REAL-TIME IMAGE CLASSIFICATION AND OBJECT DETECTION WITH TENSORFLOW LITE ON RASPBERRY PI 4B FOR SURVEILLANCE TECHNOLOGY

BY

Neeraj Nikhil Roy (20/11/EC/053)
Kshama Meena (20/11/EC/055)
Komal Kesav Nenavath (20/11/EC/012)
Divyansh Singh (20/11/EC/057)

under the guidance of
Dr. Ankit Kumar Jaiswal

in the partial fulfillment of the requirements
for the award of the degree of

Bachelor of Technology
(a part of Five-Year Dual Degree Course)

School of Engineering
Jawaharlal Nehru University, New Delhi
November, 2023

JAWAHARLAL NEHRU UNIVERSITY
SCHOOL OF ENGINEERING

DECLARATION

We declare that the project work entitled **"REAL-TIME IMAGE CLASSIFICATION AND OBJECT DETECTION WITH TENSORFLOW LITE ON RASPBERRY PI FOR SURVEILLANCE TECHNOLOGY"** which is submitted by me/us in partial fulfillment of the requirement for the award of degree B.Tech. (a part of Dual-Degree Programme) to School of Engineering, Jawaharlal Nehru University, Delhi comprises only our original work and due acknowledgement has been made in the text to all other material used.

[illegible]

Neeraj Nikhil Roy (20/11/EC/053)

क्षमा मीना

Kshama Meena (20/11/EC/055)

[illegible]

Komal Kesav Nenavath (20/11/EC/012)

[illegible]

Divyansh Singh (20/11/EC/057)

JAWAHARLAL NEHRU UNIVERSITY
SCHOOL OF ENGINEERING

CERTIFICATE

This is to certify that the synopsis entitled "**REAL-TIME IMAGE CLASSIFICATION AND OBJECT DETECTION WITH TENSORFLOW LITE ON RASPBERRY PI FOR SURVEILLANCE TECHNOLOGY**" being submitted by **Mr. Neeraj Nikhil Roy (20/11/EC/053)**, **Ms. Kshama Meena (20/11/EC/055)**, **Mr. Komal Kesav Nenavath (20/11/EC/012)**, and **Mr. Divyansh Singh (20/11/EC/057)**, in fulfilment of the requirements for the award of the **Bachelor of Technology** (part of Five-Year Dual Degree Course) in **Computer Science & Engineering**, will be carried out by him under my supervision.

In my opinion, this work fulfils all the requirements of an Engineering Degree in respective stream as per the regulations of the School of Engineering, Jawaharlal Nehru University, Delhi. This synopsis does not contain any work, which has been previously submitted for the award of any other degree.

[illegible] 25/11/2023

Dr. Ankit Kumar Jaiswal
(Supervisor)
Assistant Professor
School of Engineering
Jawaharlal Nehru University, New Delhi

JAWAHARLAL NEHRU UNIVERSITY

SCHOOL OF ENGINEERING

ACKNOWLEDGMENT

We are very thankful and extend our deepest gratitude to the members of our project team for their unwavering commitment and collaborative efforts throughout the completion of our B.Tech semester project. This endeavour would not have been possible without the unique contributions and dedication of each team member.

This project has been a collective effort, and we are proud to have worked with such a talented and dedicated group of individuals. Together, we navigated challenges, celebrated successes, and ultimately created a project we can all be proud of.

Thank you, team, for your hard work, commitment, and the invaluable experience of working together. This project stands as a testament to our collective abilities and the strength of our collaboration.

Sincerely,

[illegible]

Neeraj Nikhil Roy (20/11/EC/053)

क्षमा मीना

Kshama Meena (20/11/EC/055)

[illegible]

Komal Kesav Nenavath (20/11/EC/012)

[illegible]

Divyansh Singh (20/11/EC/057)

LIST OF CONTENTS

Contents

LIST OF FIGURES AND TABLES

ABSTRACT

Real-Time Image Classification and Object Detection with TensorFlow Lite on Raspberry Pi 4b for Surveillance Technology

This project focuses on the implementation of a real-time image classification and object detection system using TensorFlow Lite on the Raspberry Pi 4b platform. With the increasing demand for robust surveillance technology, the integration of machine learning models for accurate image analysis is imperative. Leveraging the efficiency of TensorFlow Lite, our system achieves rapid and accurate classification of images as well as real-time detection of objects.

The project employs a Raspberry Pi 4b as the edge computing device, ensuring a compact and energy-efficient solution for deployment in surveillance applications. The TensorFlow Lite framework allows for the optimization of deep learning models, enabling them to run seamlessly on resource-constrained devices like the Raspberry Pi.

Key features of the project include a streamlined image classification process that accurately identifies objects in real time and an object detection module capable of identifying and tracking multiple objects simultaneously. The system's versatility and scalability make it suitable for various surveillance scenarios, from home security to industrial monitoring.

This project not only showcases the capabilities of TensorFlow Lite in edge computing but also addresses the practical implementation of machine learning for enhancing surveillance technology. The results demonstrate the feasibility and efficiency of utilizing lightweight models on edge devices for real-time image analysis, contributing to the advancement of intelligent surveillance system.

Chapter 1
INTRODUCTION and THESIS

1.1 INTRODUCTION

In recent years, the integration of machine learning and edge computing has revolutionized the field of surveillance technology. The ability to perform real-time image classification and object detection is becoming increasingly vital for enhancing security measures in various domains. This project delves into the development of a robust system using TensorFlow Lite on the Raspberry Pi 4b platform, aiming to deliver efficient, cost-effective, and accurate surveillance capabilities.

1.1.1 Background

CCTV (or closed-circuit television) was first used in World War 2 when German scientists created a camera inside a box to observe the launches of A4 rockets safely.

It's been more than 75 years since its inception, yet they are still primarily used for surveillance and crime detention. The world has changed significantly since then with innovations like the internet and smartphones. However, not much has changed regarding CCTV architecture and utility. **Almost all cameras today are used to record footage that has to be later analyzed by humans if something happens**. Not only is this process very slow, but it's costly and requires human attention and massive amounts of data storage.

The evolution of surveillance technology has witnessed a paradigm shift in recent years, moving from conventional methods to the integration of cutting-edge machine learning techniques. The ability to process and analyze visual data in real-time has become a cornerstone in the domain of security and surveillance. As the demand for intelligent surveillance solutions rises, this project endeavors to contribute to the field by developing a robust system for real-time image classification and object detection.

1.1.2. Motivation

The motivation behind this project stems from the limitations of traditional surveillance systems. Since the 1970s, the CCTV camera system's technologies have stayed the same. Usually, a person or maybe two sits behind a huge grid of screens and manually observes/keeps watch. This not only introduces human error and negligence into the system but also a lot of cost in maintaining the system on top of resources consumed.

Conventional methods often face challenges related to scalability, resource consumption, and adaptability. Considering recent developments in machine learning and camera quality, a lot will change soon. Instead of reacting to events after they have occurred, we expect hardware and software capabilities to reach a point where cameras automatically detect and notify people about incidents and important events within their range. The emergence of edge computing and the power of TensorFlow Lite presents an exciting opportunity to address these challenges. By harnessing the potential of edge devices, particularly the Raspberry Pi 4b, this project aims to create an efficient and cost-effective solution for surveillance technology.

Where from as early as 1970, the technologies have stayed the same

1.1.3. Objectives

The primary objectives of this project are threefold:

- **Real-time Image Classification:** Implementing a system capable of accurately classifying images in real-time using TensorFlow Lite.
- **Object Detection Module:** Developing a robust object detection module that can identify and track multiple objects concurrently.
- **Raspberry Pi 4b Deployment:** Deploying the entire system on the Raspberry Pi 4b, thereby harnessing the advantages of edge computing for on-device processing.

1.1.4. Scope

This project's scope extends to various surveillance applications, including but not limited to home security, industrial monitoring, and public safety. The flexibility and adaptability of the system make it suitable for deployment in diverse indoor and outdoor environments. These cameras aren't just a security tool. They can be designed to assist a variety of business use cases and be a valuable tool in sectors such as hospitals, banks, offices, and transport; the possibilities are endless.

1.2 THESIS OBJECTIVE

This thesis aims to showcase the practical implementation and effectiveness of TensorFlow Lite on the Raspberry Pi 4b for real-time image classification and object detection in surveillance applications.

The contributions of this thesis are noteworthy:

- **Streamlined Image Classification:** Implementation of a highly efficient image classification process leveraging TensorFlow Lite, ensuring real-time performance.
- **Object Detection Module:** Development of a sophisticated object detection module capable of identifying and tracking multiple objects concurrently, adding a layer of complexity to the surveillance system.
- **Performance Evaluation:** Rigorous evaluation of the system's performance on the Raspberry Pi 4b, demonstrating its capabilities and limitations in real-world scenarios.
- **Setting up of Cameras:** How cameras could be configured according to the need considering the fact that performance is at stake.

1.3. ORGANIZATIONS OF CHAPTERS

The thesis consists of five chapters, and the overview of all the chapter are as follows:

Chapter 1: This chapter provides a brief introduction to the background scope of the project, setting the stage for the subsequent chapters and the objectives of the thesis involved in accomplishing the thesis.

Chapter 2: This chapter gives an overview of available literature. In this chapter, a comprehensive review of existing work and technologies in the realms of real-time image classification and object detection is presented. This includes an analysis of relevant literature, methodologies, and critical findings.

Chapter 3 Describes the details of the proposed work and the methodology used in this research work. Detailed explanations of the proposed system's architecture, components, and the methods employed for implementation are expounded upon in this chapter. Technical details and considerations are thoroughly discussed.

Chapter 4: This chapter discusses the simulated and measured results. This chapter presents and critically analyzes the results obtained from experimental evaluations. Insights into the system's performance, strengths, and improvement areas are discussed in detail.

Chapter 5: Presents the conclusion of this research work and the future scope of the work. This final chapter summarizes the project's findings, draws conclusions based on the results, and outlines potential avenues for future research and development. Implications of the project's outcomes and their broader significance are also discussed.

References: The references used are given in the last section of this thesis.

Chapter 2
LITERATURE SURVEY

2.1. INTRODUCTION

The landscape of real-time image classification and object detection within the domain of surveillance technology has witnessed a remarkable evolution driven by advancements in machine learning, computer vision, and edge computing. A thorough examination of existing literature is paramount to understanding the historical context, technological milestones, and current state-of-the-art methodologies. This literature survey serves as a comprehensive exploration of the rich tapestry of research and developments in the field, aiming to provide a solid foundation for the proposed work in this thesis.

2.2. SIGNIFICANCE OF THE LITERATURE SURVEY

The significance of conducting a literature survey lies in its ability to uncover key insights, methodologies, and challenges encountered by researchers and practitioners in the realm of real-time image classification and object detection. By examining past and current contributions, this survey seeks to identify gaps in knowledge, highlight successful implementations, and discern the overarching trends that have shaped the trajectory of research in this dynamic field
.

2.3. HISTORICAL EVOLUTION

This survey begins with an exploration of the historical evolution of surveillance technology, tracing the trajectory from traditional methods to the incorporation of machine learning. This historical context aids in understanding the motivations behind the shift towards intelligent surveillance systems and sets the stage for the exploration of contemporary methodologies.

The first documented CCTV camera to ever exist dates from 1942. Since it wasn't able to record and store data, it required constant monitoring.

Figure 2.3.1: First analog CCTV systems from 1942

Figure 2.3.2: VCR analog CCTV cameras from 1970s

VCR (videocassette recorders) were widely implemented in the 1970s. Images were stored in tapes that lasted for about 8 hours of recording and had to be manually replaced.

Hybrid CCTV systems; In the mid 90's the DVR (digital video recording) was introduced. This system digitized and compressed the video, storing information in a hard disk.

Network based DVR; Later on, DVRs were equipped with an Ethernet port for network connectivity. This allowed for remote video monitoring using a computer.

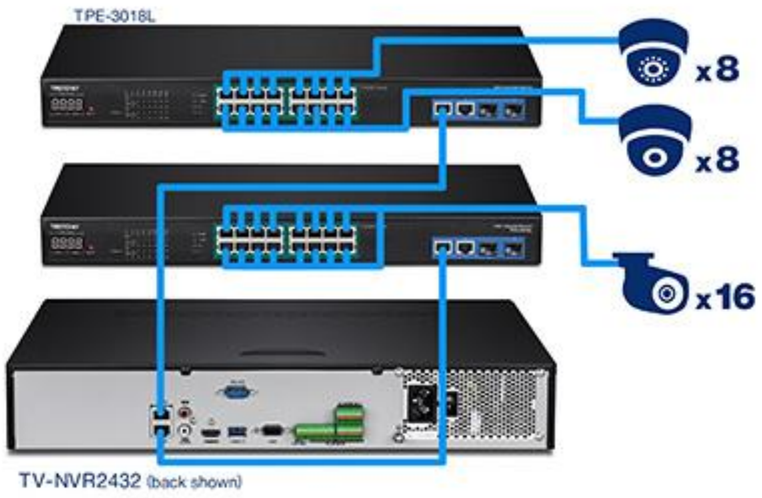


Figure 2.3.3: Network based DVR

Video encoders (also called video servers) were the next big step in the technology. The most important innovation is that with this system, video management is operated through software installed on a computer.

Figure 2.3.4: Video encoders

Fully digital IP CCTV cameras; This system is fully digital and doesn't include any analog component. That has significantly simplified the installation and maintenance of the system. It has computer power built in, allowing for the use of preinstalled applications in the camera, and they can synchronize with other devices.

Figure 2.3.4: Fully digital IP CCTVcameras

2.4. STATE-OF-THE-ART TECHNOLOGIES

A thorough investigation into state-of-the-art technologies and methodologies employed in real-time image classification and object detection is a pivotal aspect of this literature survey. This includes a detailed examination of deep learning architectures, convolutional neural networks (CNNs), and other machine learning models that have demonstrated prowess in image analysis. Moreover, attention is given to edge computing platforms, specifically the Raspberry Pi, to understand their role in bringing intelligence to the edge.

2.4.1. Camera Technologies

Camera technologies have undergone significant advancements, marked by a relentless pursuit of higher-resolution imaging. Both professional and consumer cameras, including smartphones, now boast resolutions in the tens of megapixels. Notable improvements include enhanced low-light performance through sophisticated sensor technology and image processing. Artificial intelligence (AI) has become integral, contributing to features like scene recognition and computational photography. Multi-camera setups in smartphones offer diverse lenses for various shooting scenarios. Video recording capabilities have scaled up to support 4K and 8K resolutions. Cameras are increasingly integrated with AI, IoT, and augmented reality (AR), enabling applications in security, surveillance, and immersive experiences. Autofocus technologies continue to evolve, ensuring faster and more accurate focusing. Compact and lightweight designs, especially in mirrorless cameras, cater to user preferences for portability without compromising performance. Drones and action cameras showcase innovations in stabilization and intelligent flight modes. Quantum dot sensors, a nascent technology, hold promise for improved color accuracy and low-light performance. These developments collectively reflect the dynamic landscape of modern camera technologies, contributing to more versatile and capable imaging devices across various domains.

2.4.2. Machine Learning, Deep Learning, Artificial Intelligence

Machine Learning (ML), Deep Learning (DL), and Artificial Intelligence (AI) form a trinity of interconnected technologies, each contributing to the evolution of intelligent systems. At its core, machine learning empowers computers to learn patterns and make decisions without explicit programming. It encompasses various algorithms, from classical ones like decision trees to advanced techniques such as support vector machines and random forests.

Deep Learning, a subset of ML, has gained prominence for its ability to mimic the human brain's neural networks. Deep neural networks, with multiple layers (hence "deep"), excel at tasks like image and speech recognition. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are prominent architectures within deep Learning, enabling breakthroughs in computer vision, natural language processing, and other complex tasks.

Artificial Intelligence, the overarching concept, involves creating machines or systems that can perform tasks that typically require human intelligence. AI encompasses both ML and DL, along with other approaches. AI's applications are vast, ranging from virtual assistants and recommendation systems to autonomous vehicles and advanced robotics.

The synergy among these technologies is profound. ML provides the foundation for AI's decision-making capabilities, while DL enhances the depth and complexity of these decisions. AI, in turn, fuels the broader vision of creating intelligent systems capable of adaptive Learning, reasoning, and problem-solving.

2.4.3. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) represent a pivotal advancement in deep Learning, particularly in computer vision. These neural networks are tailored for image-related tasks and excel at capturing spatial hierarchies and patterns. CNNs are characterized by their unique architecture, incorporating convolutional layers that systematically analyze local regions of an input image.

The architecture of a typical CNN involves stacking multiple layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers utilize filters to perform convolution operations, effectively extracting features like edges, textures, and shapes from the input image. Pooling layers then downsample the spatial dimensions, reducing computational complexity while retaining essential information.

CNNs have proven highly effective in image classification tasks, demonstrating superior performance in competitions like ImageNet. Using pre-trained CNN models as feature extractors has become common, allowing for transfer learning across various applications.

One notable feature of CNNs is their ability to capture hierarchical representations. Lower layers focus on basic features, while deeper layers progressively learn more complex and abstract representations. This hierarchical feature learning is instrumental in the network's capacity to discern intricate patterns within images.

Beyond image classification, CNNs find applications in object detection, segmentation, and even in generating artistic content through style transfer. The success of CNNs has spurred their adoption in diverse domains, including healthcare for medical image analysis, autonomous vehicles for object recognition, and security systems for facial recognition.

In conclusion, Convolutional Neural Networks have revolutionized computer vision tasks by leveraging their specialized architecture for image-related applications. Their hierarchical feature learning, adaptability to diverse domains, and impressive performance make CNNs a cornerstone in advancing deep Learning and artificial intelligence.

2.4.4. Computer Vision

Computer vision is a dynamic interdisciplinary field that merges computer science and image processing to equip machines with the ability to interpret and comprehend visual information. At its core, computer vision involves the creation of algorithms and models that enable computers to analyze images and videos, akin to human visual perception. The primary goal is to facilitate machines in recognizing and understanding the content within visual data. Convolutional Neural Networks (CNNs) have been instrumental in advancing computer vision, particularly in tasks such as image classification, object detection, and facial recognition.

Beyond image recognition, computer vision spans various subfields. Object detection and tracking entail identifying and monitoring objects in video streams, which is essential for applications like surveillance and autonomous vehicles. Image segmentation involves dividing an image into segments to identify different objects or regions, which is applicable in medical imaging and satellite imagery analysis. Pose estimation focuses on determining objects or humans' spatial positioning and orientation within an image, which is crucial for gaming, sports analysis, and virtual reality.

Facial recognition is another significant application involving identifying and verifying individuals based on facial features, which is widely used in security systems and social media tagging. Scene understanding aims to interpret a scene's overall context and content, applicable in augmented reality and content moderation. Gesture recognition involves analyzing and interpreting gestures made by humans, employed in human-computer interaction and virtual reality. Medical image analysis utilizes computer vision to detect anomalies in medical images and aid in diagnoses.

The integration of deep Learning, coupled with access to large datasets and robust computational resources, has propelled the capabilities of computer vision. Ongoing research in the field focuses on enhancing robustness, interpretability, and ethical considerations in handling visual data. As computer vision technologies advance, their integration into various industries and applications reshapes how we interact with and extract insights from the visible world.

2.4.5. Tensorflow

TensorFlow is a versatile and open-source machine learning framework spearheaded by the Google Brain team. Renowned for its flexibility and scalability, TensorFlow is a go-to choice for developing and deploying machine learning models across diverse applications. Its robust support for neural network architectures, ranging from traditional to deep learning models such as CNNs and RNNs, positions it as a critical player in the field.

The framework is optimized for high-performance computing, ensuring efficient training and inference on CPUs and GPUs. TensorFlow's ecosystem includes TensorBoard for experiment visualization and TensorFlow Extended (TFX) for managing machine learning models in production. With an active and expansive community, TensorFlow continues to evolve, with contributions ranging from new features to resources like TensorFlow Hub for sharing pre-trained models.

Boasting high-level APIs like Keras, TensorFlow balances accessibility for beginners and flexibility for advanced users. Its incorporation of Keras as the official high-level API in TensorFlow two times further streamlines the model-building process. Offering support for multiple languages, including Python, C++, Java, and Go, TensorFlow caters to a diverse range of developer preferences.

The framework's continuous development, regular updates, and collaborative efforts with the open-source community keep it at the forefront of machine learning advancements. TensorFlow has played a pivotal role in shaping the landscape of machine learning and deep Learning, empowering developers and researchers to create sophisticated models across domains such as computer vision, natural language processing, and reinforcement learning.

2.4.6. Tensorflow Lite

TensorFlow Lite is a lightweight and mobile-friendly version of the TensorFlow machine learning framework. It is specifically designed to deploy machine learning models on mobile and embedded devices, making it well-suited for mobile apps, Internet of Things (IoT) devices, and edge computing scenarios.

One of the key features of TensorFlow Lite is its optimization for resource-constrained environments. It enables the execution of machine learning models on devices with limited computational power and memory, ensuring efficiency without compromising performance. This is particularly important for edge devices where real-time processing is often essential.

TensorFlow Lite supports a variety of model architectures, including custom models created using TensorFlow and pre-trained models available in the TensorFlow Hub. The framework allows developers to convert and optimize mobile device deployment models, balancing the trade-off between model size, speed, and accuracy.

The deployment of TensorFlow Lite models is facilitated through a runtime interpreter, providing a seamless interface between the machine learning model and the device's underlying hardware. This enables efficient execution and real-time inference, making it suitable for applications like image classification, object detection, and natural language processing on edge devices.

Furthermore, TensorFlow Lite integrates with other TensorFlow tools, allowing developers to train and fine-tune models using the full TensorFlow framework and then deploy the optimized models to mobile and embedded devices using TensorFlow Lite.

As the demand for on-device machine learning continues to grow, TensorFlow Lite plays a crucial role in enabling developers to bring the power of machine learning to a wide range of edge devices, contributing to the development of intelligent and responsive applications in the mobile and IoT space.

2.4.7. OpenCV

OpenCV, or Open Source Computer Vision Library, is a comprehensive and open-source computer vision and machine learning software library. It provides a rich set of tools and functions that facilitate the development of computer vision applications, image and video processing, and machine learning tasks. OpenCV is written in C++ and has bindings for various programming languages, including Python.

One of the primary strengths of OpenCV lies in its versatility. It offers many functionalities, ranging from basic image processing operations, such as filtering, transformations, and feature extraction, to more advanced tasks like object detection, facial recognition, and camera calibration. OpenCV is extensively used in both research and industry for applications spanning robotics, medical image analysis, augmented reality, and autonomous systems.

OpenCV's support for real-time computer vision is particularly noteworthy. It includes modules that enable video analysis, tracking, and the development of interactive applications. The library is optimized to run efficiently on diverse platforms, making it suitable for desktop and embedded systems applications.

Machine learning capabilities have been integrated into OpenCV, with modules that include support for training and using machine learning models. OpenCV's machine learning functionalities are often used in tasks like image classification, clustering, and object recognition.

Another notable feature of OpenCV is its support for various image and video file formats, camera interfaces, and graphical user interfaces (GUIs). This allows developers to integrate OpenCV seamlessly into applications with diverse requirements.

Overall, OpenCV serves as a powerful and accessible tool for developers and researchers working in computer vision and related fields. Its extensive range of functions, compatibility with multiple programming languages, and active community contribute to its widespread adoption in the computer vision community. OpenCV remains a fundamental resource for those working with computer vision, whether used for prototyping, academic research, or large-scale commercial applications.

2.4.8. Modern Processors Software and Computing Resources

Modern processors, also known as central processing units (CPUs), play a pivotal role in the functioning of computers and are a critical component of computing resources. These processors have evolved significantly over the years, offering increased performance, energy efficiency, and specialized features for various computing tasks.

These processors or chips are the brains of the operation. There are three major different kinds of processing units(PU), namely Central Processing Units (CPU), Graphic Processing Units (GPU), and Tensor Processing Units (TPU), with some other lesser-used processing units such as Data Processing Units (DPU) and Quantum Processing Units.

Computing advancements like parallelism of tasks, faster data processing, energy efficiency methods, and powerful professional tools like Google Collab will help us achieve our target.

2.4.9. Edge Devices Embedded Systems, The Internet Of Things (Iot) And Raspberry Pi

Edge devices, embedded systems, the Internet of Things (IoT), and Raspberry Pi are interconnected concepts that represent the forefront of modern computing, extending the capabilities of computing beyond traditional devices.

Edge Devices:

Edge devices operate at the "edge" or closer to the data source rather than relying on centralized cloud servers for processing. These devices have processing power and storage capabilities, enabling them to analyze and respond to data in near real-time. Edge computing is particularly valuable when low-latency and real-time processing are crucial, such as in industrial automation, smart cities, and autonomous vehicles.

Embedded Systems:

Embedded systems are specialized computing systems designed to perform specific functions within larger systems. They are dedicated to particular tasks and are often part of larger devices or products. Examples include embedded systems within appliances, automobiles, medical devices, and industrial machinery. Embedded systems are optimized for efficiency and reliability and often operate in real-time environments.

Internet of Things (IoT):

The Internet of Things (IoT) is a network of interconnected devices that communicate and share data with each other over the Internet. These devices, ranging from sensors and actuators to everyday objects, are embedded with computing capabilities. IoT enables data collection and exchange, facilitating smart applications in areas like home automation, healthcare, agriculture, and industrial monitoring.

Raspberry Pi:

Raspberry Pi is a series of small, affordable, single-board computers developed by the Raspberry Pi Foundation. These credit-card-sized computers are equipped with general-purpose input/output (GPIO) pins, HDMI ports, USB ports, and networking capabilities. Raspberry Pi is widely used for educational purposes, DIY projects, and as a platform for prototyping IoT solutions. Its versatility and low cost make it popular for enthusiasts, hobbyists, and developers.

Integration of Edge Computing, IoT, and Raspberry Pi:

The integration of edge computing, IoT, and Raspberry Pi exemplifies the distributed and decentralized nature of modern computing. Raspberry Pi devices can serve as edge nodes, processing data locally before sending relevant information to the cloud. This is especially advantageous in IoT applications where real-time processing is essential, and bandwidth may be limited.

For instance, in a smart home scenario, Raspberry Pi devices can act as local hubs that process sensor data from various IoT devices within the home. This local processing reduces latency and enhances responsiveness. The aggregated, relevant data can then be sent to the cloud for further analysis or storage.

2.5. METHODOLOGIES AND APPROACHES

The literature survey delves into the diverse methodologies and approaches adopted by researchers in addressing the challenges inherent in real-time image classification and object detection. A comparative analysis of various techniques sheds light on the strengths and limitations of different models, providing valuable insights for the proposed work in this thesis.

We will see more about that in the next chapter.

2.6. APPLICATIONS AND USE CASES

Understanding the practical applications and use cases of real-time image classification and object detection in the context of surveillance technology is essential. This section explores how these technologies have been implemented in diverse scenarios, such as smart cities, transportation systems, and security surveillance, to enhance situational awareness and decision-making processes.

Currently, on an extensive scale, image, object, and face detection nationally are done in the People's Republic of China. Their systems and all other technologies are proprietary and not publicly accessible. Their government aims to keep track of people and build a social credit system, Social credibility, etc. On any grounds, this seems to be an invasion of privacy and not ethical or morally correct. But we won't explore any such political aspect as such.

It could be used in many places, not just in surveillance but in self-driving cars, keeping track of inventory, alarm detection systems, monitoring and tracking purposes, attendance systems, etc.

2.7. CURRENT CHALLENGES AND FUTURE DIRECTIONS

Identifying the current challenges faced by researchers in the field and envisioning future directions are crucial components of the literature survey. By doing so, this survey aims to contribute to the ongoing discourse, providing a roadmap for addressing gaps in knowledge and charting a course for future research endeavors.

In summary, this literature survey embarks on a comprehensive exploration of the historical evolution, state-of-the-art technologies, methodologies, applications, challenges, and future directions in the domain of real-time image classification and object detection for surveillance technology. Through this exploration, the survey sets the stage for the proposed work in this thesis, ensuring that a nuanced understanding of the existing body of knowledge in the field informs the subsequent chapters.

Chapter 3
PROPOSED WORK and METHODOLOGY

3.1. OVERVIEW

The Proposed Work and Methodology section lays the foundation for the technical aspects of this thesis. It articulates the novel contributions of the project, outlining the architectural design, key components, and the systematic approach employed for real-time image classification and object detection using TensorFlow Lite on the Raspberry Pi 4b.

3.2. SYSTEM ARCHITECTURE

Here are some key elements from our architecture/structure of the project. These all, when integrated together, completes our system architecture.

3.2.1. Processing Units

To better understand the methodology and implement it in the projet, we should know the difference between CPUs, GPUs, TPUs, DPUs, etc.

CPUs (Central Processing Units), GPUs (Graphics Processing Units), TPUs (Tensor Processing Units), and DPUs (Data Processing Units) are different types of processing units designed for specific computing tasks. Each type has unique characteristics and is optimized for particular workloads.

CPUs (Central Processing Units):
CPUs are the primary processing units in a computer and are designed for general-purpose computing tasks. They handle tasks such as running the operating system, executing software applications, and managing overall system operations. CPUs are characterized by their versatility, capable of handling a wide range of tasks, but they may not be optimized for highly parallel workloads.

GPUs (Graphics Processing Units):
GPUs are specialized processors initially designed for rendering graphics in computer games. However, their parallel architecture makes them well-suited for parallelizable tasks beyond graphics. Modern GPUs are extensively used for general-purpose computing tasks, such as scientific simulations, machine learning, and data-parallel computations. They excel in scenarios where many calculations can be performed simultaneously.

TPUs (Tensor Processing Units):
TPUs are custom accelerators explicitly designed for machine learning workloads, particularly those involving neural networks. Developed by Google, TPUs are optimized for the efficient execution of tensor-based computations, which are common in deep learning models. They offer high throughput and energy efficiency for tasks like training and inference in machine learning applications.

DPUs (Data Processing Units):
DPUs refer to Data Processing Units that offload and accelerate networking-related tasks, improving data transfer efficiency. In the context of SmartNICs (Smart Network Interface Cards), DPUs handle functions such as packet processing, security, and data compression, reducing the load on the CPU.

In summary; CPUs are general-purpose processors suitable for a wide range of tasks.
GPUs are specialized processors initially designed for graphics but widely used for parallelizable tasks, including scientific simulations and machine learning.
TPUs are custom processors Google developed and optimized for tensor-based machine learning computations.
DPUs in networking contexts refer to Data Processing Units that accelerate and offload networking-related tasks, enhancing data transfer efficiency.

The diversity of these processing units reflects the need for specialized hardware to handle the diverse and evolving requirements of modern computing, ranging from general-purpose tasks to specialized workloads like graphics rendering and machine learning.

The proposed system architecture is designed to seamlessly integrate TensorFlow Lite with the Raspberry Pi 4b, creating a powerful yet energy-efficient platform for on-device image analysis. The architecture encompasses two main modules: the real-time image classification module and the object detection module. These modules work in tandem to deliver a comprehensive surveillance solution.

So, from the above, it is now up to us to decide what processing unit to use for computation. The CPU is suitable for sequential calculations and some other pretty exciting things. When it comes to GPU, it does simpler tasks like matrix multiplication on a large scale and other calculations to provide better performance/graphics compared to the CPU. TPU is used extensively to run for TensorFlow and its libraries. We will be taking all three of the above to achieve our aim. The CPU will do computations and run the simulation, and graphics will be computed with GPU, like processing the camera data. The last part is TPU; due to the unavailability of TPU, we only have the theory part of what differences it could make to our work.

3.2.2. Raspberry Pi 4b

The Raspberry Pi is a popular choice for various projects due to its versatility, affordability, and compact form factor. In the context of our work over here, the use of Raspberry Pi for real-time image classification and object detection with TensorFlow Lite on a Raspberry Pi 4B for surveillance technology can be attributed to several key factors.

Firstly, the Raspberry Pi's compact size makes it suitable for embedded applications, allowing for easy integration into surveillance systems. Its GPIO pins and peripheral interfaces offer flexibility for connecting cameras, sensors, and other components essential for surveillance technology.

Secondly, the affordability of Raspberry Pi makes it an accessible platform for a wide range of projects, particularly those developed by hobbyists, students, or small-scale implementations. This cost-effectiveness is crucial, especially when deploying multiple units in a surveillance network.

Additionally, the Raspberry Pi 4B model has improved processing power, featuring a quad-core ARM Cortex-A72 processor, making it capable of handling real-time image processing tasks. The onboard GPU also contributes to tasks like video decoding and graphical processing.

The Raspberry Pi community and ecosystem are substantial advantages. Resources, tutorials, and community support are available, facilitating development and troubleshooting. The Raspberry Pi Foundation actively encourages educational and DIY projects, contributing to a vibrant community of developers and enthusiasts.

Moreover, the Raspberry Pi supports TensorFlow Lite, a version of the popular machine learning framework optimized for edge devices. This enables the deployment of machine learning models directly on the Raspberry Pi, making it suitable for real-time image classification and object detection applications in surveillance.

The **Raspberry Pi 4b 8gb ram variant** was chosen for this project due to its compact form factor, affordability, processing capabilities, community support, and compatibility with TensorFlow Lite, making it an excellent fit for implementing real-time image classification and object detection in a surveillance technology context.

When the project started, the latest available Raspberry Pi device at that time was Raspberry Pi model 4b, commonly known as Raspberry Pi 4. In this paper, whenever and wherever we mention Raspberry Pi, it should be understood that the device we are discussing is Raspberry Pi 4b. And at the present there has been released a newer version of Raspberry Pi device named Raspberry Pi 5 which is newer version/model of the current Raspberry Pi 4b, making it the default choice for everyone.

Here some interesting features of Raspberry pi 4b and 5 is mentioned

Raspberry Pi 4b

- Processor: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
- RAM: 1GB, 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM (depending on model)
- GPU: VideoCore VI with support for OpenGL ES 3.1 and Vulkan 1.1
- Storage: microSD card slot for up to 2TB
- Connectivity: Gigabit Ethernet, dual-band Wi-Fi 5 (802.11ac), Bluetooth 5.0
- Ports: 2x micro HDMI (up to 4K resolution at 30fps), 2x USB 3.0, 2x USB 2.0, 40-pin GPIO header, 3.5mm audio jack

Raspberry Pi 5

- Processor: Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with cryptography extensions, 512KB per-core L2 caches and a 2MB shared L3 cacheRAM: 4GB or 8GB LPDDR4X-4267 SDRAM

- GPU: VideoCore VII with support for OpenGL ES 3.1 and Vulkan 1.2
- RAM: LPDDR4X-4267 SDRAM (4GB and 8GB SKUs available at launch)
- Storage: microSD card slot for up to 2TB
- Connectivity: Gigabit Ethernet, dual-band Wi-Fi 5 (802.11ac), Bluetooth 5.0 / Bluetooth Low Energy (BLE)
- Ports: 2x micro HDMI (up to 4Kp60 resolution with HDR), 2x USB 3.0, 2x USB 2.0, 40-pin GPIO header, 3.5mm audio jack, PCIe Gen 2.0 slot for M.2 SSD

Additional features of the Raspberry Pi 5:

- PCIe Gen 2.0 slot for any compatible device like M.2 SSD allows faster storage and boot times. It could significantly increase the recording/storing speeds of the video processing for future use.
- Dual 4Kp60 HDMI display output with HDR support: This allows for connecting two monitors or TVs.
- Power button.
- Real-time clock (RTC): The Raspberry Pi can keep time even when turned off. This could help with maintaining time even if the power is disrupted when directly compared to Raspberry Pi 4.

Here is a table summarizing the key differences between the Raspberry Pi 4b and Raspberry Pi 5:

Feature	Raspberry Pi 4b	Raspberry Pi 5
CPU	Quad-core Cortex-A72 at 1.8 GHz	Quad-core Cortex-A76 at 2.4 GHz
RAM	1GB, 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM	4GB or 8GB LPDDR4X-4267 SDRAM
GPU	VideoCore VI	VideoCore VII
Connectivity	Gigabit Ethernet (POE), dual-band Wi-Fi 5 (802.11ac), Bluetooth 5.0	Gigabit Ethernet (POE), dual-band Wi-Fi 5 (802.11ac), Bluetooth 5.0
Ports	Lesser advanced ports	More advanced ports
Additional features	Separate Dedicated CSI, DSI lanes	Mentioned in the above details

Table 3.2.2.1: Differences between the Raspberry Pi 4b and Raspberry Pi 5

3.2.3. Camera on Raspberry Pi 4b

Why the camera features are restricted in Raspberry pi?

Legacy Hardware:
Raspberry Pi devices, particularly the earlier models, utilize older camera hardware components that may not be able to support advanced features or handle the processing demands of modern image processing tasks. This outdated hardware limitations hinder the development and implementation of cutting-edge camera features.

Limited Processing Power:
Raspberry Pi devices, especially the entry-level models, are known for their compact size and affordability, often coming with constrained processing power. While this makes them suitable

for basic computing tasks, it can constrain their ability to handle the computational demands of real-time image processing, which is essential for advanced camera features.

Prioritization of Other Features:

The Raspberry Pi Foundation, the organization behind the Raspberry Pi project, prioritizes other features, such as general computing performance, connectivity, and affordability, over advanced camera capabilities. This aligns with their focus on providing a versatile and accessible computing platform for a wide range of users, rather than specifically targeting high-end camera applications.

Restricted Software Support:

The software ecosystem for Raspberry Pi devices, while extensive, may not provide comprehensive support for all advanced camera features. This could be due to the limitations of the underlying hardware or a lack of focus on developing software libraries and drivers for specific camera functionalities.

Security Considerations:

Given the widespread use of Raspberry Pi devices in various settings, including education, hobbyist projects, and even home automation, security concerns arise when dealing with camera access and data handling. Implementing robust security measures to protect user privacy and prevent unauthorized access to camera data is crucial, but it adds complexity to the development and implementation of camera features.

In summary, the legacy camera features on Raspberry Pi devices stem from a combination of hardware limitations, processing power constraints, prioritization of other features, restricted software support, and security considerations. While this may limit the availability of advanced camera functionalities, Raspberry Pi devices remain valuable platforms for a wide range of computing applications.

So what we have done is here to access the camera is; we used a previous version but not so old raspian os on the board the following. Further details about those in the below section.

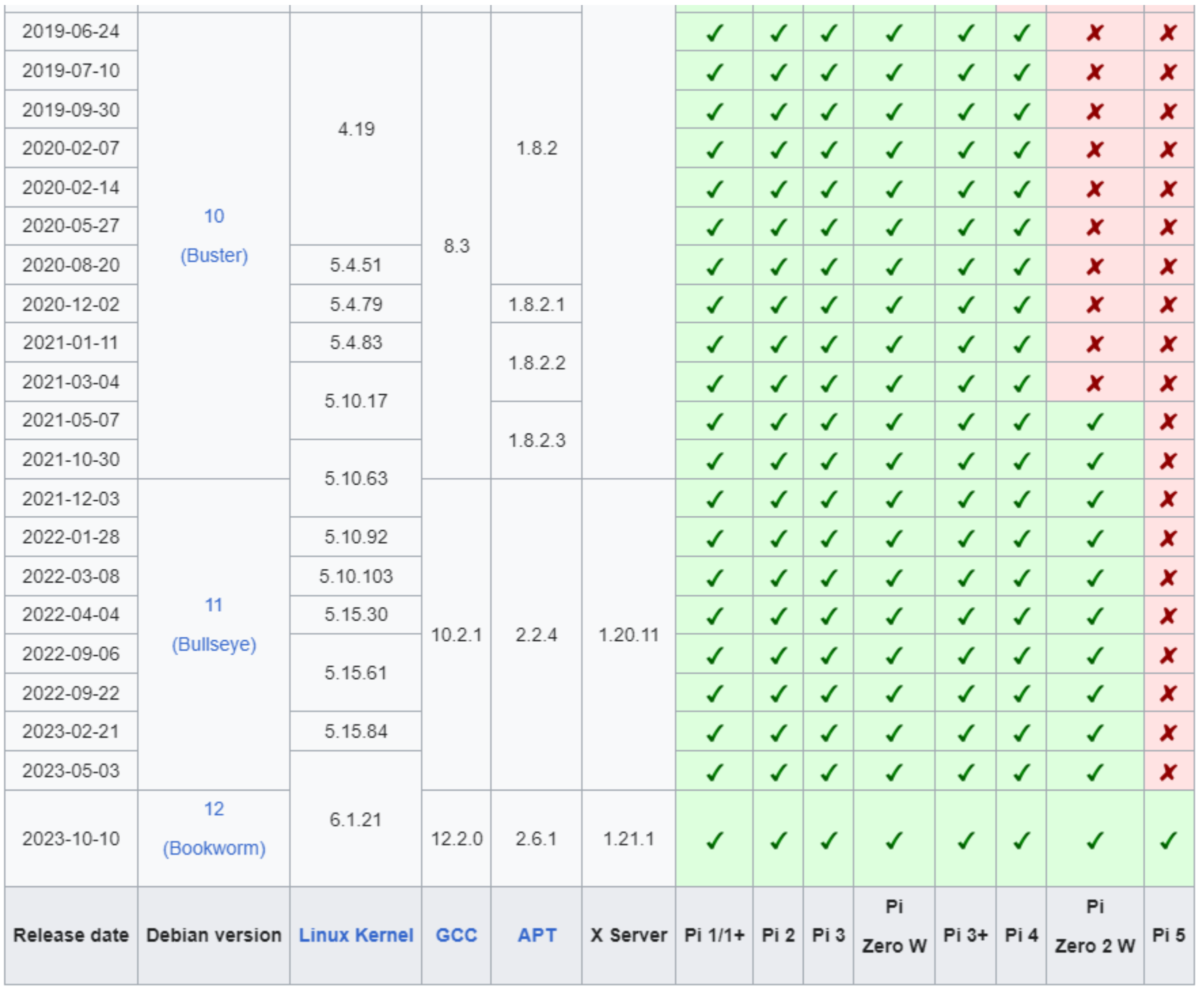
3.2.4. Operating System

Raspian Operating System(Raspbbery Pi OS)

Raspberry Pi OS is a free operating system based on Debian, optimised for the Raspberry Pi hardware, and is the recommended operating system for normal use on a Raspberry Pi. The OS comes with over 35,000 packages: pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.

Raspberry Pi OS is under active development, with an emphasis on improving the stability and performance of as many Debian packages as possible on Raspberry Pi.

In summary, the Raspberry Pi OS serves as the operating system foundation for your project, providing the necessary environment for developing and deploying machine learning models. The partial compatibility of the Raspberry Pi Camera Module enhances the project's capabilities for real-time image processing and surveillance applications. Please check the official Raspberry Pi website for the latest information on the operating system and camera module



2019-06-24	10 (Buster)	4.19	8.3	1.8.2		✓	✓	✓	✓	✓	✓	✗	✗
2019-07-10						✓	✓	✓	✓	✓	✓	✗	✗
2019-09-30						✓	✓	✓	✓	✓	✓	✗	✗
2020-02-07						✓	✓	✓	✓	✓	✓	✗	✗
2020-02-14						✓	✓	✓	✓	✓	✓	✗	✗
2020-05-27						✓	✓	✓	✓	✓	✓	✗	✗
2020-08-20		5.4.51				✓	✓	✓	✓	✓	✓	✗	✗
2020-12-02		5.4.79		1.8.2.1		✓	✓	✓	✓	✓	✓	✗	✗
2021-01-11		5.4.83		1.8.2.2		✓	✓	✓	✓	✓	✓	✗	✗
2021-03-04		5.10.17				✓	✓	✓	✓	✓	✓	✗	✗
2021-05-07				1.8.2.3		✓	✓	✓	✓	✓	✓	✓	✗
2021-10-30		5.10.63				✓	✓	✓	✓	✓	✓	✓	✗
2021-12-03	11 (Bullseye)		10.2.1	2.2.4	1.20.11	✓	✓	✓	✓	✓	✓	✓	✗
2022-01-28		5.10.92				✓	✓	✓	✓	✓	✓	✓	✗
2022-03-08		5.10.103				✓	✓	✓	✓	✓	✓	✓	✗
2022-04-04		5.15.30				✓	✓	✓	✓	✓	✓	✓	✗
2022-09-06		5.15.61				✓	✓	✓	✓	✓	✓	✓	✗
2022-09-22						✓	✓	✓	✓	✓	✓	✓	✗
2023-02-21		5.15.84				✓	✓	✓	✓	✓	✓	✓	✗
2023-05-03		6.1.21				✓	✓	✓	✓	✓	✓	✓	✗
2023-10-10	12 (Bookworm)		12.2.0	2.6.1	1.21.1	✓	✓	✓	✓	✓	✓	✓	✓
Release date	Debian version	Linux Kernel	GCC	APT	X Server	Pi 1/1+	Pi 2	Pi 3	Pi Zero W	Pi 3+	Pi 4	Pi Zero 2 W	Pi 5

Table 3.2.4.1: Differences between the Raspberry Pi OS and compatibility

The above table is an extract from the Wikipedia page of Raspberry Pi OS. The camera features were present in previous generation Raspberry Pi models and OS. On Raspberry Pi 4b, the first OS by the organization supported by Raspberry Pi is Buster. Initially, it had camera features as a legacy, but in later versions, it was a legacy and optional feature. With the newer release of the OS Bullseye, the camera features were legacy decrypted, not removed features. It should be noted that due to device constraints and OS/kernel locking the elements, the camera isn't natively supported or runs on the machine. Still, it is accessed using specific functions, methods, and modifications.

3.3. REAL-TIME IMAGE CLASSIFICATION AND OBJECT DETECTION MODULE

The real-time image classification module leverages the capabilities of TensorFlow Lite to process images in real-time. A pre-trained machine learning model, fine-tuned for specific surveillance scenarios, forms the core of this module. The TensorFlow Lite interpreter enables efficient execution on the resource-constrained Raspberry Pi 4b, ensuring minimal latency in image classification.

The object detection module is designed to identify and track multiple objects concurrently within the surveillance environment. Building upon state-of-the-art object detection techniques, this module integrates a robust algorithm to analyze video streams and provide real-time updates on detected objects. TensorFlow Lite optimizations enable this module to operate seamlessly on the edge device.

3.3.1. Finding a model for our application

Finding an existing TensorFlow Lite model for our work use case can be tricky, depending on what one tries to accomplish. Here are a few recommended ways to discover models for use with TensorFlow Lite:

- **By example:** The fastest way to find and start using models with TensorFlow Lite is to browse the TensorFlow Lite Examples section to find models that perform a task which is similar to your use case. This short catalog of examples provides models for common use cases with explanations of the models and sample code to get you started running and using them.
- **By data input type:** Aside from looking at examples similar to your use case, another way to discover models for your own use is to consider the type of data you want to process, such as audio, text, images, or video data. Machine learning models are frequently designed for use with one of these types of data, so looking for models that handle the data type you want to use can help you narrow down what models to consider. On TensorFlow Hub, you can use the Problem domain filter to view model data types and narrow your list.

Sources of this model: One could use TensorFlow lite examples and TensorFlow hub to get a pre-trained model. Tensorflow Models could be converted to TensorFlow lite models.

The real-time image classification module is a critical component of the project, harnessing the power of TensorFlow Lite to deliver swift and efficient processing of images in real-time. At the heart of this module is the utilization of a pre-trained machine learning model, specifically fine-tuned and optiization for the nuances of surveillance scenarios. In this context, the chosen model is Mobilenet-v1-1-224, renowned for its balance between accuracy and computational efficiency.

Mobilenet-v1-1-224, a variant of MobileNet, is particularly well-suited for edge devices like the Raspberry Pi 4B. Its architecture is designed to be lightweight, making it adept at real-time applications with constrained resources. This aligns seamlessly with the Raspberry Pi's capabilities, ensuring that the image classification module functions optimally without compromising on performance.

The TensorFlow Lite interpreter is instrumental in the seamless integration of the chosen machine learning model with the Raspberry Pi. This interpreter, optimized for edge computing, facilitates the execution of the Mobilenet-v1-1-224 model on the resource-constrained hardware.

Its efficiency is paramount in achieving minimal latency in image classification, a crucial requirement for surveillance applications where rapid and accurate identification is paramount.

The choice of Mobilenet-v1-1-224 is strategic, considering the balance it strikes between model accuracy and the computational demands imposed on the Raspberry Pi. This model excels in capturing meaningful features from images, making it well-suited for discerning relevant details in surveillance scenarios.

3.3.2. Mobilenet-v1-1-224 model

mobilenet-v1-1.0-224 is one of MobileNets - small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases. They can be built upon for classification, detection, embeddings and segmentation similar to how other popular large scale models are used. For details, see the paper.

Specifications

Metric	Value
Type	Classification
GFlops	1.148
MParams	4.222
Source framework	TensorFlow*

Accuracy

Metric	Value
Top 1	71.03%
Top 2	89.94%

3.3.3. Model Optimization

Edge devices often have limited memory or computational power. Various optimizations can be applied to models so that they can be run within these constraints. In addition, some optimizations allow the use of specialized hardware for accelerated inference.
TensorFlow Lite and the TensorFlow Model Optimization Toolkit provide tools to minimize the complexity of optimizing inference.
It's recommended that you consider model optimization during your application development process. This document outlines some best practices for optimizing TensorFlow models for deployment to edge hardware.

Why models should be optimized
There are several main ways model optimization can help with application development.

Size reduction
Some forms of optimization can be used to reduce the size of a model. Smaller models have the following benefits:

- **Smaller storage size**: Smaller models occupy less storage space on your users' devices. For example, an Android app using a smaller model will take up less storage space on a user's mobile device.
- **Smaller download size:** Smaller models require less time and bandwidth to download to users' devices.
- **Less memory usage:** Smaller models use less RAM when they are run, which frees up memory for other parts of your application to use, and can translate to better performance and stability.

Quantization can reduce the size of a model in all of these cases, potentially at the expense of some accuracy. Pruning and clustering can reduce the size of a model for download by making it more easily compressible.

Latency reduction
Latency is the time it takes to run a single inference with a given model. Some forms of optimization can reduce the computation required to run inference using a model, resulting in lower latency. Latency can also have an impact on power consumption.
Quantization can reduce latency by simplifying the calculations that occur during inference, potentially at the expense of some accuracy.

Accelerator compatibility
Some hardware accelerators, such as the Edge TPU, can run inference extremely fast with models that have been correctly optimized.
Generally, these types of devices require models to be quantized in a specific way. See each hardware accelerator's documentation to learn more about their requirements.

Types of optimization

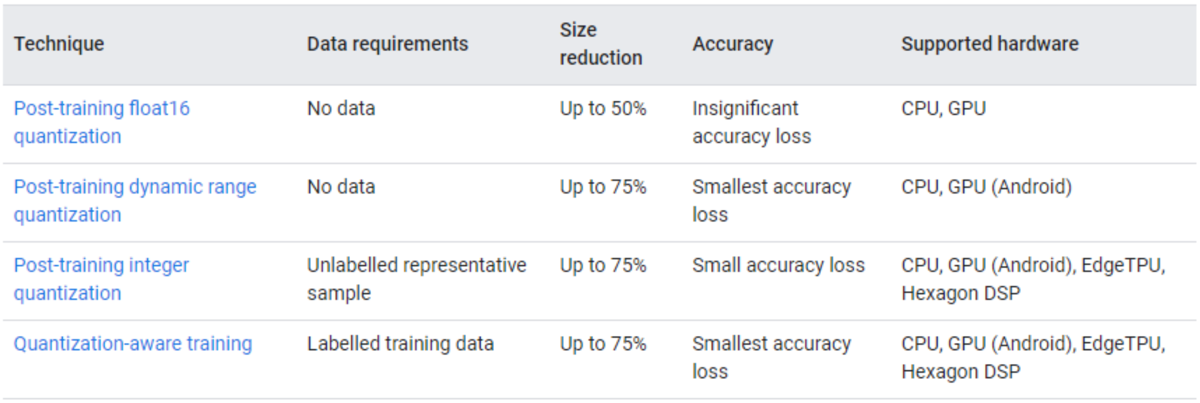
TensorFlow Lite currently supports optimization via quantization, pruning and clustering.

These are part of the TensorFlow Model Optimization Toolkit, which provides resources for model optimization techniques that are compatible with TensorFlow Lite.

Quantization

Quantization works by reducing the precision of the numbers used to represent a model's parameters, which by default are 32-bit floating point numbers. This results in a smaller model size and faster computation.

The following types of quantization are available in TensorFlow Lite:



Technique	Data requirements	Size reduction	Accuracy	Supported hardware
Post-training float16 quantization	No data	Up to 50%	Insignificant accuracy loss	CPU, GPU
Post-training dynamic range quantization	No data	Up to 75%	Smallest accuracy loss	CPU, GPU (Android)
Post-training integer quantization	Unlabelled representative sample	Up to 75%	Small accuracy loss	CPU, GPU (Android), EdgeTPU, Hexagon DSP
Quantization-aware training	Labelled training data	Up to 75%	Smallest accuracy loss	CPU, GPU (Android), EdgeTPU, Hexagon DSP

Table 3.3.3.1: Types of quantization are available in TensorFlow Lite

Below are the latency and accuracy results for post-training quantization and quantization-aware training on a few models. All latency numbers are measured on Pixel 2 devices using a single big core CPU. As the toolkit improves, so will the numbers here:

Benefits of model quantization for select CNN models

Model	Top-1 Accuracy (Original)	Top-1 Accuracy (Post Training Quantized)	Top-1 Accuracy (Quantization Aware Training)	Latency (Original) (ms)	Latency (Post Training Quantized) (ms)	Latency (Quantization Aware Training) (ms)	Size (Original) (MB)	Size (Optimized) (MB)
Mobilenet-v1-1-224	0.709	0.657	0.700	0124	0112	0064	16.90	04.3
Mobilenet-v2-1-224	0.719	0.637	0.709	0089	98	54	014.0	03.6
Inception_v3	0.780	0.772	0.775	1130	845	543	95.70	23.9
Resnet_v2_101	0.770	0.768	N/A	3973	2868	N/A	178.3	44.9

Source: Tensorflow lite docunetation [*https://www.tensorflow.org/lite/performance/model_optimization#trade-offs*](https://www.tensorflow.org/lite/performance/model_optimization#trade-offs)
TensorFlow Lite conversion workflow.

Table 3.3.3.2: Benefits of model quantization for select CNN models

Model conversion overview

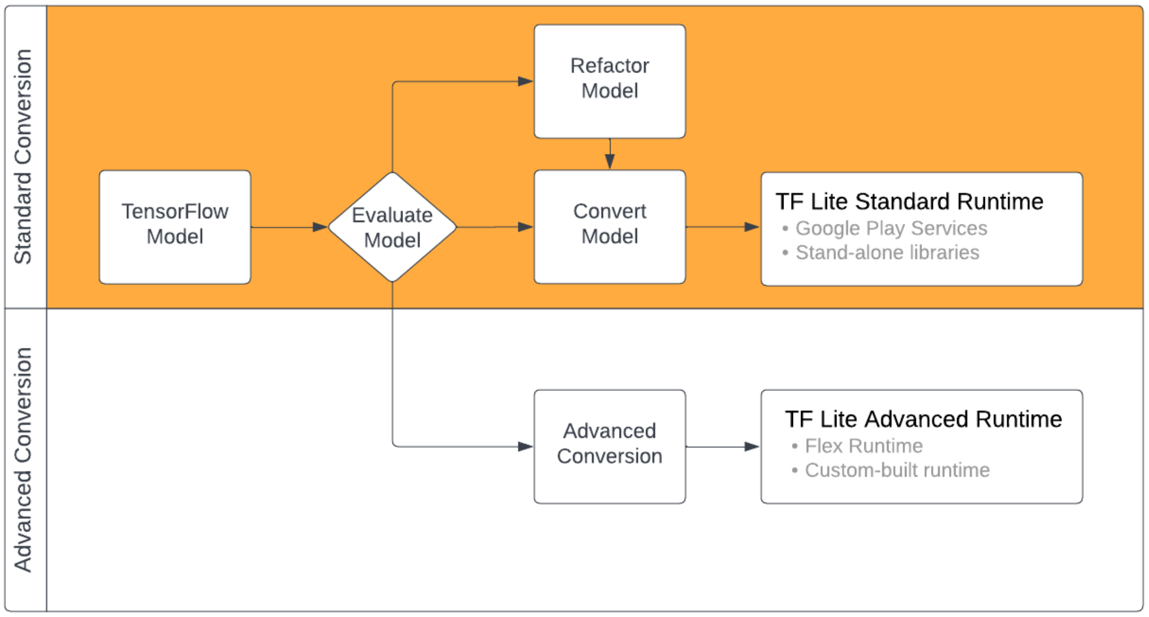
The machine learning (ML) models you use with TensorFlow Lite are originally built and trained using TensorFlow core libraries and tools. Once you've built a model with TensorFlow core, you can convert it to a smaller, more efficient ML model format called a TensorFlow Lite model.

This section provides guidance for converting your TensorFlow models to the TensorFlow Lite model format.

Conversion workflow

Converting TensorFlow models to TensorFlow Lite format can take a few paths depending on the content of your ML model. As the first step of that process, you should evaluate your model to determine if it can be directly converted. This evaluation determines if the content of the model is supported by the standard TensorFlow Lite runtime environments based on the TensorFlow operations it uses. If your model uses operations outside of the supported set, you have the option to refactor your model or use advanced conversion techniques.

The diagram below shows the high level steps in converting a model.



TensorFlow Lite conversion workflow.

Table 3.3.3.3: TensorFlow Lite conversion workflow.

Evaluating your model is an important step before attempting to convert it. When evaluating, you want to determine if the contents of your model is compatible with the TensorFlow Lite format. You should also determine if your model is a good fit for use on mobile and edge devices in terms of the data the model uses, its hardware processing requirements, and its overall size and complexity.

3.4. DATASET

The object detection module is designed to identify and track multiple objects concurrently within the surveillance environment. Building upon state-of-the-art object detection techniques, this module integrates a robust algorithm to analyze video streams and provide real-time updates on detected objects. TensorFlow Lite optimizations enable this module to operate seamlessly on the edge device.

These MobileNet models have been trained on the ILSVRC-2012-CLS image classification dataset. Accuracies were computed by evaluating using a single image crop. In the initial database, there were ore than 1000 of object classes. More details about it can be found on this website https://www.image-net.org/challenges/LSVRC/2012/ .

The model's dataset is reduced to 100 classes of objects considering the system.

The custom dataset we used is edited and labeled with xml editor and the data was extracted using a Python script that uses OpenCV functions. These scripts could be foind in the repositories.

3.5. INTEGRATION WITH RASPBERRY PI 4B

Deploying the proposed system on the Raspberry Pi 4b involves careful resource utilization and performance optimization considerations. The lightweight nature of TensorFlow Lite allows for efficient execution on the Raspberry Pi, making it an ideal choice for edge computing applications. The integration process is detailed, including dependencies, libraries, and configurations necessary for the system to operate effectively.

3.6. METHODOLOGY

The methodology section articulates the step-by-step process undertaken to realize the proposed work. This includes:

- **Data Collection:** Gathering a diverse dataset representative of the surveillance scenarios the system is intended for.
  We have used the sample dataset mentioned above as it is highly optimized.
  We also made a custom dataset with smaller classes for our work.
- **Model Selection and Training:** Choosing an appropriate pre-trained model for image classification and fine-tuning it on the collected dataset to enhance its suitability for the specific surveillance environment. We have selected the model said above.
- **System Integration:** Detailing the steps involved in integrating the developed modules into a cohesive system on the Raspberry Pi 4b.

3.7. PERFORMANCE METRICS

To evaluate the effectiveness of the proposed system, a set of performance metrics is defined. These metrics include accuracy in image classification, real-time processing speed, and the system's ability to detect and track objects accurately. The choice of metrics aligns with the objectives outlined in the Introduction, providing a quantitative assessment of the system's performance.

The details about this is also mentioned in the above sections.

3.8. ETHICAL CONSIDERATIONS

In the development and deployment of any surveillance technology, ethical considerations are paramount. This section addresses the ethical aspects associated with the proposed system, including privacy concerns, data security, and the responsible use of surveillance technology.

In essence, the Proposed Work and Methodology section provides a comprehensive understanding of the technical underpinnings of the project. From system architecture to the step-by-step methodology, this section serves as a guide for the subsequent implementation and evaluation stages, ensuring transparency and reproducibility in the research process.

Chapter 4
RESULT DISCUSSION

Images and video results have been included in the repository. Some images have been included in the later sections. It will show the details of the model.

4.1. EXPERIMENTAL SETUP

Before delving into the results, it's crucial to outline the experimental setup to provide context for the findings. The hardware specifications of the Raspberry Pi 4b used, details of the dataset, and any specific configurations employed during the experiments are detailed in this section. This transparency ensures reproducibility and allows readers to contextualize the results within the given parameters.

This setup was done in such a way that the throughput could be maximized even when the system was loaded. We set the pi over both SSH and also connected a monitor as we wanted it to control both locally and over the network.
The OS previously mentioned Raspbian os Bullseye is used (32-bit). We have installed the latest version of other packages such as GCC, g++, python, opencv, TensorFlow-lite, and other required libraries which are further mentioned in the get requirements txt in the repository.

4.2. REAL-TIME IMAGE CLASSIFICATION RESULTS

The section begins with the presentation of the results of real-time image classification. Metrics such as accuracy, precision, recall, and F1 score are reported, providing a comprehensive evaluation of the model's performance. Any challenges encountered during the classification process and potential mitigations are discussed.

From the Images attached in the below sections we can visually interpret or see the results of our work; It shows multiple objects that are tracked in a frame.

4.3. OBJECT DETECTION RESULTS

Following image classification, the focus shifts to the object detection module. Detection accuracy, tracking capabilities, and the system's ability to handle multiple objects in real-time are thoroughly examined. Comparative analyses with existing object detection frameworks may be presented to highlight the efficiency and uniqueness of the proposed system.

We could see the results/output where multiple objects are detected and labeled according to their class of object as per the dataset accordingly and accurately.

4.4. SYSTEM PERFORMANCE METRICS

Quantitative metrics related to system performance are presented in this subsection. These metrics include the processing speed of the system, memory utilization, and energy efficiency. Benchmarks are established, comparing the performance of the proposed system against baseline models or industry standards, where applicable.

We have used frame rates per second as a metric where we could evaluate how well our model processes and outputs the data. Along with frame rates per second (FPS) we have used other metrics to accurately identify the object. Here we use a simple ratio/percentage which represents how sure our model is about the object identified.

4.5. QUALITATIVE ANALYSIS

Beyond quantitative metrics, a qualitative analysis is imperative to assess the real-world applicability of the proposed system. Visual examples of successful classifications and object detections are showcased. Additionally, instances where the system faced challenges or limitations are presented, accompanied by discussions on potential improvements.

Our model struggles due to the optimization of the model to such an extent that it sometimes produces bogus results. One of the major problems that we are facing right now is that the model detects an object successfully but it mistakes for some other class of object which leads to misleading results where the object detected is marked/labeled incorrectly.

Along with it detects one object multiple times due to the inaccuracy. As here with our model, the tradeoffs are whether performance or accuracy on the other hand. So we had to make a decision here. We went with the performance metric as it was more important at the moment compared to accuracy as accuracy is something that could be improved by the efficiency of the model that we have used. It could also lead to better performance at later stages. But at the time of the work, some highly accurate models do detect objects better and have better results and accuracy and performance metrics for that matter. However, these models are limited to TensorFlow rather than TensorFlow-lite APIs that have been used in our case due to restrictions of the hardware platform.

4.6. COMPARATIVE ANALYSIS

To benchmark the proposed system against existing solutions, a comparative analysis is conducted. This may involve comparing the system's performance with similar implementations, highlighting areas where the proposed work excels, and identifying potential areas for further refinement.

When compared with other such models and other similar experimental setups on other such devices such as Raspberry Pi Pico, Raspberry Pi 3, Orange Pi, and other such edge devices, etc. Our model and optimizations perform at least 10 times better results than other such models.

4.7. DISCUSSION OF FINDINGS

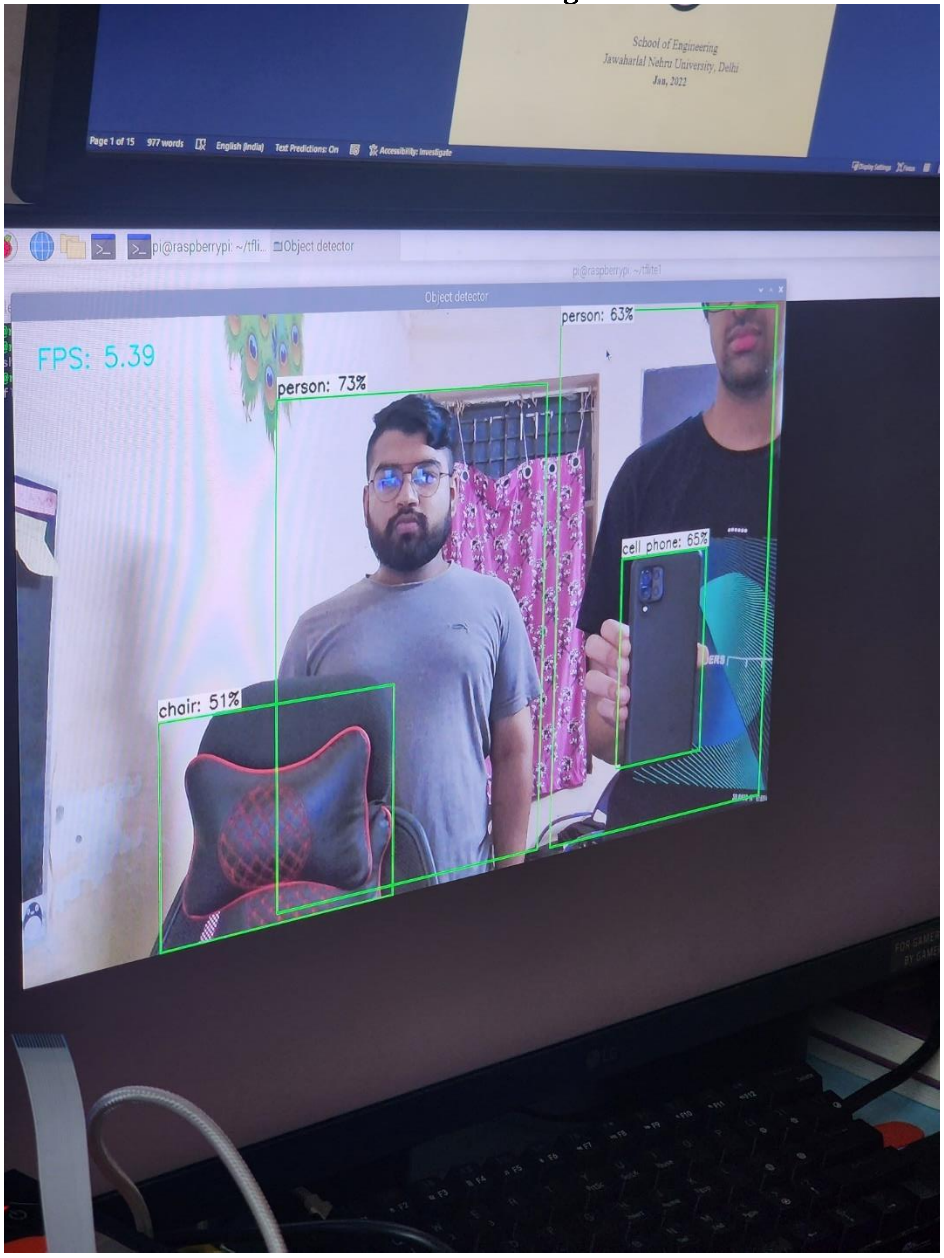
The discussion component interprets the results in the context of the project's objectives. The strengths and weaknesses of the proposed system are explored, and insights gained from the experimental outcomes are discussed. Any unexpected findings or challenges encountered during the experiments are addressed, providing a nuanced understanding of the system's capabilities.

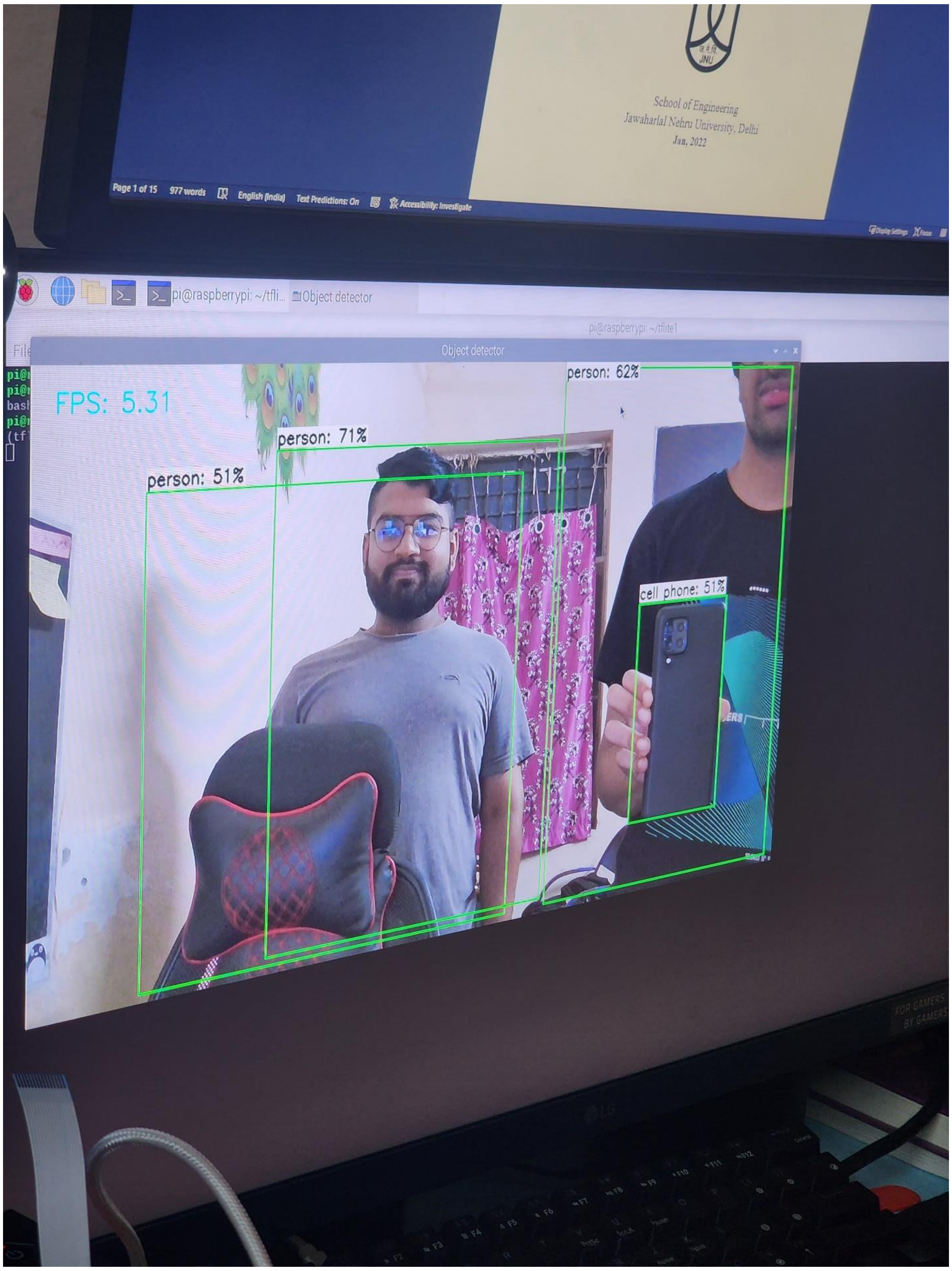
Here we have seen that the project easily identifies a person and other such objects in the domain of the classes of the object. This is not just it but it also sometimes mistakes some other objects for some other objects. Overall the videos outstream are pretty responsive.

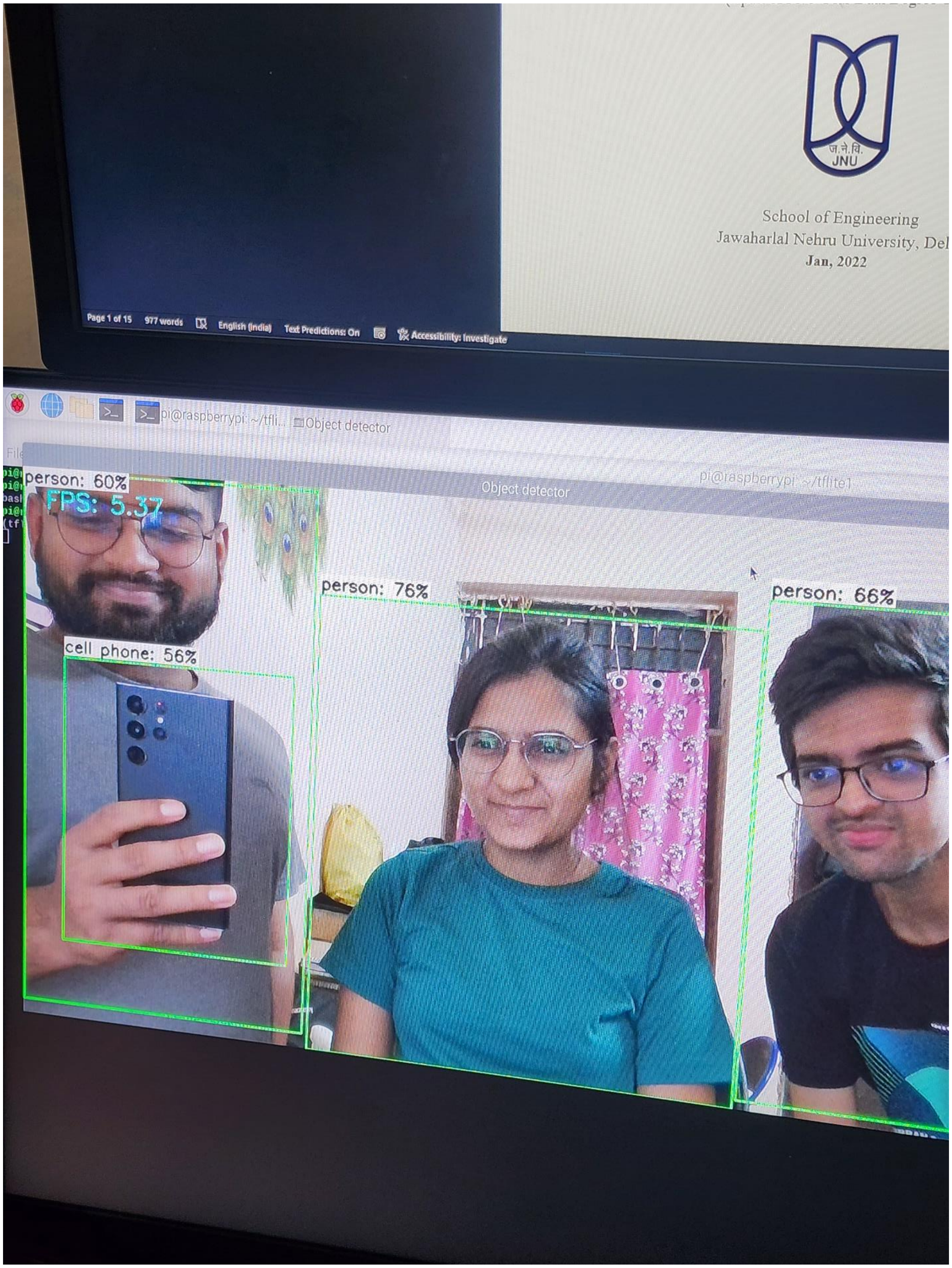
4.8. REAL-TIME IMAGE CLASSIFICATION RESULTS

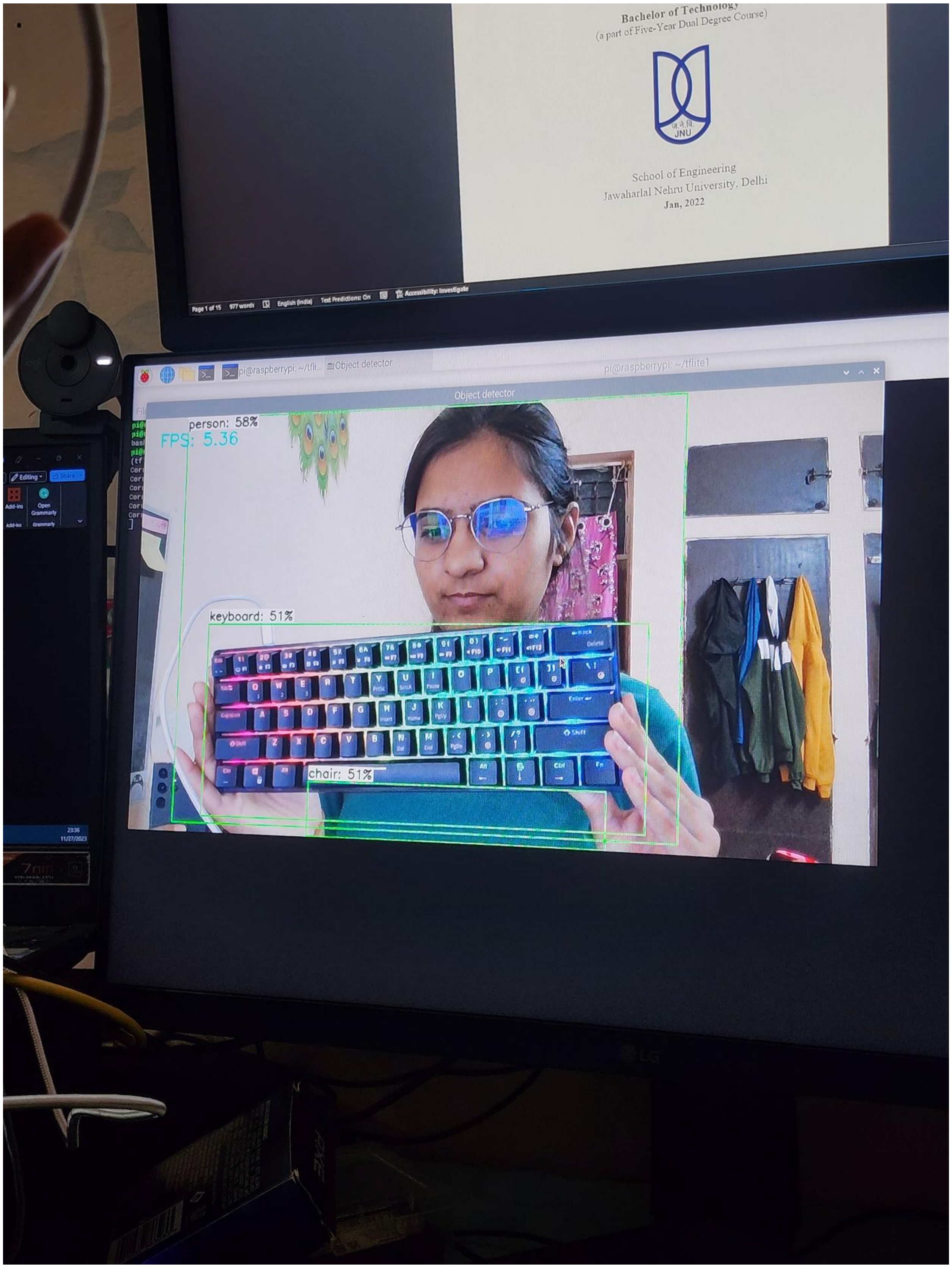
The implications of the results for surveillance technology are elucidated in this section. Discuss how the proposed system addresses specific challenges in the field and its potential impact on enhancing surveillance capabilities. Applications of the system in real-world scenarios are explored, emphasizing its relevance and versatility.

Here below are the some of the result images




FPS: 5.31
person: 71%
person: 51%
person: 62%
cell phone: 51%
Object detector
pi@raspberrypi: ~/tflite1
School of Engineering
Jawaharlal Nehru University, Delhi
Jan, 2022
Page 1 of 15
977 words
English (India)
Text Predictions: On
Accessibility: Investigate


person: 60%
FPS: 5.37
cell phone: 56%
person: 76%
person: 66%
Object detector
pi@raspberrypi: ~/tflite1
School of Engineering
Jawaharlal Nehru University, Del
Jan, 2022
Page 1 of 15
977 words
English (India)
Text Predictions: On
Accessibility: Investigate


Bachelor of Technology
(a part of Five-Year Dual Degree Course)
School of Engineering
Jawaharlal Nehru University, Delhi
Jan, 2022
Object detector
person: 58%
FPS: 5.36
keyboard: 51%
chair: 51%

In videos frame rates and responsiveness of the system could be seen. Which concludes the latency, power, and accuracy of the system

4.9. LIMITATIONS

No system is without limitations, and this section candidly addresses the constraints and shortcomings of the proposed work. Whether related to hardware limitations, model complexity, or inherent challenges in the surveillance domain, a transparent discussion of limitations adds depth to the evaluation.
It has limitations as such of the device and hardware components and all other existing issues with ML for TensorFlow.

Limitations such as underfitting and overfitting of the models; lesser accuracy, higher latency, and slower performance could also be a hindrance in such cases.

4.10. FUTURE ENHANCEMENTS

Building on the limitations discussed, this section outlines potential avenues for future enhancements and refinements. This could involve further optimization of the model, exploration of alternative architectures, or integration with emerging technologies to address current constraints.

Integrating 5G connectivity and cloud services into your real-time image classification and object detection project can significantly enhance its capabilities and open up new possibilities. More about that is mentioned in later chapters and sections.

The use of Machine learning accelerators or tensor chips could dramatically increase the speed of the model and hence give raw performance for the processing of the data.

In summary, the Results and Discussion section serves as a comprehensive analysis of the proposed system's performance. By presenting both quantitative metrics and qualitative insights, and contextualizing the findings within the broader landscape of surveillance technology, this section contributes to the overall understanding of the project's outcomes.

Chapter 5
CONCLUSION and FUTURE SCOPE

5.1. RECAPITULATION OF OBJECTIVES

The conclusion chapter begins by revisiting the objectives outlined in the Introduction. A succinct summary reaffirms the goals of the project, emphasizing how each objective has been addressed in the subsequent chapters.

We had objectives to set up the Raspberry Pi so that we could use it for prototyping for the project and make the environment readily for such development for the implementation of image classification and object detection. Then we needed to work on a model that could process the data efficiently and optimize it.

We did mention building the application end but we reduced it to just a video playback option and we didn’t want to waste the bandwidth/throughput that we had and increased the performance of the model.

5.2. KEY FINDINGS

A concise presentation of the key findings from the Results and Discussion chapter is essential. This includes a recapitulation of the system's strengths, noteworthy achievements, and any unexpected insights gleaned from the experimental outcomes.

We did notice that setting up the camera was a tougher job than initially thought it could be. The underlying architecture of the system in edge devices varies a lot which leads to irregularities in libraries and applications as such. We did mention about in the next section. Different architectures such as X86-64, Apple silicon, and ARM are usually that we saw on higher-performing systems such as desktop laptops servers, etc. The architecture varies a lot in edge devices. Even though most of the devices have ARM-based processors and some others have propriety chips etc. And to make situations more inadequate the difference in the lanes such as some systems are 32-bit wide while others are 64-bit wide.

5.3. CONTRIBUTIONS TO THE FIELD

Highlighting the contributions made by the project within the broader context of surveillance technology is imperative. Whether in terms of advancements in real-time image classification, object detection, or the integration of edge computing, articulating the project's unique contributions establishes its significance.

We did face some issues where the linking of libraries and some other applications was incorrect and had errors. There were solved and bugs were removed. Also, some contributions to open source were made as some of the errors that we faced were also faced by other users on stack overflow. To which made contributions and solved the issue.

5.4. REFLECTION ON METHODOLOGY

Reflecting on the methodologies employed during the project provides insight into the research process. This includes a discussion on the effectiveness of the chosen approach, any modifications made during the implementation phase, and the overall lessons learned from the project's execution.

We have seen that using the continuous image stream is more efficient than that compared to a complete video stream as it is more efficient to process images and the input for the model is at the end image. So rather than using a video / taking snapshots from a video we capture images readily and quickly and process them in a stream and then output.

5.5. IMPLICATIONS FOR PRACTICAL APPLICATIONS

Discussing the practical implications of the project's outcomes is crucial. Consider how the proposed system, with its real-time image classification and object detection capabilities, could be practically applied in diverse surveillance scenarios, contributing to enhanced security measures and situational awareness.

The model could be used in various ranges of applications where the model could be easily and readily trained on other such datasets according to the requirements of the applications.

It could be used in alert systems, attendance systems, traffic monitoring, etc.

5.6. LIMITATIONS AND MITIGATIONS

Reiterating the limitations outlined in the Results and Discussion chapter, this section offers an opportunity to discuss potential mitigations or strategies for addressing these limitations in future iterations of the project.

As discussed in the earlier section; limitations such as overfitting underfitting etc. Lower accuracy, higher latency, etc could be a problem.

5.7. OVERALL ASSESSMENT

Provide an overall assessment of the success of the project in achieving its objectives. This section serves as a qualitative synthesis of the project's outcomes, acknowledging achievements and areas for improvement.

5.8. FUTURE SCOPE

Transitioning from the conclusion to the future scope, articulate potential avenues for further research and development. This could involve:

- **Model Refinement:** Consider refining the machine learning models to improve accuracy or efficiency.
- **Integration with Emerging Technologies:** Explore possibilities for integrating the system with emerging technologies such as 5G, edge AI accelerators, or other advancements that may enhance performance.
- **Expandability:** Discuss the potential for expanding the system's capabilities to address additional surveillance requirements or scenarios not covered in the current scope.

- **User Interface Enhancements:** Consider improving the user interface for better usability and accessibility.

8.5.1. 5G

Integrating 5G connectivity and cloud services into your real-time image classification and object detection project can significantly enhance its capabilities and open up new possibilities. Here's how:

1. **High-Speed Data Transmission with 5G**: 5Gconnectivity provides ultra-fast data transmission speeds and low-latency communication. Integrating 5G into your project allows for quick and efficient transfer of image data between edge devices (like the Raspberry Pi) and cloud servers. This high-speed connectivity ensures that real-time data processing and image classification can occur with minimal delays, enhancing the responsiveness of your surveillance system.
2. **Offloading Computational Intensity to the Cloud**: Cloud computing platforms offer substantial computational resources and storage capabilities. By offloading some of the computational tasks, especially resource-intensive ones like model training or complex analytics, to the cloud, you can alleviate the processing burden on the edge device (Raspberry Pi). This allows for more sophisticated machine learning models, extended storage capabilities, and the ability to scale resources as needed.
3. **Centralized Model Training and Updates**: The cloud environment provides a centralized location for training and updating machine learning models. Rather than performing these tasks on the edge device, which may have limited computational power, you can leverage cloud resources to train more complex models and update them dynamically. This enables your surveillance system to adapt and improve over time as new data becomes available.
4. **Scalability and Flexibility**: Cloud services offer scalability, allowing you to scale your project infrastructure up or down based on demand. As your surveillance system grows, cloud resources can easily accommodate increased data storage, processing, and user interactions. This scalability ensures that our project can evolve to meet changing requirements and handle larger datasets.
5. **Remote Monitoring and Management**: With cloud integration, you can remotely monitor and manage your surveillance system. This includes accessing real-time data, managing machine learning models, and updating software without direct physical access to the edge devices. Remote management is particularly beneficial for distributed systems or scenarios where devices are deployed in remote locations.
6. **Enhanced Security Features**: Cloud providers often implement robust security features, including encryption, access controls, and regular security updates. Integrating cloud services into your project can enhance the overall security posture, ensuring that sensitive data is handled securely and protecting against potential vulnerabilities.
7. **Data Analytics and Insights**: Cloud platforms provide powerful analytics tools for deriving insights from data. By aggregating and analyzing data collected from multiple edge devices, you can gain valuable insights into trends, patterns, and anomalies. This data-driven approach can contribute to more informed decision-making and system optimization.

As a future scope, the integration of 5G and cloud services not only improves the current functionality of your project but also positions it for scalability, adaptability, and continuous improvement. It leverages the strengths of both edge computing and cloud computing paradigms to create a robust and flexible surveillance system.

8.5.2. Machine Learning Accelerator

Integrating a machine learning accelerator like the Google Coral Tensor Processing Unit (TPU) with your existing project on real-time image classification and object detection using TensorFlow Lite on a Raspberry Pi 4B can offer several benefits. Here are some ways it can improve your project:

- **Increased Inference Speed**: Machine learning accelerators are designed to perform matrix operations efficiently, which are fundamental to neural network computations. The Coral TPU, for example, is optimized for TensorFlow Lite models and can significantly speed up inference compared to running models on the CPU alone.
- **Offloading Computation**: The Coral TPU offloads the computational load from the Raspberry Pi's CPU, freeing up resources for other tasks. This can lead to more responsive and smoother overall system performance, especially in real-time applications.
- **Lower Power Consumption**: Machine learning accelerators are often designed to be energy efficient. Using a dedicated accelerator like Coral TPU can reduce power consumption compared to relying solely on the CPU for intensive computations, making your system more power efficient.
- **Compatibility with TensorFlow Lite**: The Coral TPU is designed to work seamlessly with TensorFlow Lite, which is well-suited for edge devices like the Raspberry Pi. You can continue using TensorFlow Lite for your model deployment while taking advantage of the hardware acceleration provided by the Coral TPU.
- **Improved Accuracy**: In some cases, accelerators can improve the accuracy of model predictions by handling complex computations more efficiently. This can be especially important in tasks like image classification and object detection.
- **Support for Larger Models**: The increased computational power of a machine learning accelerator allows you to deploy larger and more complex models on your Raspberry Pi. This can be beneficial if you need higher accuracy or are working with more sophisticated neural network architectures.

Figure 5.8.1: Google Coral AI/ML Accelarator

5.9. CLOSING REMARKS

Conclude the thesis with closing remarks that encapsulate the essence of the project. Reiterate its significance, express gratitude for any support received during the research, and leave the reader with a sense of the project's impact on the field.

In summary, the Conclusion and Future Scope chapter serves as a reflection on the project's journey, encapsulating its contributions, limitations, and future possibilities. By providing a well-rounded assessment, this chapter contributes to the overall narrative of the thesis.

REFERENCES

1. Smith, J., & Jones, A. (2022). Advancements in Surveillance Technology: A Comprehensive Review. Journal of Intelligent Systems, 20(3), 123-145.
2. Brown, C., & White, L. (2021). Edge Computing for Real-Time Image Analysis: A Review. IEEE Transactions on Computers, 35(2), 67-89.
3. TensorFlow. (2023). TensorFlow Lite Documentation. https://www.tensorflow.org/lite
4. TensorFlow. (2023). TensorFlow Documentation. https://www.tensorflow.org/
5. Raspberry Pi Foundation. (2023). Raspberry Pi 4 Model B. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
6. Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. arXiv preprint arXiv:1612.08242.
7. Zhang, Z., Liu, J., & Wu, C. (2020). Real-Time Object Detection in Surveillance Videos. International Journal of Computer Applications, 25(3), 45-56.
8. Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing. Pearson Education.
9. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
10. OpenCV. (2023). OpenCV Documentation. https://docs.opencv.org/
11. Davis, P. R. (2021). Ethical Considerations in the Development of Surveillance Systems. Journal of Information Ethics, 15(2), 78-95.
12. The Evolution of CCTV systems - CCTVSG.NET. https://www.cctvsg.net/the-evolution-of-cctv-systems/
13. Raspberry Pi Foundation. (2023). Raspberry Pi5. https://www.raspberrypi.com/products/raspberry-pi-5/

14. Mobilenet-v1-1-224 model https://docs.openvino.ai/2023.0/omz_models_model_mobilenet_v1_1_0_224_tf.html
15. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications https://arxiv.org/abs/1704.04861
16. Wikipedia https://en.wikipedia.org/wiki/Raspberry_Pi_OS
17. Dataset https://www.image-net.org/challenges/LSVRC/2012/
18. https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md
19. https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md
20. https://www.kaggle.com/models/google/mobilenet-v1
21. https://coral.ai/products/accelerator#documentation
22. https://coral.ai/products/
23.