MULTI-LAYERED ARCHITECTURE FOR ANOMALY DETECTION IN SURVEILLANCE NETWORKS

CSD416 Project Phase II

MDL19CS005 CSU19B03 ADITHYA A
MDL19CS008 CSU19B05 ALDRIN JENSON
MDL19CS105 CSU19B23 GOURI HARIHARAN
MDL19CS120 CSU19B37 NAYANA VINOD
B. Tech Computer Science & Engineering

Department of Computer Engineering
Model Engineering College
Thrikkakara, Kochi 682021
Phone: +91.484.2575370
http://www.mec.ac.in
hodcs@mec.ac.in

June 2023

Model Engineering College Thrikkakara
Department of Computer Engineering

C E R T I F I C A T E

This is to certify that, this report titled ***Multi-Layered Architecture for Anomaly Detection In Surveillance Networks*** is a bonafide record of the work done by

MDL19CS008 CSU19B05 ALDRIN JENSON

Eighth Semester B. Tech. Computer Science & Engineering

students, for the course work in **CSD416 Project Phase II**, which is the second part of the two semester project work, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, B. Tech. Computer Science & Engineering of **APJ Abdul Kalam University**.

Guide
Dr. Preetha Theresa Joy
Professor
Department of Computer Engineering
Model Engineering College

Coordinator

Dr. Preetha Theresa Joy
Professor
Department of Computer Engineering

Head of the Department

Dr. Preetha Theresa Joy
Professor
Department of Computer Engineering

June 15, 2023

Acknowledgements

This project would not have been possible without the kind support and help of many individuals. We would like to extend my sincere thanks to all of them.

First of all, We would like to thank our esteemed Principal, Prof. (Dr.) Jacob Thomas V, for his guidance and support in maintaining a calm and refreshing environment to work in and also for providing the facilities that this work demanded.

We are highly indebted to our Project Coordinator, Project Guide and the Head of the Department, Dr. Preetha Theresa Joy, Professor for her guidance, support and constant supervision throughout the duration of the work as well as for providing all the necessary information and facilities that this work demanded.

We offer our sincere gratitude to all our friends and peers for their support and encouragement that helped us get through the tough phases during the course of this work.

Last but not the least, we thank the Almighty God for guiding us through and enabling us to complete the work within the specified time.

Aldrin Jenson

Abstract

Due to limited performance of manual monitoring, law enforcement agencies are having difficulty in capturing or preventing anomalous incidents. An optimized and feature-based, intelligent, anomaly detection framework which operates in Convolutional Neural Networks(CNN) with bi-directional Long-Short-Term Memory(LSTM) and Resnet50 Models with reduced time complexity can be used to make it more efficient and accurate. Horizontal scaling can be used to ensure that requests are handled efficiently in edge nodes. Thus, the project aims to create a multi-layered system for processing multiple live feeds on a cloud computing platform while ensuring easy scalability without compromising on performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Proposed Project

The project aims to create a multi-layered system for processing multiple live CCTV feeds on a cloud computing platform while ensuring easy scalability without compromising on performance using CNN and Bi-Directional LSTM model.

1.1.1 Problem Statement

"Over 460,000 crimes are reported accross the coutnry every year. This is due to the lack of proper surveillance and policing. How can we build a platform for processing multiple live CCTV feeds and automatically detecting crimes and alerting the first responders while ensuring easy scalability without compromising on performance "

1.1.2 Proposed Solution

We propose an optimized and feature-based, intelligent, anomaly detection framework which operates in Convolutional Neural Networks(CNN) with bi-directional Long-Short-Term Memory(LSTM) and Resnet50 Models with reduced time complexity can be used to make it more efficient and accurate. Horizontal scaling can be used to ensure that requests are handled efficiently in edge nodes. Thus, the project aims to create a multi-layered system for processing multiple live feeds on a cloud computing platform while ensuring easy scalability without compromising on performance.

1.1.3 Objectives

The main objectives are:

- Allows anomaly detection on public CCTVs to report offenses live to the authorities.
- The admin dashboard contains info-graphics about the number of alerts and tips received along with the respective heat map.
- The CCTV IP Cameras stream live video feeds to the nearest edge node for processing.

Chapter 2

System Study Report

2.1 Literature Survey

1. **CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks :** The authors of this present an optimized and feature-based intelligent framework for efficient anomaly detection in CCTV surveillance networks. The proposed framework uses a Convolutional Neural Network (CNN) with bi-directional Long-Short-Term Memory (LSTM) and Resnet50 model to extract spatiotemporal features from a series of frames. These features are then passed to a multi-layer BD-LSTM model to accurately classify ongoing anomalous and normal events in complex surveillance scenes. Extensive experiments on various anomaly detection benchmark datasets were performed to validate the effectiveness of the proposed framework, with an accuracy of 85.3% on the UCF crime dataset and 89% on the UCFCrime2Local dataset. The proposed framework showed an improvement of 3.41% and 8.09% in accuracy compared to state-of-the-art methods.

2. **Unsupervised Anomaly Detection and Localization Based on Deep Spatiotemporal Translation Network :** A novel unsupervised anomaly detection and localization method is presented, called Deep Spatiotemporal Translation Network (DSTN) is presented in this paper. The DSTN method is based on a combination of Generative Adversarial Network (GAN) and Edge Wrapping (EW) techniques. The method is trained using only the frames of normal events, generating dense optical flow as temporal features. During testing, all video sequences are input into the system, and unknown events are considered anomalies because the model only knows normal patterns. To take advantage of both appearance and motion features, a fusion of background removal and real optical flow frames is used, along with a concatenation of the original and background removal frames. The performance of anomaly localization is improved at the pixel level by using the Edge Wrapping framework to reduce noise and suppress non-related edges of abnormal objects. The DSTN model has been tested on several publicly available anomaly datasets, including UCSD pedestrian, UMN, and CUHK Avenue. The results show that it outperforms other state-of-the-art algorithms in terms of frame-level and pixel-level evaluation, as well as time complexity for abnormal object detection and localization tasks. However, the model is computationally intensive and can be complex and costly to implement.

3. **Improved YOLOv5-Efficient Object Detection Using Drone Images under Various**

Conditions : In this paper, an improved YOLOv5 model was proposed to obtain high-accuracy object detection at various altitudes and weather conditions. The architecture of YOLOv5 was modified for better performance. The model was trained with VisDrone dataset and other drone captured images. Improved YOLOv5 model when tested achieved higher precision object detection than the existing RCNN networks and previous YOLO models. However, this model was trained and tested for only 3 categories of objects.

4. **A High-Performance Parallel Approach to Image Processing in Distributed Computing :** This paper presents two main contributions. The first is the use of ZeroMQ to separate video data streams into individual frames and distribute them among nodes. The second is the use of OpenMP on multi-core platforms for spectral analysis of images. ZeroMQ is a highly performant asynchronous messaging library that is well-suited for persistent distributed systems that need to be resilient to faults and network instability. OpenMP technology is used as an auxiliary tool to facilitate parallel processing. High processing speed is achieved through the use of a parallel DCT algorithm, resulting in good acceleration. The performance of this approach is observed to be particularly high in applications that follow the principles of stream processing.

5. **Semi-Distributed Load Balancing for Massively Parallel Multi-computer Systems :** In large parallel and distributed systems, load balancing is often done using centralized or completely distributed ways; however, this work provides a semi-distributed alternative. By dividing the interconnection structure of a distributed or multiprocessor system into independent symmetric zones (spheres) centered at some control points, the suggested technique employs a two-level hierarchical control. The center nodes, referred as as schedulers, keep state information with little overhead while scheduling work within their domains of influence optimally. The suggested technique has lower overhead in terms of control messages than a fully distributed strategy while still producing excellent performance in terms of response time and improved resource efficiency. It has also been demonstrated that it is less susceptible to the underlying network's communication delay.

6. **Live Video Analytics at Scale with Approximation and Delay-Tolerance :** This article provides an overview of VideoStorm, a video analytics system that handles a huge number of video analytics queries on live video streams. Resource management is essential since processing vision involves substantial expenses. As a result, the resource-quality trade-off with multi-dimensional configurations and the variety of quality and lag targets are two important aspects of video analytics. In contrast to the typically utilized fair sharing of resources in clusters, VideoStorm's offline profiler develops a query resource quality profile while its online scheduler allots resources to queries to optimum performance on quality and lag. The quality of real-world queries improved by up to 80% after being deployed on a 101-machine Azure cluster, while processing data from active traffic cameras showed a 7-percent reduction in lag.

7. **Video Analytics in Elite Soccer: A Distributed Computing Perspective :** In this paper, various methods of performing analysis on Soccer footage videos are mentioned along with an approach to parallelize the work load using distributed computing. A popular method of tracking is using Local Position Measurement(LPM) using GPS and Radio signals. A novel method to perform computations on the edge - Fog computing is also being considered.

8. **A Fog-Based Security Framework for Large-Scale Industrial Internet of Things Environments:** This paper explores the problems with the traditional cloud computing architecture for Industrial Internet of Things(IIoT) devices. Fog Layer computing is proposed as a better method to solve the issues along with protection from DDoS attacks in IIoT systems. However few concerns regarding authentication and energy concerns still remains to be explored.

2.2 Proposed System

The proposed system will use a Convolutional Neural Networks(CNN) with bi-directional Long-Short-Term Memory(LSTM) and Resnet50 Models to create a multi-layered architecture consisting of a distributed system for processing multiple live feeds on a cloud computing platform by dividing work among nodes and achieving parallelisation of anomaly detection. The system would include views for checking and verifying anomalies, an admin dashboard to see past anomalies, their location, time stamps, any false positives recorded, etc. The architecture would also include APIs for cameras and other sensors to connect and send camera data which will be processed and evaluated by the Resnet50 model. The system would be made efficient by using distributed computing over the edge.

Chapter 3

Software Requirement Specification

3.1 Introduction

3.1.1 Purpose

This document is for the purpose of explaining what our project is about. An optimized and feature-based, intelligent framework which operates in CCTV surveillance networks using Convolutional Neural Networks(CNN) with bi-directional Long-Short-Term Memory(LSTM) and Resnet50 Model can be used for efficient anomaly detection. The project aims to create a multi-layered architecture consisting of a distributed system for processing multiple live feeds on a cloud computing platform by dividing work among nodes and achieving parallelisation of anomaly detection.

3.1.2 Intended Audience and Reading Suggestions

This document contains software functionality, software and hardware requirements, and user documentation.

- **Developer:** The developer who wants to read, change, modify or add new requirements to the existing program may need first to consult this document and update the requirements in an appropriate manner so as not to change the actual purpose of the system or make the system inconsistent.

- **User:** The user of this program reviews the diagram and the specification provided in the document to determine whether the software has all the suitable requirements and if the software developer has implemented all of them. They can also consult the user guide in the event of any confusion for clarification.

- **Tester:** The tester needs this document to prepare his test cases to validate that the initial requirements of this project is actually implemented in the deliverable.

This document need not be read sequentially; users are encouraged to jump to any section they find relevant.

3.1.3 Project Scope

The project aims to create a multi-layered system for processing multiple live feeds on a cloud computing platform while ensuring easy scalability without compromising on performance using CNN and Bi-Directional LSTM model.

3.1.4 Overview of Developer's Responsibilities

- Analyze the problem and propose a solution.
- Choose a programming language which is most suitable and can support most of the libraries and can support of the libraries and frameworks required for the project
- After suitable programming language is chosen the developers must learn more about the language and the libraries and framework that are needed.
- Selection of appropriate libraries, frameworks, and their respective documentation.
- Implement the best algorithm chosen to solve the problem using the frameworks and libraries of choice.
- Test the system to ensure that it gives the desired results.
- The result should be presented in a way that is convenient and easy for the users to understand.

3.2 Overall Description

3.2.1 Product Perspective

An optimized and feature-based, intelligent framework which operates in CCTV surveillance networks using Convolutional Neural Networks(CNN) with bi-directional Long-Short-Term Memory(LSTM) and Resnet50 Model can be used for efficient anomaly detection. The project aims to create a multi-layered architecture consisting of a distributed system for processing multiple live feeds on a cloud computing platform . The authorities can leverage live CCTV anomaly detection to track offenses in public areas.

3.2.2 Product Functions

This project aims to provide the following:

- Allows anomaly detection on public CCTVs to report offenses live to the authorities.
- The admin dashboard contains info-graphics about the number of alerts and tips received along with the respective heat map.
- The CCTV IP Cameras stream live video feeds to the nearest edge node for processing.

The different components of the system will be:

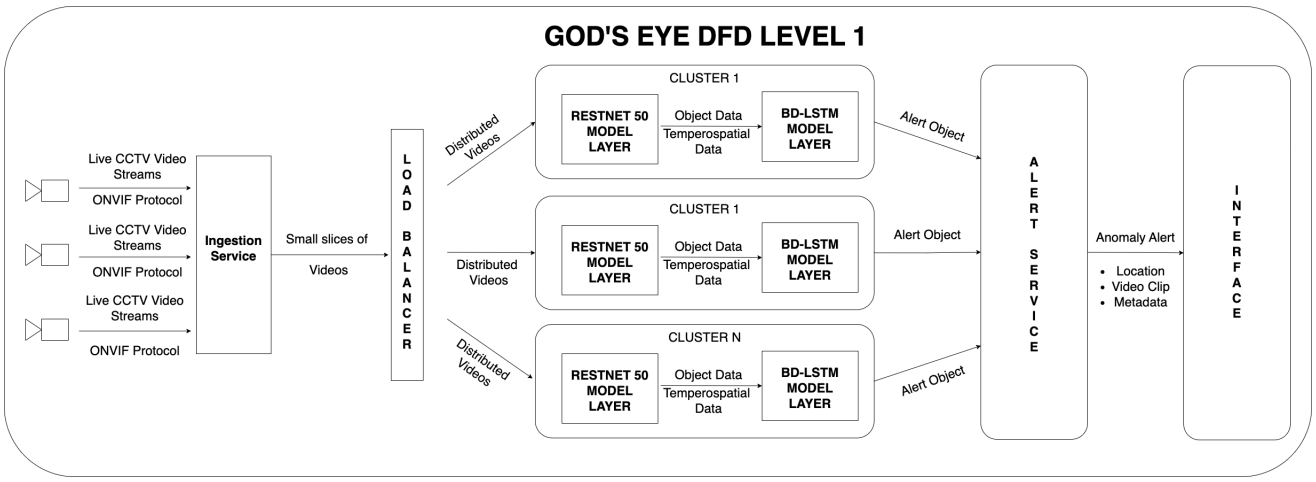


Figure 3.1: System Architecture

3.2.3 Operating Environment

Porject requires a Linux, Windows or Mac OS based server that supports CUDA, Javascript, Python and PostgreSQL.

3.2.4 Design and Implementation Constraints

- Using this system is fairly simple and intuitive. A user familiar with basic browser navigation skills should be able to understand all functionality provided by the system.
- The system should work on most home desktop and laptop computers which support JavaScript and HTML5.
- The system will be intended to run on Firefox 4 and above, Google Chrome 10 and above and Internet Explorer 8 and above.
- System shall be able to interface with other components according to their specifications.
- The system is limited by its operating server in terms of the maximum number of users it can support at a given time.

3.2.5 User Documentation

- User documentation will consist of the several components usually expected of a modern web-based software application, including a tutorial, help pages and FAQs with an online request form.
- Detailed user's manual will be provided in a ReadMe file in the GitHub repository
- https://docs.python.org/3
- https://flask.palletsprojects.com/en/2.2.x
- https://www.tensorflow.org/api-docs

3.2.6 General Constraints

The following are the general constraints of the project:

- More than one instance of video anomaly detection cannot be tested.
- Only registered users have the ability to access the platform.
- Certain anomalies cannot be replicated Eg: homicide.

3.2.7 Assumptions and Dependencies

- The project is to be hosted on a cloud provider such as Google Cloud or AWS and uses a cloud based GPU provider for its highly scalable GPUs.
- It is assumed that a clear video stream is received as input.

3.3 External Interface Requirements

3.3.1 User Interfaces

Gods Eye User Interface - Admin Dashboard for centralized management:

The admin dashboard should contain infographics about the number of alerts and tips received along with a heatmap of places where the alerts were received. It should show the contact details of active volunteers.

- User Authentication using Supabase Auth.
- Dashboard includes :
    1. Map with all live alerts taken from 'Alerts' table.
    2. 'Alerts' page displaying all alerts as cards.
    3. 'Cameras' with data of all cameras in 'Cameras' table.
    4. 'Responders' with details of all authorised responder profiles as cards.
    5. 'Reports' showing Alert and anomaly detection data yearwise.
    6. 'Account' with personal account information of user.

3.3.2 Hardware Interfaces

- CCTV IP Cameras connected to the network. The CCTV IP Cameras that are connected to the network using IP protocol, stream live video feeds to the nearest edge node for processing.

3.3.3 Software Interfaces

- Video Ingestion Interface - Python based NVR system that ingests parallel video streams from multiple network attached CCTV cameras. This will be an edge layer and will be located nearest to the hardware node.
- Load Balancer Layer - The video streams form the NVR layer is digested by the Load Balancing Layer and the streams are distributing parallely across the distributed system network. The load balancing will be done by a Kubernetes based Engine.
- Distributed Computing Layer - This layer will consist of heterogeneous computing clusters that are distributed across several regions running a container based instance of the Anomaly Detection model. The containers will be managed and orchestrated by the Docker Swarm resource manager.
- Model Layer - This is the layer where the main anomaly detection takes place. This consists of a pre-trained RestNet50 + BD-LSTM model that runs on its own container. There can be multiple instances of this container running on a particular computing node and these can be run over multiple computing nodes. This takes the live CCTV feeds assigned to it by the load balancer, performs anomaly detection, and outputs alert triggers to the alert layer if any anomaly is detected.

- Alert Layer - The layer responsible for listening to the Anomaly detection alert triggers and passing this on to the respective police control center.
- User Interface - This is the primary point of interaction between the authorities and the system. The control center authorities would be able to monitor and respond to alerts as per requirement. The First responders would have their own interface to accept the alerts. The admin will have an admin dashboard to see the working of the system.

3.3.4 Communications Interfaces

- ONVIF Protocol based on IP - Open Network Video Interface Forum Protocol works on top of Internet Protocol to send live CCTV videos streams over the network onto our ingestion service.
- REST API - The main communication interface between User Interface and the server. Works based on HTTP requests.
- Webhooks - Used to send alerts from the Alert Service to the User Interface.

3.4 Hardware and Software Requirements

3.4.1 Hardware Requirements

- Server with GPU - A server with at least 6 CPU cores, and 8GB NVIDIA CUDA based GPU.
- CCTV IP cameras - CCTV IP Cameras connected to the network. Must be of sufficient clarity and mounted on a stable support.

3.5 Software Requirements

- Server running Linux - A server with which can run Linux or any UNIX based operating system with sufficient support for NVIDIA CUDA based GPU.
- Containerization Support - The server should support containerization and orchestration using Docker.

3.6 Functional Requirements

3.6.1 Functional requirement 1 - Public violence and Anomaly detection

- 24/7 access to raw CCTV footage from public cameras.
- Be able to perform continuous video analysis
- Be able to detect of anomalies using custom algorithms.
- Be able to send alerts to Admin in case an anomaly is detected.
- Maintain logs of all the anomalies having a confidence score above 80% along with their timestamps.

3.6.2 Functional requirement 2 - Central Admin Dashboard with Authentication

- There should be a central admin console in which an alert message should be given whenever a new anomaly is being detected.
- There should be proper authentication enabled so that only authorized personnel can access it.
- Show all anomalies having confidence score above 90% in the admin dashboard.
- The admin should have the option to accept or reject a detection as anomaly or not and the system should be able to learn and improve based on the admin's suggestions.
- The dashboard should show graphs, heat-maps and other charts visualising the location, time and severity of the anomalies detected for easy analysis
- The Dashboard should also have necessary features required for the admin to contact healthcare workers, police officers and other government officials immediately depending on the severity of the anomaly.

3.6.3 Functional requirement 3 - Caching and Edge computing

- The first layer of computing should be done closest to the source of the video streams as to reduce the upstream load.
- The system should be able to cache the video streams for faster processing.
- A snapshot of a single frame of the video clip where anomaly is detected to be maintained in cache for investigation and logging.

3.7 Non-functional Requirements

3.7.1 Performance Requirements

The project should satisfy the following performance requirements:

- The Software system must be online 24/7.
- The system should be able to support thousands of concurrent users.
- The system should be able to handle processing videos from hundreds of cameras at the same time.
- The system should allow for maintainability and should have features for easy enhancements and updates.
- The system should be able to scale elastically depending on workload.
- System should be highly reliable and easily be able to recover from failures.
- The code should be optimized for high response times.
- The system should be able to run successfully in all modern web browsers.

3.7.2 Safety Requirements

- The system should show terms conditions and privacy policy to the user on the login page .
- Only minimum amount of data must be collected from the user.
- e-KYC to be done by the user using their Aadhar card and a background check must be done on the users and volunteers to ensure that they are actual responsible citizens.

3.7.3 Security Requirements

- The data standards must follow PCI-DSS compliance
- The terms and conditions for the user must be legally binding.
- There must be a privacy policy for the user declaring which all data are collected and stored.
- All data stored must be PCI-DSS compliant.
- Any identify or proof that we take must be masked and the number should be hidden or tokenized before storage.
- Multiple levels of encryption must be securely done on the data to prevent missuse of data.
- Tokenization and encryption practices followed should be compliant with later standards.
- The encryption method to be followed is AES256 encryption.

3.7.4 Software Quality Attributes

- Features for administration should be provided by the system.
- There should be features for intervention and monitoring any faults that may occur within the system.

Chapter 4

System Design

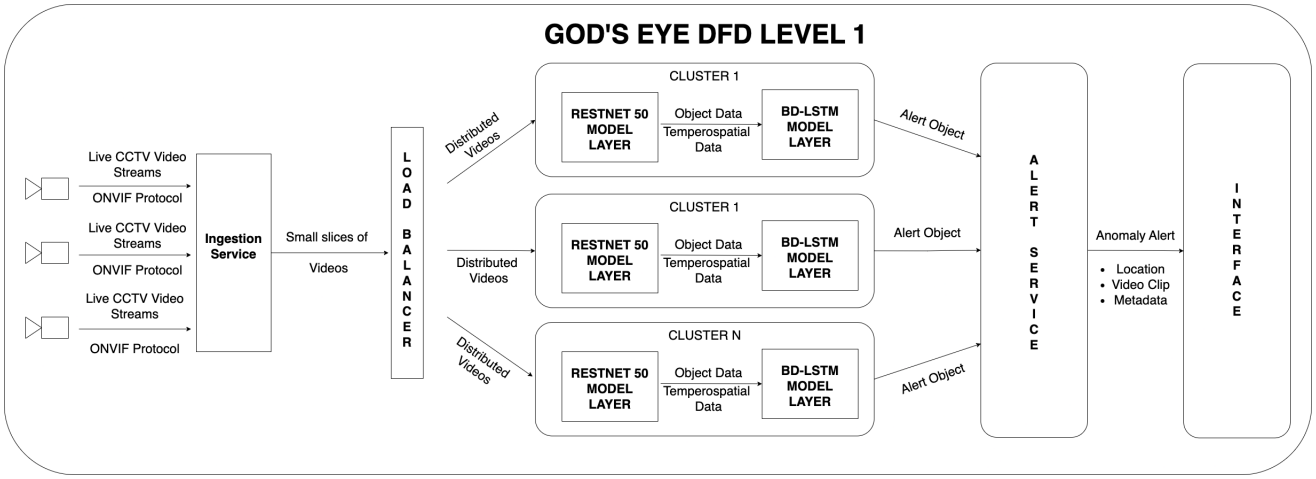


Figure 4.1: System Design

Chapter 5

System Architecture

5.1 Video Ingestion Interface

Python based NVR system that ingests parallel video streams from multiple network attached CCTV cameras. This will be an edge layer and will be located nearest to the hardware node.

5.2 Load Balancer Layer

The video streams form the NVR layer is digested by the Load Balancing Layer and the streams are distributing parallely across the distributed system network. The load balancing will be done by a Kubernetes based Engine.

5.3 Distributed Computing Layer

This layer will consist of heterogeneous computing clusters that are distributed across several regions running a container based instance of the Anomaly Detection model. The containers will be managed and orchestrated by the Kubernetes resource manager.

5.4 Model Layer

This is the layer where the main anomaly detection takes place. This consists of a pre-trained RestNet50 + BD-LSTM model that runs on its own container. There can be multiple instances of this container running on a particular computing node and these can be run over multiple computing nodes. This takes the live CCTV feeds assigned to it by the load balancer, performs anomaly detection, and outputs alert triggers to the alert layer if any anomaly is detected.

5.5 Alert Layer

The layer responsible for listening to the Anomaly detection alert triggers and passing this on to the respective police control center.

5.6 User Interface

The primary point of interaction between the authorities and the system would be the admin dashboard. The control center authorities would be able to monitor and respond to alerts as per requirement. The First responders would have their own interface to accept the alerts as well, but the feature rich admin dashboard will be our most important User facing interface.

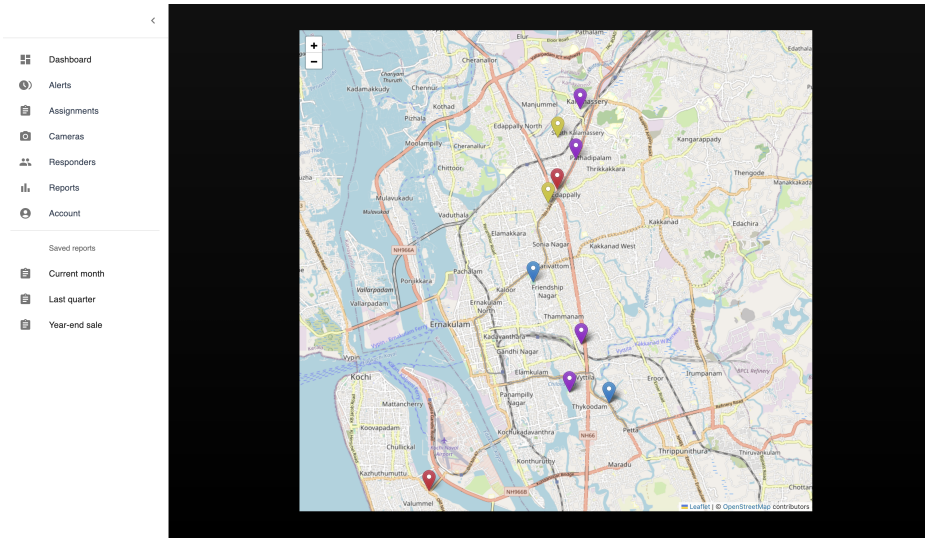


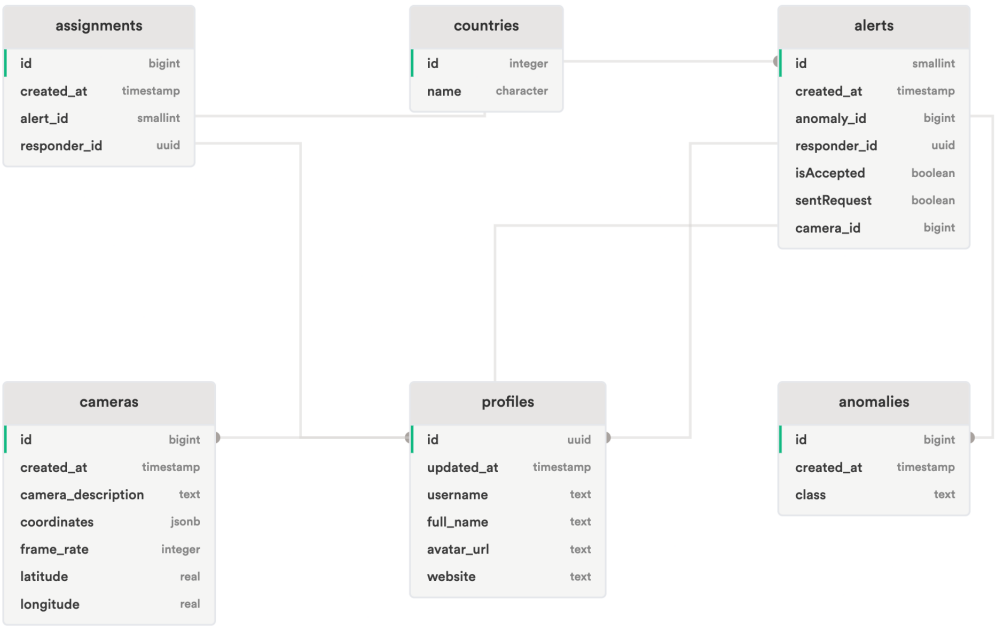
Figure 5.1: Map UI

Chapter 6

Data Description

6.1 Database design

The db design, entities and the schema.



6.2 Libraries and Packages Used

- **Axios:** Axios is a popular JavaScript library used in React for making HTTP requests from the client-side to interact with APIs and fetch data asynchronously.
- **Cookies:** The cookies library in React is a lightweight JavaScript library that simplifies the handling and management of HTTP cookies in React applications, providing convenient methods for reading, writing, and manipulating cookies.

- **React-Leaflet:** React-Leaflet is a React library that provides a simple and declarative way to integrate Leaflet maps into React applications, allowing for interactive and customizable map components.
- **Open AI CLIP:** a neural network by OpenAI which efficiently learns visual concepts from natural language supervision. CLIP can be applied to any visual classification benchmark by simply providing the names of the visual categories to be recognized, similar to the "zero-shot" capabilities of GPT-2 and GPT-3.
- **OpenStreetMaps API:** The OpenStreetMap API in React provides access to open-source map data and allows developers to incorporate interactive maps, geocoding, and routing functionality into their React applications.
- **Supabase:** an open source NoSQL datbase for building secure and performant Postgres backends with minimal configuration. allowing developers to define schemas, perform database operations, and handle data validation seamlessly.

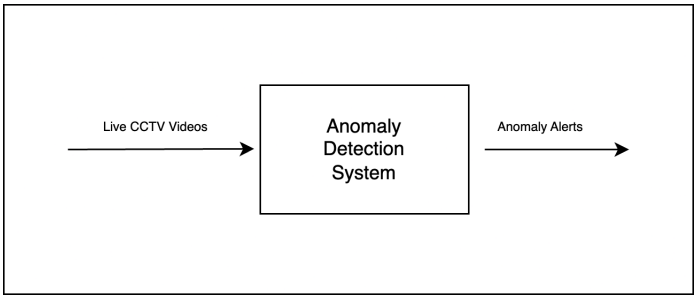
6.3 Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through a system modeling its process aspects. Often it is a preliminary step used to create an overview of the system that can later be elaborated. DFDs can also be used for the visualization of data processing (structured design) and show what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.

6.3.1 Level 0 DFD

Level 0 data flow diagram or context diagram is designed to be an abstract view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

- **DFD Level 0**



Chapter 7

Implementation

- **Implementation - Frontend**
  - **Text Stack** : Next.js with TypeScript
  - **API** : Supabase, OpenStreet Maps
  - **Architecture** :
    * User Authentication using Supabase Auth.
    * Dashboard includes :
      1. Map with all live alerts taken from 'Alerts' table.
      2. 'Alerts' page displaying all alerts as cards.
      3. 'Cameras' with data of all cameras in 'Cameras' table.
      4. 'Responders' with details of all authorised responder profiles as cards.
      5. 'Reports' showing Alert and anomaly detection data yearwise.
      6. 'Account' with personal account information of user.
- **Implementation - Backend Server**
  - **Text Stack** : Python Flask server with OpenAI Clip model for Anomaly classification
  - **API**: Created with Python Flask server and Supabase dashboard
  - **Architecture** :
    * Hosted instance of Supabase database connected to backend
    * Flask server listening for images frames in batches
    * Once a set of image batch has been received, pass these on to the model to be analysed for anomaly detection
    * Once the model detects the prescence of anomaly, identify the image frame in an anomalous time frame and pass these onto the OpenAI CLIP model to classify the type of anomaly
    * Once result has been processed, add this as a new entry to the database.
    * Repeat this indefinitely

    * Containerise these process files and handle scaling using docker-swarm.

- **Implementation - Ingestion Server**

  - **Text Stack** : Python process with OpenCV for image processing
  - **API**: HTTP API for sending image files to the primary backend server.
  - **Architecture** :
    * Listen to video frames from multiple cameras
    * Converts image frames from jpg or png formats to .tiff format using openCV
    * Groups frames into batches of 100 frames and then sends them to the primary backend API for processing.
    * Attaches timestamp and cameraId to image frames as metadata.
    * Loop this process indefinitely for all incoming camera feeds.
    * RUn this process using threads to ensure that there is independence between each camera feed.
    * Have option to also show a debug window to show the current live feed of any given camera using OpenCV library.

- **Implementation - Deep Learning Model**

  - **Text Stack** : Tensorflow, Keras, Numpy, Pandas, Scikit Learn.
  - **Architecture** :
    * Pretrained RESTNET-50 CNN model for object detection.
    * Bi Direcitonal LSTM Model with 3 LSTM layers.
    * Groups frames into batches of 100 frames and then sends them to the primary backend API for processing.
    * Attaches timestamp and cameraId to image frames as metadata.
    * Loop this process indefinitely for all incoming camera feeds.
    * RUn this process using threads to ensure that there is independence between each camera feed.
    * Have option to also show a debug window to show the current live feed of any given camera using OpenCV library.

7.1 Algorithms

7.2 Development Tools

1. **Node.js (version 18 or later):** Node.js is an open-source JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code outside of a web

browser, enabling server-side development. Node.js provides an event-driven, non-blocking I/O model that makes it lightweight and efficient, making it suitable for building scalable and high-performance applications.

2. **Node Package Manager (npm):** Node Package Manager (npm) is the default package manager for Node.js, allowing developers to easily discover, install, and manage third-party libraries and tools for their JavaScript projects. It is a command-line tool that facilitates package installation, version management, dependency resolution, and project configuration. With a vast repository of packages, npm enables developers to leverage existing code and libraries to accelerate development.

3. **Visual Studio Code (VS Code):** Visual Studio Code is a free and open-source source code editor developed by Microsoft. It is widely used by developers for various programming languages and supports a wide range of features such as code highlighting, IntelliSense (code completion), debugging, version control integration, and extensions for additional functionality. It provides a user-friendly interface, customizable settings, and a powerful ecosystem of extensions, making it popular among developers working on different platforms and projects.

4. **Postman:** Postman is a powerful API testing software that simplifies the process of testing and debugging APIs. It provides a user-friendly interface for sending HTTP requests, managing environments, and inspecting responses. With features like automated testing, request chaining, and response validation, Postman helps streamline API development and ensures the smooth functioning of APIs in various scenarios.

5. **Git and GitHub:** Git is a distributed version control system that allows developers to track changes, collaborate, and manage source code repositories efficiently. It enables multiple developers to work on a project simultaneously while keeping a complete history of changes, allowing for easy branching, merging, and version management. GitHub is a web-based hosting service that leverages Git for version control. It provides a platform for developers to host, manage, and share their Git repositories. GitHub offers additional features such as issue tracking, pull requests, project management tools, and collaboration functionalities, making it a popular choice for open-source projects and team collaboration.

Chapter 8

Testing

Software testing is a crucial process in software development that involves evaluating a software system or application to identify defects, errors, and ensure its quality and functionality. It includes various techniques such as unit testing, integration testing, system testing, and acceptance testing to verify that the software meets the specified requirements and works as expected. Testing helps uncover bugs, inconsistencies, and usability issues, enabling developers to fix them before the software is released. By ensuring the reliability, performance, and user satisfaction of the software, testing plays a vital role in delivering high-quality and reliable software solutions.

8.1 Testing Methodologies

Software Testing Methodology is defined as strategies and testing types used to certify that the Application Under Test meets user expectations. Test Methodologies include functional and non-functional testing to validate the Application Under Test. Each testing methodology has a defined test objective, test strategy and deliverable. Software testing methodology is for making sure that software products/systems developed have been successfully tested to meet their specified requirements and can successfully operate in all the anticipated environments with required usability and security. Software testing methods are traditionally divided into white and black box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases. White box testing by seeing the source code tests internal structures or workings of a program, as exposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit. While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. Black box testing treats the software as a black-box, examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Here the black-box testing used for the system. The testing methods applied were:

1. Unit Testing It is a software development process in which the smallest testable parts of an application called units are individually and independently scrutinized for proper operation.

2. Integration Testing It is the phase in software testing in which individual software modules

are combined and tested as a group. It occurs after unit testing.

3. System Testing System testing of software or hardware is type testing conducted on a complete, integrated system to evaluate the system's compliance with its specific requirements.System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

8.2 Unit Testing

Unit testing finds problems early in the development cycle. This includes both bugs in the programmer's implementation and flaws or missing parts of the specification for the unit. The process of writing a thorough set of tests forces the author to think through inputs, outputs, and error conditions, and thus more crisply define the unit's desired behaviour. The cost of finding a bug before coding begins or when the code is first written is considerably lower than the cost of detecting, identifying, and correcting the bug later; bugs may also cause problems for the end-users of the software. Each page on web app and module was tested and checked whether it is fully working and corrected the errors.

8.3 Frontend Test Cases

Test Case	Description	Preconditions	Test Steps	Expected Result	Status
Registering as Responder	A user who creates an account on the anomaly detection website is registered as a responder and has access to the interface	Ensure proper profile creation and confirmation	Sign up with Reponder mail id and password. Confirm via link sent to mail id to register as responder	Responder profile is created and added as card in page 'Responders'. Profile can be updated from 'Account' page	[Pass]
Fetch Alerts	An alert is created as soon as an anomaly is detected and displayed on the map and alerts page	Ensure that anomaly id, type, location supplied from the model are not null values	Select 'Dashboard' from side bar. Map with all alerts from the db is displayed. Click a marker to expand the alert card to view anomaly id, location and assign responder.	Map with all alerts from db is displayed. Each marker holds an alert card. All alert cards from db are also fetched onto 'Alerts' page.	[Pass]
Add Alert	Manually add alert from alerts page by supplying alert information and camera id	Ensure that camera and anomaly id existing in the database is supplied	Click 'Add Alert' in Alerts page. Supply valid alert, anomaly and camera id and optional responder id if assigned. Wait for alert card to be generated on Map and 'Alerts'	Alert is generated on map at camera location of supplied camera id. Marker colour of alert corresponds to supplied alert id. Alert card is displayed in 'Alerts' and on clicking marker.	[Pass]
Fetch Cameras	All cameras added into the database are displayed as cards on the 'Cameras' page	Ensure that camera data of location, latitude and longitude are not null values	Select 'Cameras' from sidebar	All cameras from database are displayed as cards with location, coordinates and frame rate	[Pass]
Add Camera	A camera can be added to the database from add camera option in page 'Cameras'	Ensure that camera data of location, latitude and longitude are not null values and give exact location coordinates	Click 'Add Camera' button on page 'Cameras'. Supply valid description, latitude, longitude and frame rate. Hit 'Add Camera' and wait for camera card to be displayed	As soon as data is entered a camera id is generated and added to the database. A corresponding camera card is created and displated in 'Cameras' page	[Pass]

Test Case	Description	Preconditions	Test Steps	Expected Result	Status
Assign Responder	Assign responder through Alert card in Map or 'Alerts' page	Responder needs to be registered in the site	Click 'Assign Responder' in alert card from Map or 'Alerts' page and choose responder name from drop down menu	Alert card updates with id of assigned responder in Map and 'Alerts' page	[Pass]

8.4 Backend Test Cases

Test Case	Description	Preconditions	Test Steps	Expected Result	Status
Accept multiple frames from Ingestion servers	Create API routes for being able to accept multiple image frames from different cameras/ingestion servers	Ensure that only image files are to be accepted	Run the main server. Use Postman or API testing services to upload multiple image files to the route. Test also using the ingestion servers.	Images are accepted and a 200 okay result is received as response	[Pass]
Anomaly Classifier successful classification	Testing of OpenAI CLIP anomaly classifier with the pre-defined 4 anomaly classes.	Run the main backend server. Ensure that only image files are accepted.	Pass in an image having the description under the 4 defined anomaly types. Wait a few seconds for the function to process and compare the final output class with the actual class group of image	The class name predicted should match the anomaly type of the image.	[Pass]
Connection to Database	Testing the connection of backend to Supabase Database	Ensure that Supabase Database URL and API keys are added as environment variable for the system to connect	Run the python main server. Check the logs to get the "Database connection successful message". Check if any error messages are found.	Successful database connection is made without any error messages.	[Pass]
Saving anomaly alerts to database	Once the model detects and classifies an anomaly, log that entry to the database.	The set of frames received are predicted as Anomalous and anomaly classification has been done on these frames.	Check database entries of "Anomaly" table before testing. Send a set of anomalous image frames to server. Once the server detects anomalies and classifies them, check database again to verify that a new entry has been made corresponding to this anomaly.	New database entry has been made corresponding to the new anomaly added.	[Pass]

8.5 Ingestion Server Test Cases

Test Case	Description	Preconditions	Test Steps	Expected Result	Status
Be able to connect to multiple cameras	The ingestion server should be able to work after being connected to multiple cameras. Test using internal device camera or using any IP based webcameras.	Ensure that cameras used support HTTP protocols.	Run the ingestion server. Ensure that the cameras are connected and the images frames are being processed. Verify that after every 10 seconds batches of frames are sent to primary backend API for processing.	A debug window should popup with live feed from the cameras connected. After every regular intervals, batch these images together and send it to backend for processing.	[Pass]
Conversion of PNG images to tiff	Ensure that the images received from the video are in the correct format required for the backend to process.	Images are being processed by multiple cameras and a debug window of current frames are being shown.	Connect one or 2 cameras to the ingestion server. Verify that the final images sent to the Primary Backend are in TIFF format.	The final images sent to Primary backend will be in Tiff format	[Pass]

Chapter 9

Graphical User Interface

The graphical user interface is a user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notations, instead of text-based user interfaces, typed command labels or text navigation.GUIs were introduced in reaction to the perceived steep learning curve of command-interfaces which require a command to be typed on a computer keyboard.

Designing the visual composition and temporal behaviour of a GUI is an important part of software application programming in the area of human-computer interaction. Its goal is to enhance the efficiency and use case for the underlying logical design of a stored program, a designed discipline name usability. Methods for user-centred design are used to ensure that the visual language introduced in the design is well-tailored to the tasks. For several decades GUIs were controlled exclusively by a mouse and keyboard. While these types of input devices are sufficient for desktop computers, they do not work as well for mobile devices, such as smartphones and tablets. Therefore, mobile operating systems are designed to use a touchscreen. Many mobile devices can now be controlled by spoken commands as well.

9.1 GUI Overview

A GUI uses a combination of technologies and devices to provide a platform that users can interact with, for the task of gathering and producing information. Traditionally, browsers acted themselves as independent and isolated software. All the features that were necessary for the browsers to perform, were implemented in browsers itself. Limitation of this method was that the users had no control over the performance of the browser, or at least they could not do anything to improve its performance.

Nowadays, browser performance can be improved by users, especially as the user demands. Users can either develop or get modules such as apps, extensions, plug-ins which can work within the browser. These modules are usually optional and can be installed to improve the user experience of the browser. These modules perform a specific task, which it is designed to do when the user asks to. For instance, there are browser apps which help to give users some reminder, browser extensions which can fill forms automatically, plugins which improve the visual effect of the browser

9.2 Main GUI Components

The main GUI components are:

1. Landing page: This page is the home page which mentions the details about the platform such as it's features and benefits.
2. Signup page: This is the page where the users can create new accounts by entering their details.
3. Login page: This is the page where the users can login using their credentials.
4. Dashboard page: This is the page that shows all the details that the user requires including the anomalies and metrics of the past quarter.
5. Anomaly listing page: This page displays the various details of the anomaly, including the person associated, date and place of ocurances etc , current status and a map showing the location of anomaly.

- **Homepage**

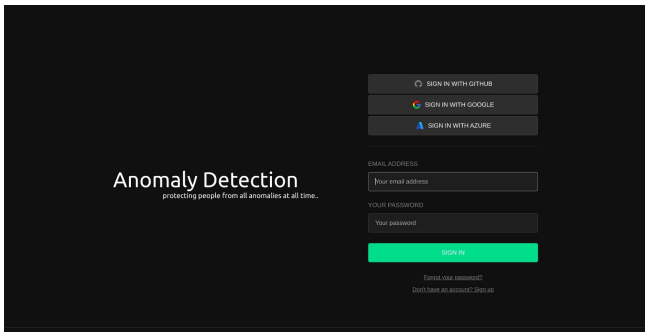


Figure 9.1: Home UI

.

- **Alerts Page**
- **Responders Page**
- **Reports Page**
- **Account Page**

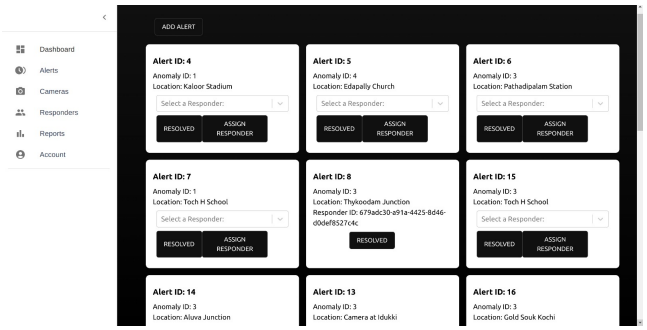


Figure 9.2: Alerts Page

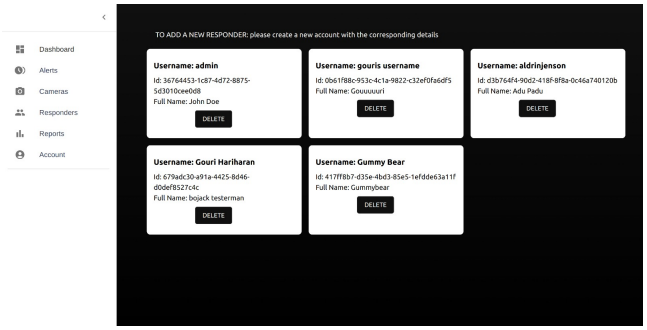


Figure 9.3: Responders UI

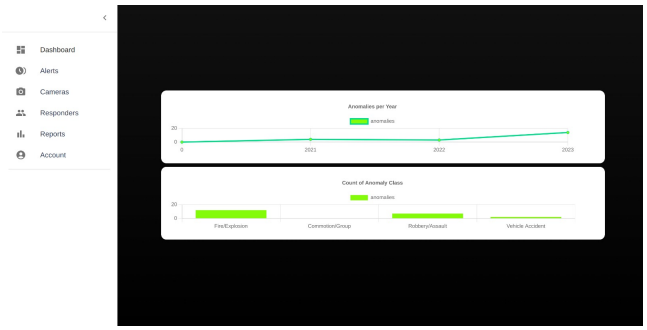


Figure 9.4: Reports UI

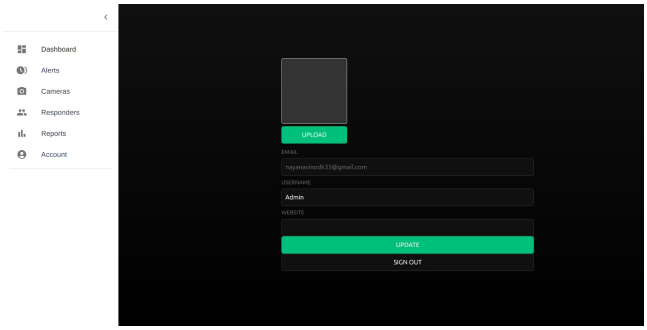


Figure 9.5: Account UI

Chapter 10

Results

- **Output of CNN + LSTM Model - Anomaly Detected**

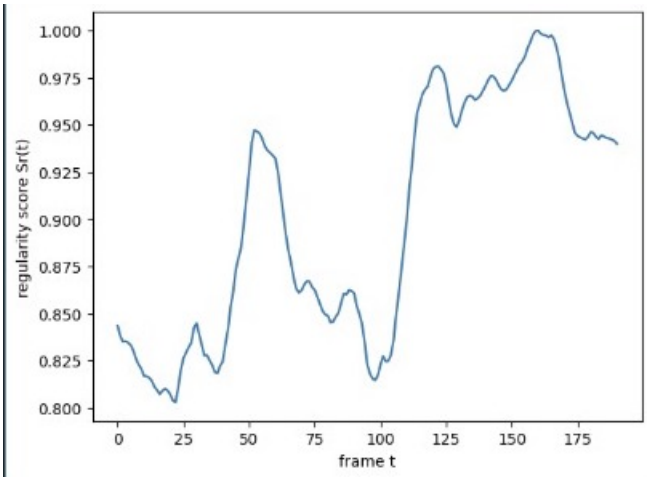


Figure 10.1: LSTM Output

- **Output of CNN + LSTM Model - Anomaly Not Detected**

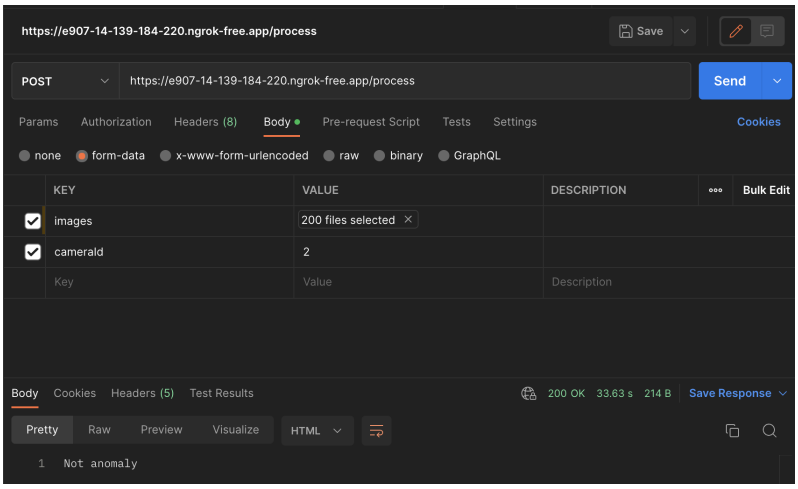


Figure 10.2: Postman

Figure 10.3: Confusion Matrix

Chapter 11

Conclusion

1. **An optimized efficient deep features-based intelligent anomaly detection framework using CNN with bi-directional LSTM and Restnet 50 Mode ls**
2. **Operate in CCTV surveillance networks with reduced time complexity using distributed systems.**

Chapter 12

Future Scope

Testing Horizontal Scaling of GPU load using GPU clusters.
Implementation of Raspberry Pi ONVIF processing hardware.
Mobile App for first responders to use to recieve and track alerts

References

[1] **CNN features with bi-directional LSTM for real-time anomaly detection in surveillance network** - Waseem Ullah,Amin Ullah,Ijaz Ul Haq,Khan Muhammad,Muhammad Sajjad,Sung Wook Baik, 2020

[2] **Unsupervised Anomaly Detection and Localization Based on Deep Spatiotemporal Translation Network** - Thittaporn Ganokratanaa, Supavadee Aramvith, Nicu Sebe - 2020

[3] **Improved YOLOv5: Efficient Object Detection Using Drone Images** - Hyun-Ki Jung and Gi-Sang Choi - 2022

[4] **A High-Performance Parallel Approach to Image Processing in Distributed Computing** - Mekhriddin Rakhimov, Doniyor Mamadjanov, Abulkosim Mukhiddinov - 2020

[5] **Semi-Distributed Load Balancing for Massively Parallel Multicomputer Systems** - Ishfaq Alunad and Arif Ghafoor - 1991

[6] **Live Video Analytics at Scale with Approximation and Delay-Tolerance** - Haoyu Zhang, Microsoft and Princeton University; Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, and Paramvir Bahl, Microsoft; Michael J. Freedman, Princeton University - 2017

[7] **Video Analytics in Elite Soccer: A Distributed Computing Perspective** - Debesh Jha, Ashish Rauniyar, H avard D. Johansen, Dag Johansen, Michael A. Riegler, P al Halvorsen, Ulas Bagci 2016, 2021

[8] **A Fog-Based Security Framework for Large-Scale Industrial Internet of Things Environments** - Hejia Zhou, Shantanu Pal, Zahra Jadidi, and Alireza Jolfaei,2022)