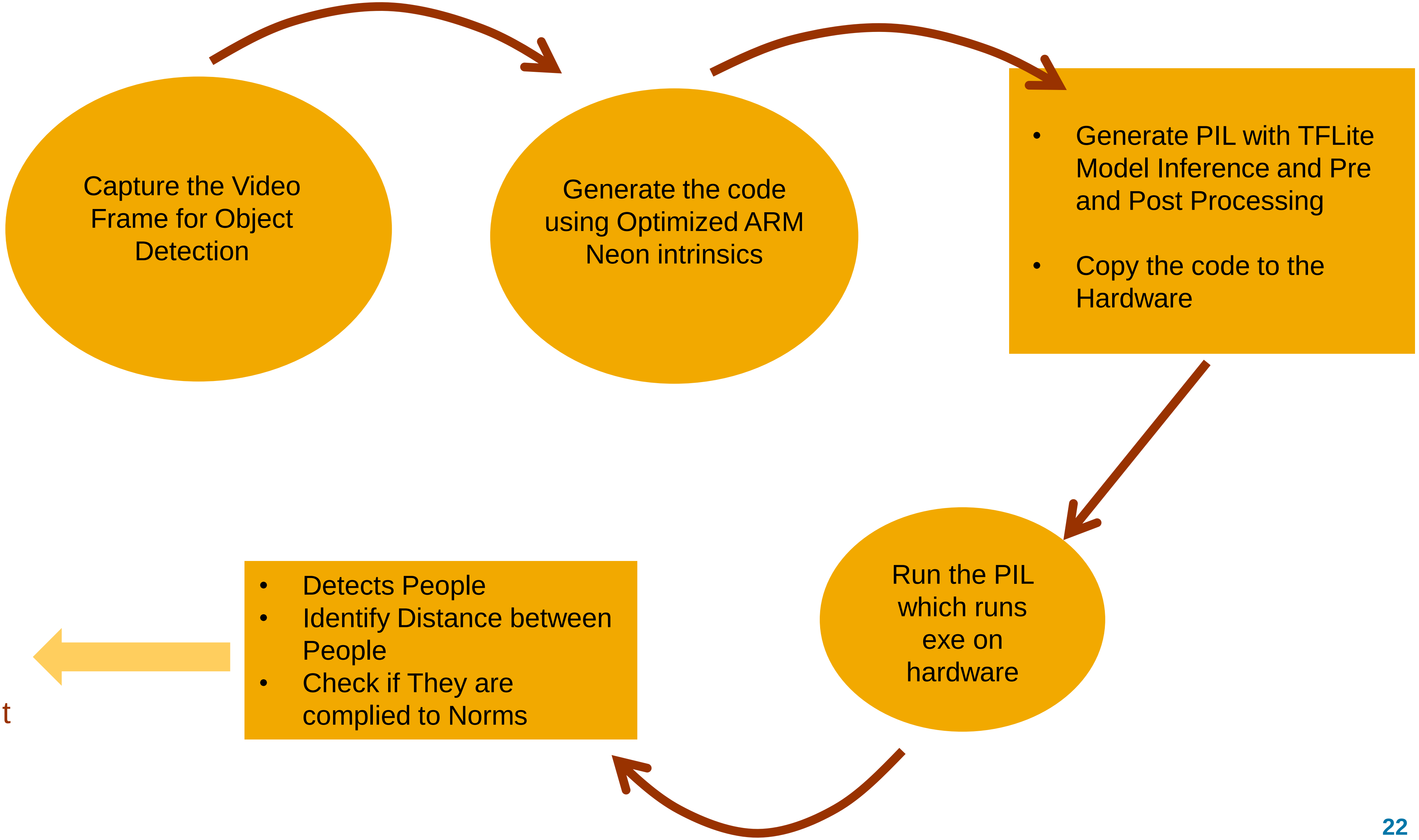


```
%f_resp14  Frequency Response, V1.4
%          Displays amplitude ratio 20*log10(A0/AS) in dB and phase
%          in deg versus log10(f), where f is extracted from vS(t)
%          Circuit input (left channel):   vS(t)=AS*cos(2*pi*f*t)
%          Circuit ouptut (right channel): v0(t)=A0*cos(2*pi*f*t+phi)

%          2-15-07, P. Mathys

[y Fs] = wavread('lpf01.wav'); %Read recorded wav-file
                               %containing one sweep
T = 0.02;                      %Averaging time constant
vS = y(:,1)';                  %Input vS(t) is left channel
AS = sqrt(2*avgT(vS.^2,Fs,T)); %Amplitude of vS(t)
vScos = vS./AS;                %Normalized cosine from vS(t)
dvScosdt = Fs*[0 diff(vScos)]; %d[vScos]/dt
                               %Derivative of normalized cosine
ffest = sqrt(2*avgT(dvScosdt.^2,Fs,T/2))/(2*pi);
                               %Estimated frequency
v0 = y(:,2)';                  %Output v0(t) is right channel
A0 = sqrt(2*avgT(v0.^2,Fs,T)); %Amplitude of v0(t)
magHdB = 20*log10(A0./AS);     %Magnitude of H in dB

ord = 1000;                    %Filter order
vSsin = shift90(vScos,Fs,ord); %Shift vScos by -90 deg
a = avgT(2*v0.*vScos,Fs,T);    %a = A0*cos(phi)
b = avgT(2*v0.*vSsin,Fs,T);    %b = -A0*sin(phi)
phi = atan2(-b,a);             %Phase phi of v0(t), modulo 2*pi
phi = unwrap(phi);             %Phase unwrapped (can be left out)
```



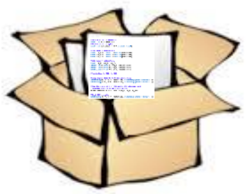
Capture the Video Frame for Object Detection

Generate the code using Optimized ARM Neon intrinsics

- Generate PIL with TFLite Model Inference and Pre and Post Processing
- Copy the code to the Hardware

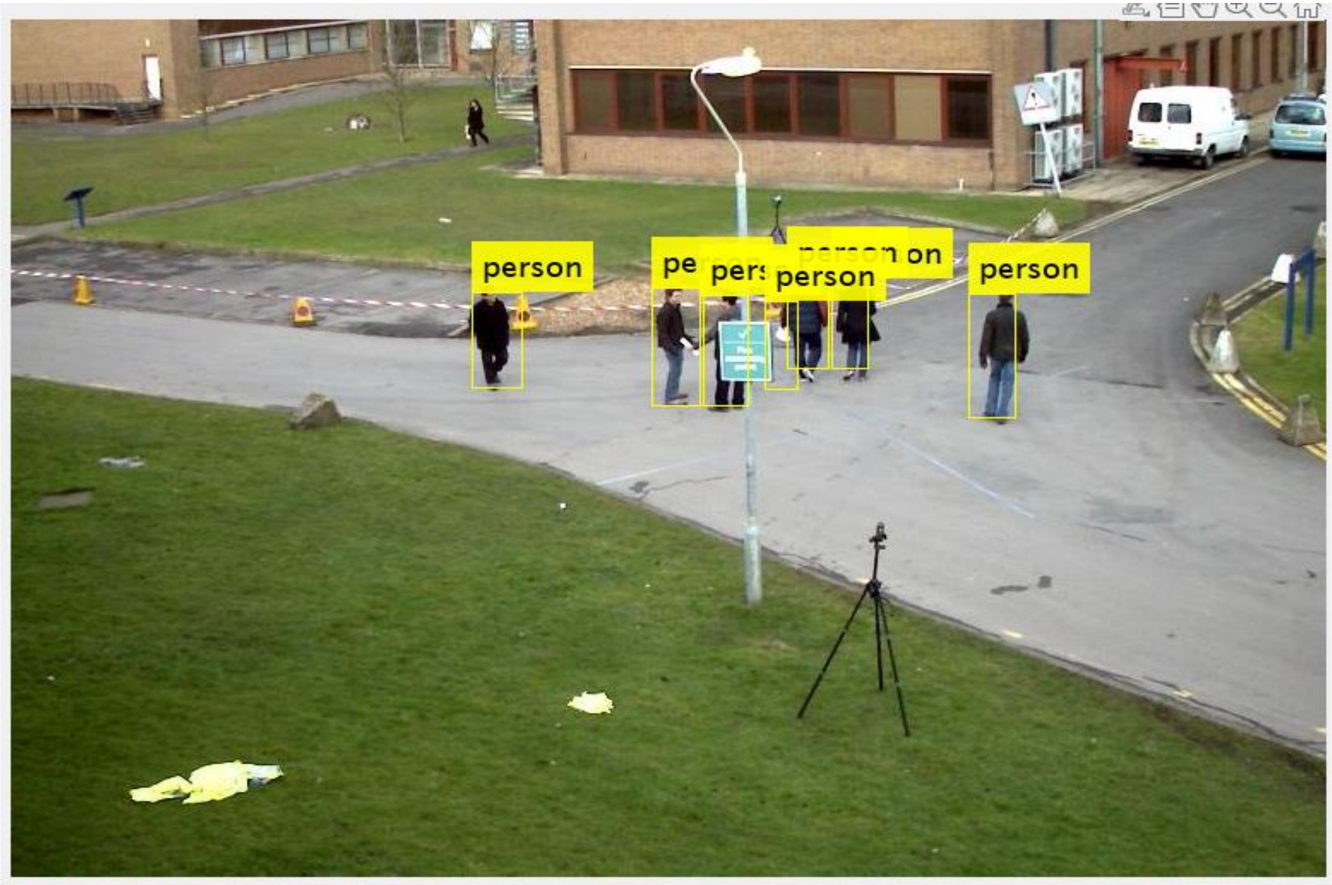
Run the PIL which runs exe on hardware

- Detects People
- Identify Distance between People
- Check if They are complied to Norms



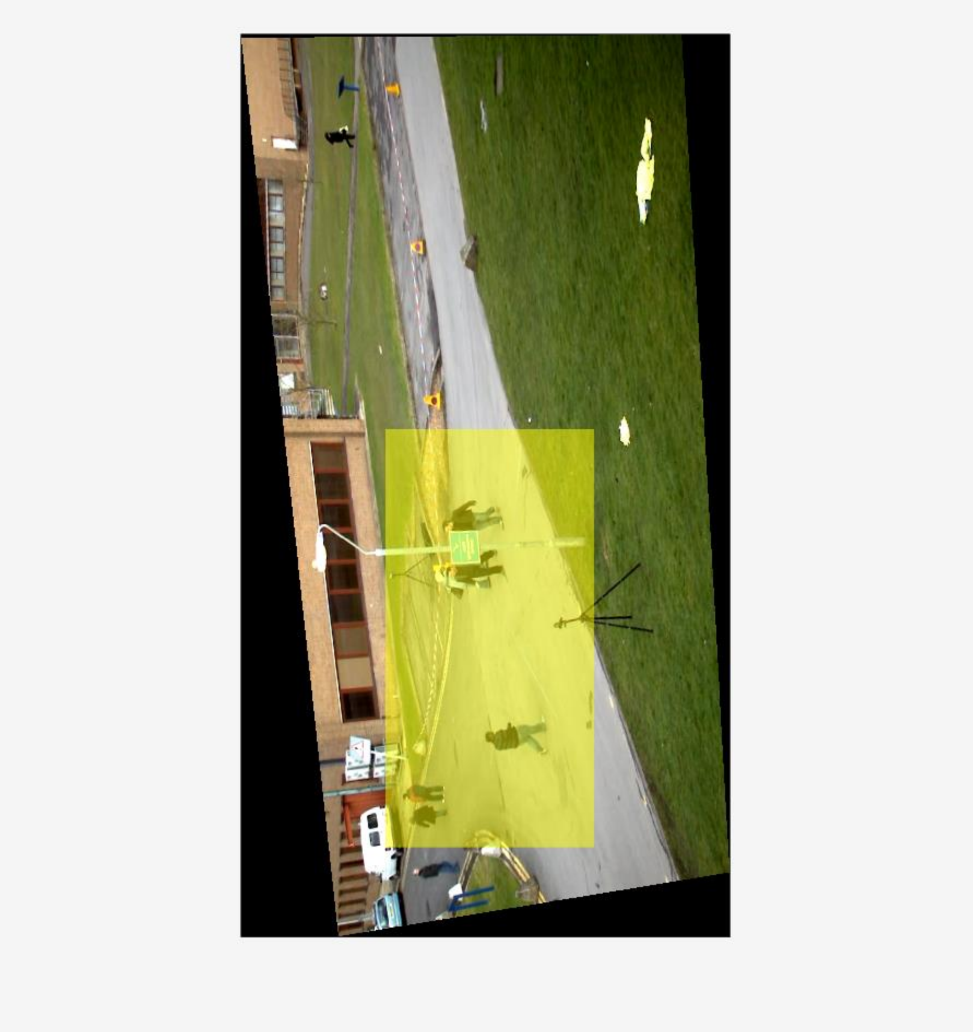
Retrieve Output Back to Host

Output Screenshots

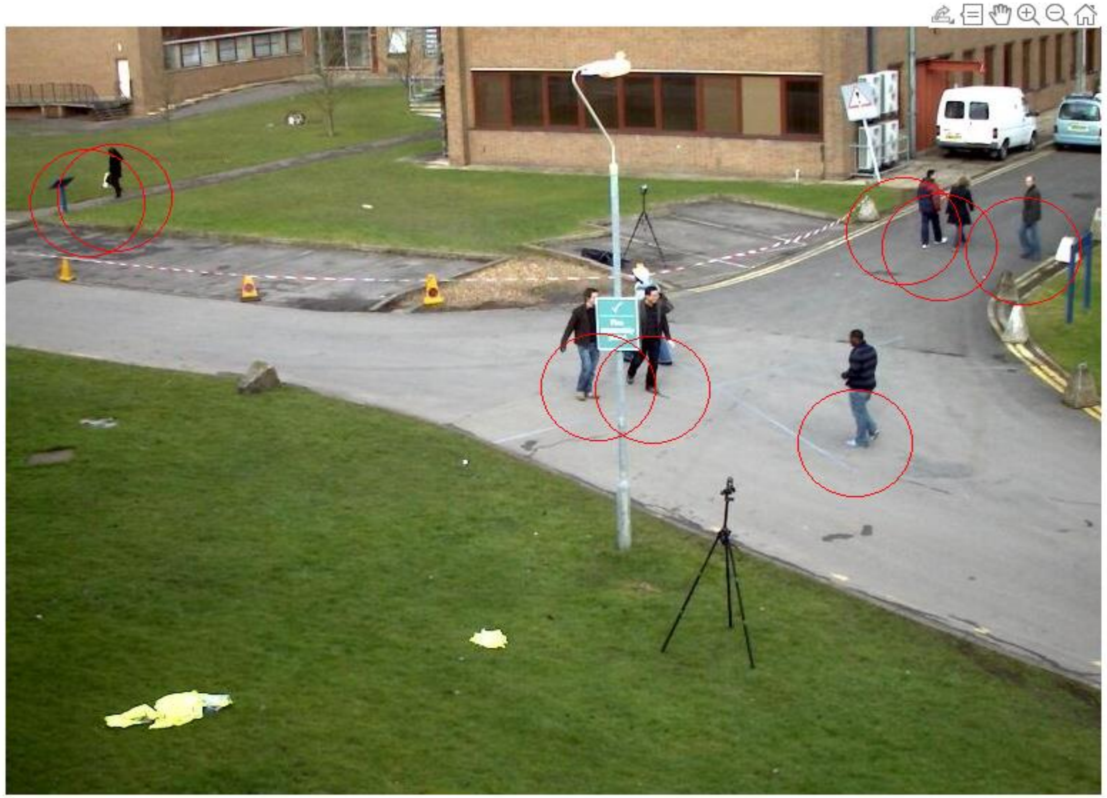


Step 1- Detection of people using Quantized Object detection Model on Raspi HW

Output Screenshots

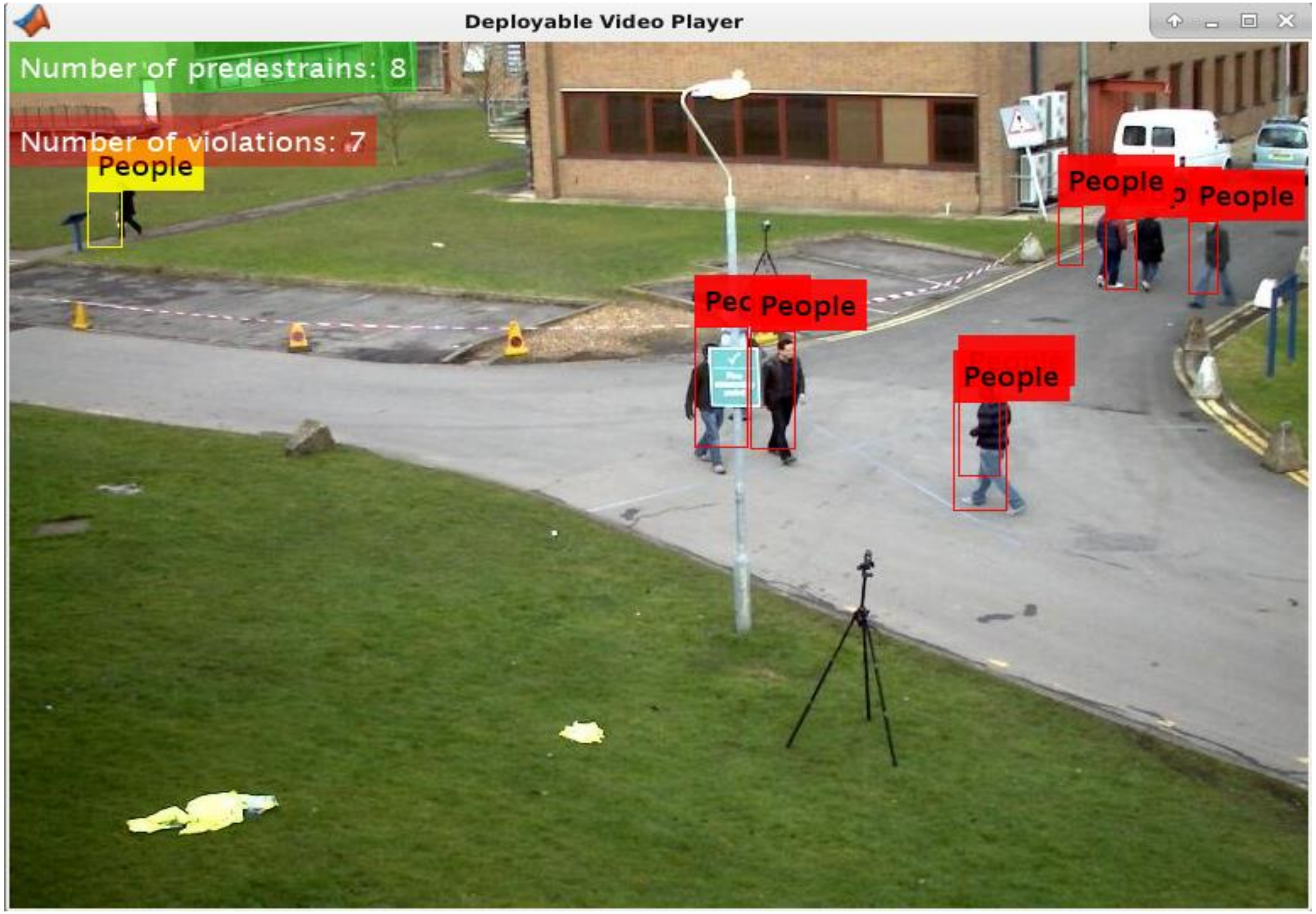


Step 2- Convert to Bird Eye View



Step 2- Determining the centroids for each detected person using Bird Eye View

Output Screenshots

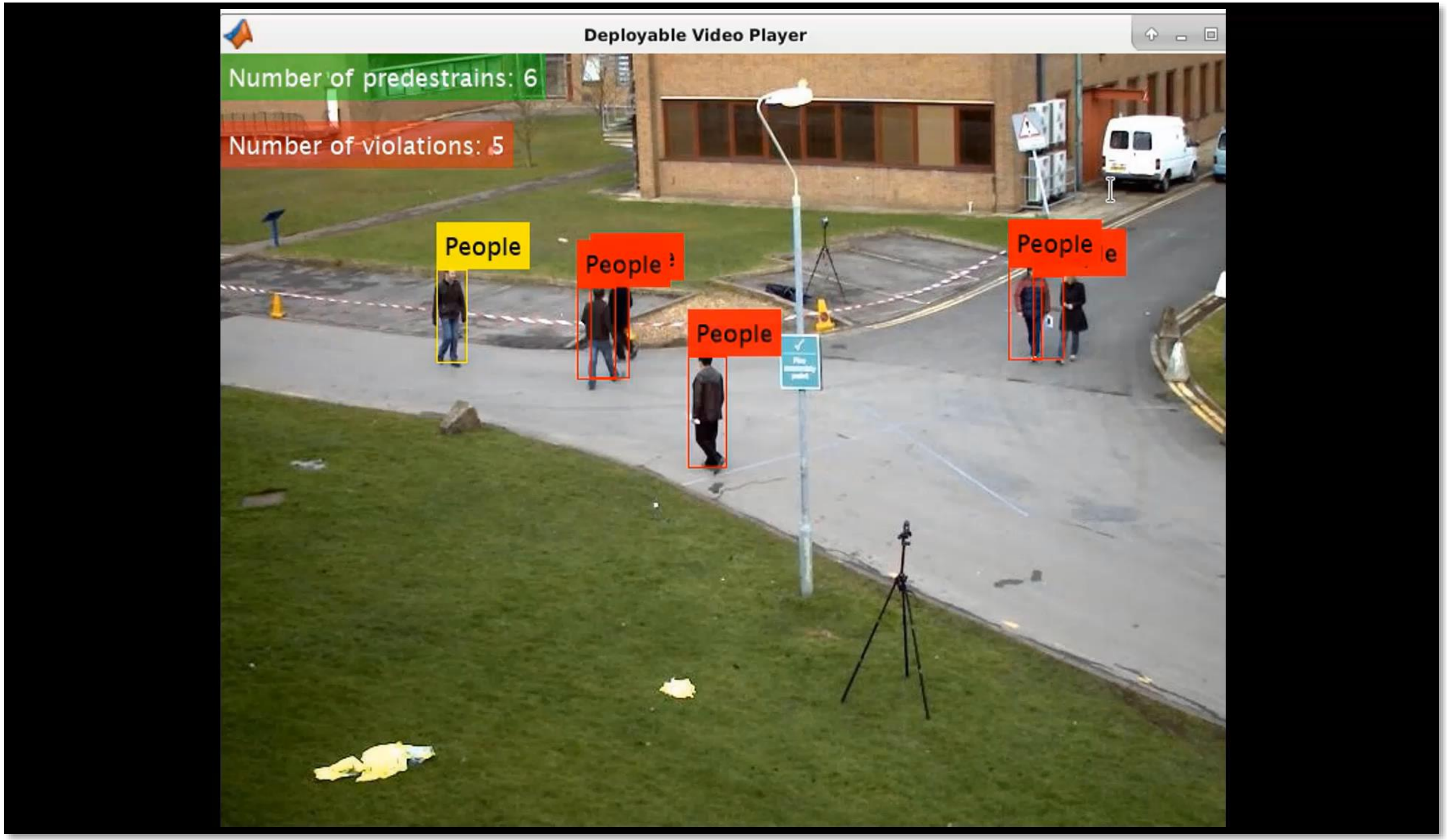


Number of predestrains: 8

Number of violations: 7

Step 3- Computing the pairwise distances between all centroids and mark as red when distance < 6feet

Output Screenshots - Demo



Deployable Video Player

Number of predestrains: 6

Number of violations: 5

SOURCE CODE – GITHUB LOCATION

The Complete Source code is found in the following GitHub location:

https://github.com/vyshnavijayanthiMBAHR/MBA_4thSemProject_VyshnaviHR_SRM.git

CONCLUSION

Summary of Findings

- Successfully optimized deep learning models for real-time object detection on embedded devices.
- Achieved efficient deployment on Raspberry Pi, focusing on social distancing applications.
- Enhanced system performance through model pruning, quantization, and library integration.

Benefits of the Approach

- **Efficiency and Accuracy:** Improved detection accuracy and processing speed in resource-constrained environments.
- **Real-Time Capability:** Enabled immediate data processing and decision-making with edge computing strategies.

Implications for Real-Time Object Detection

- Develop automated systems for continuous monitoring and reporting.
- Facilitate strategic decision-making with real-time insights and performance benchmarks.
- Enhance operational efficiency in diverse fields such as public safety and autonomous systems.

.

FUTURE ENHANCEMENTS

Advanced Model Optimization

- Explore cutting-edge optimization techniques like neural architecture search to further improve model efficiency and accuracy.

Broader Device Support

- Extend deployment capabilities to a wider range of embedded devices and platforms, enhancing versatility and scalability

Enhanced Edge Computing:

- Develop more sophisticated edge computing strategies to further reduce latency and improve real-time processing capabilities.

Comprehensive Testing:

- Conduct extensive field testing to validate system performance in diverse environmental conditions and scenarios.

REFERENCES

Libraries

- **Intel MKL-DNN.** Documentation and resources available at: Intel MKL-DNN
- **NVIDIA cuDNN.** Documentation and resources available at: NVIDIA cuDNN
- **ARM Compute Library.** Documentation and resources available at: ARM Compute Library
- **TensorFlow Lite** Documentation and resources available at: TensorFlow Lite

Books and Publications

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

  A comprehensive resource on the foundations and advancements in deep learning.

- Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.

  Discusses the YOLO (You Only Look Once) model, a popular choice for real-time object detection.

Technology references

https://www.mathworks.com/products/matlab.html
https://www.tableau.com/learn/articles/data-visualization
Raspberry Pi Documentation
TensorFlow Lite

Complete Source code is found in the following GitHub location: https://github.com/vyshnavijayanthiMBAHR/MBA_4thSemProject_VyshnaviHR_SRM.git

THANK YOU