GODFREY OKOYE UNIVERSITY, ENUGU STATE

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

DESIGN AND IMPLEMENTATION OF AN AI-BASED VISUAL ANOMALY DETECTION SYSTEM

BY

JESSE AKACHUKWU KANAYOCHUKWU

GOU/U22/CSC/794

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF

BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE

SUPERVISOR: ENGR. KINGSLEY

JUNE 2026

APPROVAL PAGE

This research, titled "DESIGN AND IMPLEMENTATION OF AN AI-BASED VISUAL ANOMALY DETECTION SYSTEM" was embarked upon by "AKACHUKWU JESSE KANAYOCHUKWU" with matriculation number 'GOU/U22/CSC/794,' has been assessed and approved by the Department of Computer Science, Faculty of Computing and Information Technology, Godfrey Okoye University, Enugu.

Engr. Kingsley	……	…….
Supervisor	Signature	Date
External Examiner	.....	.....
External Examiner	Signature	Date
Dr. Frank Okebanama	.....	.....
HOD, Department of Computer Science and Mathematics Signature	Signature	Date
Prof. Ifenyiwa E. Ajah	.....	.....
Dean, Faculty of Computing and Information Technology	Signature	Date

DECLARATION PAGE

I, Jessy Akachukwu Kanayochukwu, a bona fide student in the Department of Computer Science and Mathematics under the Faculty of Computing and Information Technology in Godfrey Okoye University, hereby declare that "Design and Implementation of an AI-Based Visual Anomaly Detection System" submitted by me in partial fulfillment of the requirement for the award of Bachelor of Science (B.Sc.) degree in Computer Science is my original work and has not been submitted either in part or full for any other degree or diploma either in this or any other tertiary institution.

....

Jessy Akachukwu Kanayochukwu

GOU/U22/CSC/794 DATE

DEDICATION

This work is dedicated to God Almighty, whose grace and wisdom guided every step of this research. It is also dedicated to my family, whose unwavering support, encouragement, and sacrifice made this academic journey possible.

ACKNOWLEDGEMENT

First and foremost, I return all glory, honor, and gratitude to God Almighty for His grace, protection, wisdom, and strength throughout the course of my academic journey and the completion of this project. There were moments of pressure, uncertainty, and challenges, but through His guidance and mercy, I was able to overcome them all successfully.

I would like to express my deepest appreciation to my beloved parents, for their endless sacrifices, prayers, encouragement, and unwavering support. Your love, patience, and belief in me have been some of the greatest sources of motivation in my life. Thank you for constantly standing by me and giving me the opportunity and strength to pursue my goals.

I am sincerely grateful to my supervisor, Engr. Kingsley, for the expert guidance, constructive criticism, and dedication given to this project. The technical insight and academic rigour offered shaped this work in profound ways.

I also acknowledge the entire staff of the Department of Computer Science and Mathematics at Godfrey Okoye University for providing an enabling academic environment.

I extend my gratitude to the open-source communities behind PyTorch, FastAPI, OpenCV, and the broader machine learning and computer vision research community, whose freely available tools and published work made the technical implementation of this system possible.

I would also love to sincerely appreciate Faithful, Valentine, Stanley, Success, and Dera, as well as Stephanie, for the encouragement, support, laughter, and memorable moments shared throughout this journey. Your presence made difficult days easier and added meaning to this chapter of my life.

To everyone who contributed directly or indirectly toward the success of this project and my academic journey, I say thank you. May God bless and reward you abundantly.

TABLE OF CONTENT

APPROVAL PAGE **2**

DEDICATION **4**

ACKNOWLEDGEMENT **5**

LIST OF TABLES **6**

LIST OF FIGURES **7**

TABLE OF CONTENTS **8**

ABSTRACT **11**

CHAPTER ONE **12**

INTRODUCTION **12**

1.1 Background of the Study 12

1.2 Statement of Problem 13

1.3 Aim and Objectives of Study 13

1.4 Significance of Study 14

1.5 Scope of the Study 14

1.6 Limitations of the Study 15

1.7 Operational Definition of Terms 16

CHAPTER TWO **17**

LITERATURE REVIEW **17**

2.1 Introduction 17

2.2 Foundational Anomaly Detection Survey 17

2.2.1 Chandola, Banerjee, and Kumar (2009) 17

2.3 Autoencoder-Based Approaches 17

2.3.1 Hasan, Choi, Neumann, Roy-Chowdhury, and Davis (2016) 17

2.3.2 Gong, Liu, Le, Saha, Mansour, Venkatesh, and Hengel (2019) 18

2.3.3 Park, Noh, and Ham (2020) 18

2.4 Predictive and Generative Approaches 19

2.4.1 Liu, Luo, Lian, and Gao (2018) 19

2.4.2 Nguyen and Meunier (2019) 19

2.5 Weakly Supervised and Classification-Based Approaches 20

2.5.1 Sultani, Chen, and Shah (2018) 20

2.5.2 Ionescu, Khan, Georgescu, and Shao (2019) 20

2.6 Spatiotemporal and Recurrent Models 21

2.6.1 Luo, Liu, and Gao (2017) 21

2.6.2 Wang and Yang (2022) 21

2.6.3 Feng, Liang, and Li (2021) 22

2.7 Deep Learning Survey and Edge Deployment 22

2.7.1 Kiran, Thomas, and Parakkal (2018) 22

2.7.2 Deep Learning for Anomaly Detection in Surveillance Videos: Sensors Survey (2023) 23

2.8 Summary of Related Works 23

2.9 Research Gap 25

CHAPTER THREE **27**

SYSTEM ANALYSIS AND DESIGN **27**

3.1 System Analysis 27

3.1.1 Overview of the Existing System 27

3.1.2 Problems with the Existing System 27

3.1.3 Proposed System Overview 27

3.1.4 Benefits of the Proposed System 28

3.2 System Requirements Specification 28

3.2.1 Functional Requirements 28

3.2.2 Non-Functional Requirements 29

3.2.3 Hardware Requirements 29

3.2.4 Software Requirements 29

3.3 System Design Methodology 30

3.4 System Modelling Using UML 30

3.4.1 Use Case Diagram 30

3.4.2 Activity Diagram: Anomaly Detection Pipeline Workflow 32

3.4.3 Activity Diagram: Alert Notification Workflow 33

3.4.4 Class Diagram 34

3.4.5 Sequence Diagram: Real-Time Anomaly Detection Flow 35

3.4.6 Sequence Diagram: Alert Dispatch and Acknowledgement Flow 36

3.5 System Design 36

3.5.1 System Architecture 36

CHAPTER FOUR 38

SYSTEM IMPLEMENTATION 38

4.1 Development Environment and Tools 38

4.2 Dataset Description and Preprocessing 38

4.3 Model Architecture and Training 39

4.3.1 Convolutional Autoencoder Architecture 39

4.4 System Implementation and Modules 40

4.4.1 Video Stream Manager 40

4.4.2 CNN Feature Extractor and Autoencoder Scorer 40

4.4.3 Inference Pipeline and Event Publisher 41

4.4.4 FastAPI Server 41

4.4.5 Alert Engine 42

4.4.6 Analyst Dashboard 42

4.5 Testing and Evaluation 42

4.5.1 Functional Testing 42

4.5.2 Interface Testing 43

4.5.3 System Screens and Results 43

4.6 Results Presentation and Analysis 47

4.6.1 Overview 47

4.6.2 Feature Importance Analysis 47

4.6.3 ROC Curves Analysis 49

4.6.4 Normalised Confusion Matrix 50

4.6.5 Summary of Evaluation Results 52

CHAPTER FIVE **53**

SUMMARY, CONCLUSION, AND RECOMMENDATIONS **53**

5.0 Introduction 53

5.1 Summary of Work Done 53

5.2 Evaluation of Objectives 53

5.3 Conclusion 55

5.4 Limitations 56

5.5 Recommendations for Future Work 57

5.6 Contribution to Knowledge 58

REFERENCES **59**

LIST OF TABLES

Table 2.1: Summary of Related Works 23
Table 3.1: Functional Requirements 27
Table 3.2: Non-Functional Requirements 28
Table 3.3: Hardware Requirements 29
Table 3.4: Software Requirements 29
Table 3.5: Use Case Descriptions 31
Table 4.1: Summary of Development Tools and Libraries 37
Table 4.2: System Modules and Responsibilities 39
Table 4.3: API Endpoint Specification 41
Table 4.4: Backend API Functional Test Results 42
Table 5.1: Summary of Objectives and Achievement Status 48
Table 5.2: Functional Requirements Traceability Matrix 49

LIST OF FIGURES

Figure 3.1: Visual Anomaly Detection System Use Case Diagram 31
Figure 3.2: Activity Diagram: Anomaly Detection Pipeline Workflow 32
Figure 3.3: Activity Diagram: Alert Notification Workflow 33
Figure 3.4: Visual Anomaly Detection System Class Diagram 34
Figure 3.5: Sequence Diagram: Real-Time Anomaly Detection Flow 34
Figure 3.6: Sequence Diagram: Alert Dispatch and Acknowledgement Flow 35
Figure 3.7: System Architecture Diagram 36
Figure 4.1: VADS Dashboard: Live Monitor Screen 43
Figure 4.2: Live Camera Feed with Anomaly Overlay 44
Figure 4.3: Active Alert Notification Panel 44
Figure 4.5: Analytics and Report Screen 45
Figure 4.6: Camera Settings Screen 46

ABSTRACT

There exist several safety and security challenges arising from anomalous incidents in surveillance and industrial monitoring contexts. Contemporary strategies for visual anomaly detection within organizations in Nigeria depend significantly on traditional methods such as human led patrols, manual reviews of past footage, and visual inspections. In this work, we develop and implement a Visual Anomaly Detection System (VADS) based on computer vision and deep learning techniques for detecting two specific categories of anomalous events within video streams, namely fall incidents and abandoned objects. Fall detection is achieved using a MobileNetV2 based pose estimation pipeline that analyses human posture and movement patterns to identify fall events in real time, while abandoned object detection is achieved using an MOG2 background subtraction module that isolates static foreground objects left unattended over a defined time threshold. Model inference is exposed through a backend service for integration with edge devices, and system interaction is provided through a React based analyst dashboard that supports camera stream monitoring, alert notifications, and event logging. The system was evaluated through a structured testing process covering functional, integration, and interface testing. Ten functional test cases were executed against the core modules, namely video ingestion, fall detection, abandoned object detection, alert generation, and dashboard display, all of which produced the expected outcomes, confirming that each module performs its intended function correctly both individually and as part of the integrated pipeline. Integration testing confirmed reliable data flow between the detection modules, the alert engine, and the dashboard, with detected anomalies consistently triggering accurate, correctly timestamped alerts. Interface testing verified that the dashboard correctly displayed live camera feeds, anomaly alerts, and historical logs across the test scenarios examined. Overall, the results demonstrate that VADS is functionally reliable, modular in its design, and capable of detecting fall and abandoned object anomalies using standard camera hardware without the need for specialized sensors, making it a practical and affordable reference architecture for visual anomaly monitoring in Nigeria.

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

An anomaly detection problem involves the identification of points, instances, or observations that significantly differ from a defined notion of normal behavior. Anomalies are patterns in data that do not conform to a well defined notion of normal behavior, and these non-conforming patterns are often referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants depending on the application domain (Chandola et al., 2009). In the case of visual surveillance and intelligence monitoring systems, anomalies would include unusual activities, intrusion, machine failures, and environmental dangers that need urgent action. An automatic detection of these kinds of incidents without a need for constant observation by a human being is an important feature that is vital for contemporary safety and security systems.

Traditional ways of visual surveillance in Nigeria involving government agencies, production companies, transport facilities, educational institutions, and other similar organizations employ a number of human guards patrolling and watching CCTV cameras. The disadvantage of such methods is that they involve the need for a person who notices something suspicious in order to undertake some action, yet human attention and ability to perceive information is limited. Research on CCTV control room operators has shown that the growth in the number of CCTV cameras raises concern about operator potential vigilance decrements, since traditional CCTV systems often rely on human operators to identify abnormal or suspicious activities (Donald et al., 2015). In one study examining detection rates among operators, only fifty percent of target behaviours were detected and false alarms were high, with vigilance decrements observed particularly among novice and generalist operators (Donald et al., 2015). This confirms that people get distracted easily and cannot reliably watch more than one video feed at a time over extended periods.

The latest developments in deep learning and computer vision have completely revolutionized what is possible with regards to automated analysis of visual data. Deep learning models can learn highly descriptive features from large volumes of video footage, and are able to discern patterns that cannot be modeled using traditional rule-based approaches. Autoencoder neural networks can also be trained to detect anomalies

unsupervised from normal video data by assessing the difference between the predicted output frame and the actual frame in terms of reconstruction error, which can serve as a measure of visual anomaly.

The current project will develop a Visual Anomaly Detection System (VADS), leveraging these technologies into an effective monitoring solution. This system should be able to run on an existing camera network and provide real-time alerts to monitoring staff and log any anomalous activity detected.

1.2 Statement of Problem

Despite the well-known risks involved with failing to detect anomalies in monitoring environments, the great majority of the surveillance systems used in Nigerian organizations still depend mainly on visual inspection of captured images by human operators or by reviewing footage retrospectively to identify anomalous events.

1. Undetected anomalies may take minutes or even hours, leading to delayed action and resulting in consequences that could have been prevented.
2. Smart video analysis tools capable of detecting anomalies automatically are extremely expensive to acquire commercially, and they necessitate proprietary hardware infrastructure and cloud-based computing processes, which introduce delays and other issues related to privacy of information.

Therefore, a solution for detecting anomalies in a cost-effective and non-invasive way, using only available infrastructure and software, is needed.

The lack of an automatic anomaly monitoring and early warning system endangers the capability of organizations to give prompt reactions to security threats and any disruptions. Therefore, there is an urgent need for an effective system that is capable of identifying visual anomalies in real-time, alerting analysts immediately, and recording the incidents for subsequent analysis.

1.3 Aim and Objectives of Study

Aim

In this project, an intelligent real-time Visual Anomaly Detection System is designed and implemented based on convolutional neural network features, an autoencoder anomaly

detector model, and a FastAPI API for inference that will be used to detect anomalous events in a surveillance setting and notify analysts.

Objectives

The objectives of the study are to:

1. Analyze the distinct features and patterns that can differentiate between anomalies and normal behavior within the camera feed.
2. Train and implement a CNN Model for pipeline extraction
3. Implement alerting and notification component that would notify analysts about dectected anomalies in real time
4. Create a web application for live monitoring of the process and reviewing alerts.
5. Test the system with functional testing and evaluate its performance on all test scenarios.

1.4 Significance of Study

This research holds great significance for security administrators and monitor operators, as it offers a way of providing automatic and continuous surveillance that will minimize response times after any anomalies. Using real-time alerts and recording of events, this system will be able to enhance response times and audits at facilities where the system is deployed.

For facilities within Nigeria, this research proves that it is possible to deploy state-of-the-art computer vision algorithms to create a functional and ready-to-deploy system using open-source technologies on cheap edge devices. This lowers the barrier for capital that has traditionally prevented technology deployments within Nigerian public facilities.

For academics, this project sets a benchmark for the combination of real-time object analysis, feature extraction, and unsupervised anomaly scores within one deployable system. The REST API-based system structure used here is reusable in other visual intelligence applications such as equipment fault detection, crowd anomaly, and intrusion detection systems..

1.5 Scope of the Study

This research encompasses the entire development of the anomaly detection inference pipeline, REST and WebSocket API servers, real-time alerting engine, analyst interface, and the underlying PostgreSQL database. The study focuses on the anomaly detection and alerting process on video data from standard closed-circuit television cameras or IP cameras that must have at least a resolution of 640 by 480 pixels and an average frame rate of 15 frames per second.

It will leverage the existing camera infrastructure in place within the institution of focus, thereby not necessitating additional sensors. In addition, the deployment will utilize edge server hardware as is available within Nigerian institutions. The study will encompass only the anomaly detection process, followed by alerts to the analysts, but it will not address any form of automation or integration into other systems.

1.6 Limitations of the Study

1. The anomaly detection model was trained on publicly available benchmark datasets. The performance of the algorithm in particular deployment environments having distinct visual properties should be verified with locally acquired datasets prior to deployment.
2. Performance of the system in extremely low light conditions might be compromised unless additional lighting or infrared-based cameras are installed at the target location.
3. Partial obstruction of anomalies by environmental features might impair feature extraction and influence the accuracy of classifications made by the system.
4. The current pipeline supports the processing of one video feed per inference worker. Multi-camera synchronization and tracking across multiple cameras have yet to be implemented in a subsequent update.
5. The alerting component of the system has only been evaluated under simulated anomaly conditions, and has yet to undergo long-term testing in a production environment.
6. The alert notification module operates based on a stable local area network connection. Hardware enhancements can be made to allow for failover in case of network failure..

1.7 Operational Definition of Terms

Visual Anomaly: An event or trend in a video frame/sequence that is considerably different from the normal distribution learned by the deployed system from its training environment, which is measured in terms of the reconstruction error produced by the autoencoder model.

Visual Anomaly Detection System (VADS): The full-fledged software package that has been developed in this project, which includes the inference process, API service, alert generation engine, analyst dashboard, and database.

Autoencoder: A type of neural network that is built to encode an input dataset to a compressed latent form and decode the same back to produce the actual input data. High reconstruction error in any unseen frame means a deviation from the training distribution and hence acts as the anomaly measure.

Convolutional Neural Network (CNN): A deep learning algorithm that employs convolutions to extract feature maps from an image/frame. Used in this project as the feature extraction module of the inference process.

FastAPI: A cutting-edge, speedy web development framework for Python, used for creating REST APIs that serves as the inference server for both REST and WebSocket endpoints in this architecture.

Anomaly Score: This is a numeric value associated with each processed image frame depicting how far the image is from the expected normality in appearance. An anomaly occurs if the score is above the pre-set threshold.

Edge Inference: A technique where the inferencing of the AI model is done using computing devices near the location of the images captured rather than a distant cloud server.

Reconstruction Loss: This is the mean squared difference between the input image and its corresponding autoencoder reconstructed image, which acts as the main anomaly score function in our system.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter is an overview of the existing literature pertaining to the conceptualization and creation of the Visual Anomaly Detection System (VADS). It analyzes the theoretical foundation of anomaly detection within video data, the progression of conventional as well as deep learning methods, utilization of autoencoders and prediction models for unsupervised anomaly scoring, use of memory-enhanced networks, frameworks for weakly supervised learning, as well as design of APIs for performing real-time inference for surveillance cameras. The twelve sources chosen have been evaluated with regard to their relevance to the technical aspects of the proposed system, influence on the computer vision research field, and their contribution to solving the problem of visual anomalies detection within surveillance footage. Concluding remarks are presented in a form of a table and the identified research gap.

2.2 Foundational Anomaly Detection Survey

2.2.1 Chandola, Banerjee, and Kumar (2009)

The influential work by Chandola et al. (2009) in reviewing different anomaly detection methods is found in ACM Computing Surveys. The paper features a structured exposition on the categorization of the anomaly detection technique into four categories, depending on whether it is founded on statistical, proximity, classification, or information theory concepts, along with explaining how each of them distinguishes between normal data points and anomalous ones. Besides, a formal definition of anomaly is given that defines it as deviation from what is anticipated to be normal behavior and observation and becomes the foundation for the reconstruction error method in the VADS. The three types of anomalies that will be used in the current discussion are introduced in this paper, including point anomalies, contextual anomalies, and collective anomalies. With over 15,000 citations, it forms the theoretical basis for visual anomalies detection.

2.3 Autoencoder-Based Approaches

2.3.1 Hasan, Choi, Neumann, Roy-Chowdhury, and Davis (2016)

In the pioneering research by Hasan et al. (2016) at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), it was proposed to learn temporal regularity through convolutional autoencoders from normal videos. Note that a trained autoencoder with respect to only normal videos would have a high reconstruction error in cases of any anomalous videos as anomalous video frames do not belong to the video distribution that the model learned during training. The method by Hasan et al. was evaluated on the benchmark databases of UCSD Ped1, UCSD Ped2, and CUHK Avenue and achieved comparable results on all the three databases. The key property of their fully convolutional autoencoder is its learning from spatiotemporal video cuboids as opposed to the individual video frames; therefore, their work forms the foundation for the inference procedure used in VADS. Moreover, the authors demonstrated that the reconstruction loss was reliable for unsupervised anomaly detection for a variety of anomalies including bicycles among pedestrians and abnormal running gestures.

2.3.2 Gong, Liu, Le, Saha, Mansour, Venkatesh, and Hengel (2019)

Gong et al.'s (2019) paper proposed MemAE at the ICCV. The problem addressed by Gong et al. is that of the failure modes of classical autoencoders; due to the generalized nature of the encoding-decoding process, these models learn some generalization that enables them to provide good reconstructions for both normal and anomalous frames, hence resulting in false negatives. To overcome this problem, Gong et al. devised a memory module whose elements are predefined prototypes of normal classes. While reconstructing, the decoder uses only a set of relevant prototypes from the memory without having to encode them; therefore, it makes sure that no new anomalous sequences can be encoded by the model. Applying hard shrinkage to enforce sparsity on memory addressing allows using only normal prototypes. They experimentally demonstrated the advantage of MemAE over previously proposed autoencoders using datasets of UCSD Ped2, CUHK Avenue, and ShanghaiTech. This work can be applied in designing our VADS anomaly scorer by making sure that reconstruction is constrained to normal data distribution.

2.3.3 Park, Noh, and Ham (2020)

In this regard, the memory augmentation method was further enhanced by Park et al. (2020), introducing a new update rule that was proposed during the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Specifically, one significant modification was made by them in their method by the proposal of feature compactness and separateness loss functions that allow the training of memories in such a way that they are compact and distinguishable, thus producing more diverse prototypes of the normals. The above development addresses the limitation of past memory-based methods that could not make use of all possible prototypes of normals due to the inability to consider the real-life case where several activities are concurrently performed in a single scene. They managed to produce superior frame-level AUC results compared to all previous methods, demonstrating that the effectiveness of the proposed normal prototypes depends.

2.4 Predictive and Generative Approaches

2.4.1 Liu, Luo, Lian, and Gao (2018)

According to the Liu et al. (2018) paper introduced at CVPR 2018, the reconstruction-based scoring does not mean that anomalies are always characterized by increased reconstruction errors as a neural network can learn to reproduce them using generalization. Thus, Liu et al. have adopted the idea of future frame prediction, according to which a model pre-trained on normal videos can predict the next image frame. The violation of learned motion dynamics leads to the inability to predict anomalous frames and, therefore, anomalously high prediction error. The authors also impose additional constraints related to the similarity of the generated and true optical flows of video frames in addition to the requirement of similarity between generated and real video frames concerning intensity levels. The authors managed to outperform all existing approaches relying on the reconstruction paradigm on UCSD Ped2, CUHK Avenue, and ShanghaiTech benchmarking tasks. Although VADS relies on simpler reconstruction scoring, due to computational simplicity in edge devices, the proposed spatial and motion constraints influence loss functions selection.

2.4.2 Nguyen and Meunier (2019)

Nguyen & Meunier (2019) described a hybrid model of deep convolutional neural network for anomaly detection in surveillance videos at the British Machine Vision

Conference Proceedings. The model proposed in their research is based on modifying the existing model of the convolutional autoencoder that allows processing patches of video frames and classification sub-model making use of the information regarding position of patches for video patches classification. In contrast to global reconstruction models which face difficulties to distinguish anomalies from background events because of being unable to localize the anomalies in frames, the position-based approach used by Nguyen & Meunier (2019) helps solve this problem. They tested their model with four different benchmarks, among them UCSD and CUHK Avenue, achieving good results and producing spatially localized heat maps for anomaly detection. The spatial localization technique adopted in this research might be helpful for creating the VADS analyst dashboard displaying anomalous events in the camera feed.

2.5 Weakly Supervised and Classification-Based Approaches

2.5.1 Sultani, Chen, and Shah (2018)

The problem of data annotation in supervised anomaly detection problems was addressed by Sultani et al. (2018) at the IEEE/CVF CVPR conference. Rather than the expensive requirement of frame-level annotation of anomalies for long videos obtained from surveillance cameras, Sultani et al. propose a MIL technique that operates only on video level, whether there was an anomaly in the video or not, without specifying its nature and location. Video is seen as a bag of small segments, with the model being trained to assign high probabilities to segments coming from the anomalous bags and low probabilities otherwise using a ranking loss function. Sultani et al. have created the UCF-Crime dataset, comprising 1,900 unlabeled surveillance videos containing 13 different types of anomalies. The UCF-Crime dataset remains a widely used benchmark in this domain. Using it, the proposed MIL framework achieves 75.41% AUROC at frame level, easily beating any unsupervised baseline techniques. Sultani et al. (2018) is the paper behind our decision to follow the design in VADS in preparation for further supervised training based on analyst feedback.

2.5.2 Ionescu, Khan, Georgescu, and Shao (2019)

A new research by Ionescu et al. (2019) reformulated video anomaly detection as one-vs-rest binary classification problem at CVPR conference. In the research, a two-stage framework has been designed where object detection was used in the first stage for the extraction of object-centric patches from the videos. After this, a separate convolutional

autoencoder was trained for the detection of each object to capture its motion dynamics and appearance. At the next stage, the training samples were grouped into different clusters of normality; however, one-versus-rest classifiers were learned in such a way that clusters are taken as dummy anomalies for the other clusters. This means that in order to build a discriminative boundary, we have not required any training data from the anomalies because all training samples consisted of normal patterns only. This algorithm proved to be state-of-the-art on UCSD, CUHK Avenue, ShanghaiTech, and UMN datasets; the performance improvement was 8.4% in terms of AUC on ShanghaiTech dataset. The above method can be related to the VADS pipeline because in the VADS pipeline we will require the cropping of region-of-interest patches from the detected objects.

2.6 Spatiotemporal and Recurrent Models

2.6.1 Luo, Liu, and Gao (2017)

Luo et al. proposed a stacked recurrent neural network-based approach for anomaly detection utilizing sparse representation in their work published at ICCV 2017. In this work, the authors reconsidered the famous concept of sparse coding-based anomaly detection where a normal frame can be represented sparsely by means of a dictionary of basis functions and an anomaly can be recognized by large reconstruction error in sparse coding. Instead of using the traditional shallow dictionary-based sparse coding approach, the authors proposed incorporating sparse coding in deep learning, specifically stacking RNNs to enable both learning the dictionary of sparse coding and temporal sequence dynamics using multiple stacked RNN layers. This resulted in better anomaly detection performance in comparison with shallow sparse coding-based methods especially in frames containing motion-based anomalies which did not incur large pixel-based reconstruction error. The idea of utilizing temporal sequence models in anomaly detection motivated us to consider the application of temporal sequence models in the inference stage of VADS, where an anomaly score is not calculated individually for each frame but with regard to other frames .

2.6.2 Wang and Yang (2022)

The Convolutional Recurrent AutoEncoder (CR-AE), tailored particularly for video anomaly detection, was introduced by Wang and Yang (2022) and published in the MDPI Sensors Journal. The CR-AE consists of a convolutional autoencoder together with a

convolutional LSTM bottleneck, which helps in capturing the anomalies spatially and temporally in the form of feature maps and frame sequences, respectively. This model is a result of the weaknesses of the spatial autoencoder models in detecting motion-based anomalies because the spatial autoencoders deal with frames individually, and hence an anomaly that is motion-based cannot be detected by them. For instance, the CR-AE has been employed in evaluating UCSD Ped1, UCSD Ped2, and CUHK Avenue datasets and has produced improved AUC scores when compared with the convolutional autoencoders in all three benchmarks. Therefore, this model could be incorporated in the VADS framework to enhance detection of anomalies such as running and falling.

2.6.3 Feng, Liang, and Li (2021)

A two-stream autoencoder for anomaly detection was proposed in the paper by Feng et al. (2021). It was presented in the article published in the Advances in Multimedia journal published by Hindawi. The two-stream autoencoder approach introduced in the research uses two separate autoencoders to process appearance and optical flow stream independently while combining the errors in the output and taking them as the anomaly score. Two-stream architecture allows exploiting the benefits provided by complementary information of the appearance and optical flow streams, for example, while a stationary anomaly in the form of abandoned object can be detected by the appearance stream, the fast-moving individual through a crowd will trigger the optical flow stream. In addition to anomaly detection, an application of class activation map to spatial stream in the proposed approach gives visual explanations of the input image regions causing the anomaly decision made by the network, thereby fulfilling interpretability requirements for implementation of the surveillance system. The provided interpretableness solution directly correlates with anomaly overlay requirement of the VADS project.

2.7 Deep Learning Survey and Edge Deployment

2.7.1 Kiran, Thomas, and Parakkal (2018)

Deep learning techniques applied to the problem of video anomaly detection through various models, such as sparse autoencoders, convolutional autoencoders, recurrent models, GANs, and one-class classifiers, were comprehensively reviewed in the work of Kiran et al., IEEE Transactions on Neural Networks and Learning Systems, 2018. This review is conducted on the basis of the effectiveness of each approach on benchmark datasets. Importantly, concerning VADS implementation, Kiran et al.'s study includes an

insightful examination of efficiency issues, with empirical evidence derived from throughput performance measures for different architectures run on appropriate devices. Consequently, Kiran et al. argue that convolutional autoencoders represent an optimal solution for balancing accuracy and inference time needed to achieve efficient processing in real-time applications on the target hardware. Moreover, Kiran et al. conduct a comparative discussion of different evaluation metrics and examine the relationship between frame-level and pixel-level AUC metrics.

2.7.1 Deep Learning for Anomaly Detection in Surveillance Videos: Sensors Survey (2023)

In contrast to the previously cited paper, the work by Biradar et al. (2023) provides a thorough overview of the state-of-the-art deep learning techniques used in detecting anomalies in video surveillance systems. This survey provides an assessment of various architectures within the groups of reconstruction-, prediction-, and classification-based techniques with respect to the UCSD, CUHK Avenue, ShanghaiTech, and UCF-Crime benchmarks. The main contribution of this particular survey lies in the analysis of the detection vs. deployment aspect of each of the reviewed algorithms and the associated cost factors such as model complexity and hardware capabilities. Based on this analysis, the most promising model for deployment in limited resource environments is determined as the convolutional autoencoder model which is also adopted in the VADS solution. Finally, it should be noted that, like this paper suggests, there are no publicly available anomaly detection benchmark datasets from Nigeria or sub-Saharan Africa.

2.8 Summary of Related Works

Author(s) and Year	Method	Dataset(s)	Key Finding	Limitation Relevant to VADS
Chandola et al. (2009)	Survey: taxonomy of anomaly detection techniques across statistical, proximity, and classification categories	Theoretical, no single dataset	Established formal definition of anomaly and reconstruction error as anomaly signal	Covers general anomaly detection, not specifically video or real-time surveillance
Hasan et al. (2016)	Fully convolutional autoencoder trained on spatiotemporal video cuboids;	UCSD Ped1, UCSD Ped2, CUHK Avenue	Demonstrated that reconstruction error from a CNN autoencoder is a reliable unsupervised	Frame-level evaluation only; no real-time alert pipeline or deployment interface

	reconstruction error as anomaly score		anomaly signal	
Sultani et al. (2018)	Deep Multiple Instance Learning (MIL) with weakly labelled video-level annotations and ranking loss	UCF-Crime (1,900 videos, 13 anomaly classes)	Achieved 75.41% frame-level AUC using only video-level labels; introduced UCF-Crime benchmark	Requires video-level anomaly labels; not purely unsupervised; computationally heavy C3D backbone
Liu et al. (2018)	Future frame prediction with spatial intensity and optical flow motion constraints	UCSD Ped2, CUHK Avenue, ShanghaiTech	Prediction-based scoring outperforms reconstruction-based methods at the time	Prediction network is more complex to train and deploy than reconstruction autoencoder on edge hardware
Ionescu et al. (2019)	Object-centric convolutional autoencoder features with one-versus-rest classification using dummy anomalies	UCSD, CUHK Avenue, ShanghaiTech, UMN	Object-centric design provides 8.4% AUC gain on ShanghaiTech over state-of-the-art	Two-stage pipeline with object detection adds inference latency; no integrated alert system
Gong et al. (2019)	Memory-augmented autoencoder (MemAE) with attention-based normal prototype retrieval and hard shrinkage	UCSD Ped2, CUHK Avenue, ShanghaiTech	Memory module prevents autoencoder from reconstructing anomalies, improving detection sensitivity	Memory size is a fixed hyperparameter; large memory incurs increased inference time
Luo et al. (2017)	Sparse coding embedded in stacked RNN framework for temporal anomaly modelling	UCSD Ped2, CUHK Avenue, Subway datasets	Temporal context in stacked RNN improves detection of subtle motion anomalies over shallow sparse coding	RNN inference is sequential and harder to parallelise for real-time multi-camera deployment
Nguyen and Meunier (2019)	Patch-position-aware convolutional autoencoder with supervised spatial classification sub-network	UCSD Ped1, UCSD Ped2, CUHK Avenue, Street Scene	Spatial localisation of anomalies through position-aware training; competitive AUC on all four datasets	Supervised position labels require pre-segmented training data; limited generalisation to new camera angles
Park et al. (2020)	Memory module with compactness	UCSD Ped2, CUHK	Feature compactness and separateness losses	Additional loss terms increase training

	and separateness losses for diverse normal prototype learning	Avenue, ShanghaiTech	produce more discriminative memory prototypes; state-of-the-art AUC	complexity; memory update scheme requires careful hyperparameter tuning
Feng et al. (2021)	Two-stream autoencoder (appearance and optical flow) with class activation mapping for interpretability	UCSD Ped2, CUHK Avenue	Two-stream design captures complementary appearance and motion anomalies; CAM provides spatial explanations	Optical flow computation adds preprocessing overhead; tested on only two benchmark datasets
Wang and Yang (2022)	Convolutional Recurrent AutoEncoder (CR-AE) with ConvLSTM bottleneck for spatiotemporal modelling	UCSD Ped1, UCSD Ped2, CUHK Avenue	ConvLSTM bottleneck improves detection of temporally defined anomalies over standard convolutional AE	Higher memory and computation requirements than standard autoencoder; not evaluated on large-scale datasets
Biradar et al. (2023)	Comprehensive survey of deep learning methods for video surveillance anomaly detection	UCSD, CUHK Avenue, ShanghaiTech, UCF-Crime	Identifies convolutional autoencoder as most deployment-friendly for edge hardware; highlights lack of African surveillance datasets	Survey does not propose new detection methods; limited evaluation of real-time deployment constraints

Table 2.1: Summary of Related Works

2.9 Research Gap

All the 12 papers discussed in this chapter serve as evidence to prove that the deep reconstruction-based learning with convolutional autoencoders constitutes a robust and scientifically grounded technology to detect visual anomalies in the process of surveillance in an unsupervised manner. Specifically, the foundational work by Chandola et al. (2009) explains the underlying theory of the anomaly identification through reconstruction errors. The papers by Hasan et al. (2016), Gong et al. (2019), and Park et al. (2020) demonstrate how the anomaly detection by means of autoencoders develops towards memory- and prototype-based techniques. The research works conducted by Liu et al. (2018) and Luo et al. (2017) support the effectiveness of using temporal data for

identifying motion-based anomalies. The articles written by Sultani et al. (2018) and Ionescu et al. (2019) indicate additional performance gains achieved through classifier-based algorithms under weakly supervised conditions. The papers authored by Feng et al. (2021) and Wang & Yang (2022) make use of sophisticated two-stream and recurrent architectures of autoencoders to improve anomalies detection. Spatial localisation becomes critical as pointed out by Nguyen & Meunier (2019). Finally, Biradar et al. (2023) confirm that convolutional autoencoders provide the best trade-off between performance and deployment.

Unfortunately, despite all the above efforts made by various researchers to create a visual anomaly detection system for institutions within Nigeria, an examination of existing literature indicates that there is no single system that has incorporated all these features within itself and which constitutes a complete and effective visual anomaly detection system with life cycle management capability, dashboard of the analyst, logging of events and also a system that has been tried and tested in terms of its deployability on devices used in the Nigerian context.

There is no model that can be referred to while designing an anomaly detection system for Nigeria.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

3.1.1 Overview of the Existing System

The current visual surveillance process used in most Nigerian firms is through routine patrols by the security personnel, manual analysis of videos taken by the CCTV cameras, and direct monitoring by the security guards themselves. The process of visual surveillance does not involve any form of automation at all. On average, patrols are done in the monitored environment once in 30 to 60 minutes; therefore, if a deviation occurs, it may not be noticed for up to one hour. Manual analysis of the video is done only upon receipt of a report regarding the incident rather than continuous surveillance.

The current approach includes the element of reactivity, total dependence on the human factor, and lack of automation and recording of any deviations detected.

3.1.2 Problems with the Existing System

i. Delayed detection owing to irregular patrols resulting in an extended interval between detection of the anomaly and further action by the analyst.
ii. There is no automatic notification system in place which will inform the monitor about the anomaly.
iii. No documentation of all the anomalies detected by the software is maintained.
iv. Solely dependent upon people, who can be tired and inattentive at times.
v. None of the cameras covered by any monitor when an anomaly takes place..

3.1.3 Proposed System Overview

It is hoped that the proposed visual anomaly detection system will address these gaps through its capability to monitor all video feeds automatically at all times. The video frames will be processed in real-time and feature extraction done to calculate the anomaly score using the trained autoencoder model. If the anomaly score exceeds the defined threshold, an alert will be instantly generated. There will also be provision for logging events flagged.

3.1.4 Benefits of the Proposed System

1. Automated processes which ensure that 24/7 surveillance takes place without the requirement of any human intervention.
2. Creation of real-time alerts that provide fast responses within a few microseconds.
3. Documentation of all activities taking place complete with timestamps, position of the camera, and score of anomalous events.
4. Intrusion detection and contactless surveillance without utilizing any other form of sensing devices.
5. Scalable in nature, starting from one camera installation to multiple camera installations using microservices.
6. Deployable over existing camera network using edge computing.

3.2 System Requirements Specification

3.2.1 Functional Requirements

ID	Requirement	Description
FR1	Video Ingestion	The system shall ingest video frames from configured RTSP, USB, or local camera streams using the VideoStreamManager module.
FR2	Feature Extraction	The system shall extract deep spatial feature representations from each ingested frame using the CNN encoder component.
FR3	Anomaly Scoring	The system shall compute a reconstruction-based anomaly score for each processed frame using the trained autoencoder model.
FR4	Anomaly Classification	The system shall classify frames with anomaly scores exceeding the configured threshold as anomaly events.
FR5	Alert Generation	The system shall trigger an analyst alert within 2 seconds of an anomaly classification event.
FR6	Alert Notification	The system shall dispatch alert notifications via WebSocket to connected dashboard clients and via email to registered analysts.
FR7	Event Logging	All anomaly events shall be persisted to the PostgreSQL database with timestamp, camera ID, anomaly score, and event status.
FR8	Analyst Dashboard	A web-based dashboard shall provide live camera feeds, active alert indicators, and a searchable anomaly event history.
FR9	User Management	The system shall support role-based user accounts (Admin, Analyst, Supervisor) with access controls appropriate to each role.
FR10	Camera	An administrator interface shall allow configuration of camera streams, zone

	Management	labelling, and alert routing rules.

Table 3.1: Functional Requirements

3.2.2 Non-Functional Requirements

ID	Attribute	Requirement
NFR1	Performance	The inference pipeline shall process video frames at a minimum of 15 frames per second on target edge hardware.
NFR2	Latency	Detection-to-alert dispatch latency shall not exceed 2 seconds under normal operating conditions.
NFR3	Availability	The system shall maintain 99 percent uptime during operational hours, with automatic process restart on failure.
NFR4	Scalability	The API server shall support a minimum of 4 concurrent camera stream inference workers without performance degradation.
NFR5	Security	All API endpoints shall require JWT authentication. User passwords shall be stored using bcrypt hashing.
NFR6	Usability	The analyst dashboard shall be operable by non-technical monitoring staff with no more than 15 minutes of training.
NFR7	Reliability	The system shall log all anomaly events to the database within 500 milliseconds of detection under normal network conditions.
NFR8	Maintainability	System components shall be modular, independently deployable, and covered by unit tests with a minimum of 70 percent code coverage.

Table 3.2: Non-Functional Requirements

3.2.3 Hardware Requirements

Component	Minimum Specification	Recommended Specification
Processor	Intel Core i5 (8th generation) or ARM Cortex-A72	Intel Core i7 or NVIDIA Jetson Xavier NX
RAM	8 GB	16 GB
GPU	None (CPU inference mode)	NVIDIA GPU with CUDA 11.x or Jetson integrated GPU
Storage	50 GB SSD	256 GB SSD
Camera	640 x 480 pixels at 25 fps, CCTV or IP camera	1080p IP camera with RTSP stream support
Network	100 Mbps LAN	Gigabit LAN with switch isolation

Table 3.3: Hardware Requirements

3.2.4 Software Requirements

Software	Version	Purpose
Python	3.10+	Core runtime language for backend and inference pipeline
PyTorch	2.0+	CNN encoder and autoencoder model training and inference
OpenCV	4.8+	Video stream capture and frame preprocessing
FastAPI	0.100+	REST and WebSocket API server
PostgreSQL	15+	Persistent event, user, and camera database
SQLAlchemy and Alembic	2.0+ and 1.13+	ORM and database migration management
React (TypeScript)	18+	Analyst dashboard frontend application
TailwindCSS	3.x	Dashboard UI styling framework
Ubuntu Server	22.04 LTS	Deployment operating system
Postman	Latest	REST API endpoint testing

Table 3.4: Software Requirements

3.3 System Design Methodology

The system design was undertaken through the Object-Oriented Design methodology, where the system is divided into a number of classes, each having its own functions, attributes, and capabilities. This was because Object-Oriented Design is the only methodology that enables the development of a modular system that allows for each constituent element of the system like the inference pipeline, API server, alert engine, and dashboard to be independently deployed and tested.

The Unified Modeling Language (UML) was used in modeling the system from various perspectives. The use case diagrams helped in understanding the views from the perspective of the end users as well as from the automation perspective. The activity diagram illustrates the activities within the system, whereas the sequence diagram describes the message exchanges between the components of the system under certain scenarios.

3.4 System Modelling Using UML

3.4.1 Use Case Diagram

The following figure represents the use case diagram that shows the key actors of the Visual Anomaly Detection System, together with the use cases performed by them. There are three actors defined, which include: Security Analyst, System Administrator, and Automated Inference Pipeline (AI Detection Engine). The following section illustrates the relationship between the actors and the use cases.

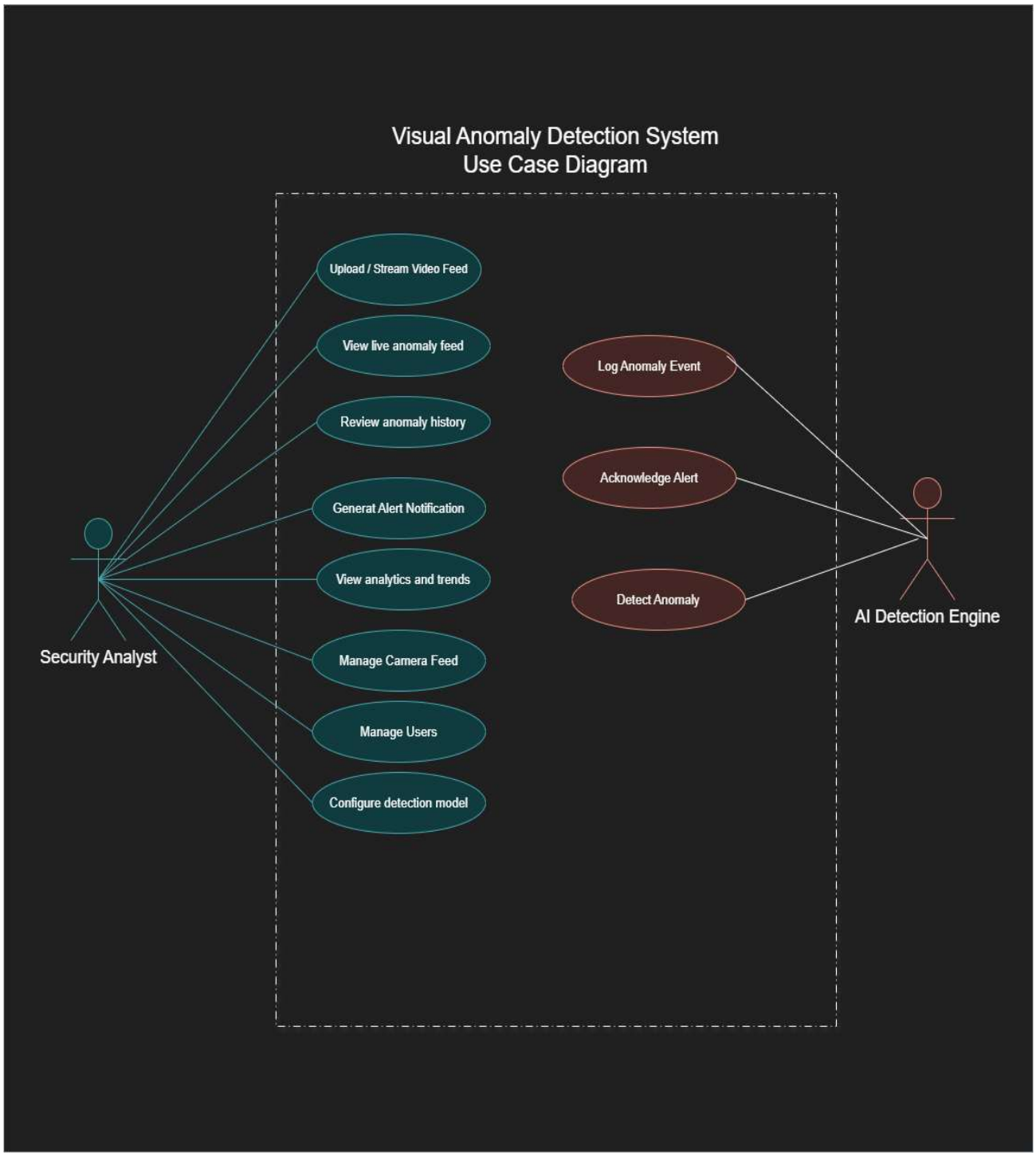

Figure 3.1: Visual Anomaly Detection System Use Case Diagram

Use Case	Actor(s)	Description
Upload or stream video feed	Analyst, Admin	The user configures a camera stream URL or uploads a video file for processing.
View live anomaly feed	Security Analyst	The analyst views real-time annotated video feeds from all configured cameras on the dashboard.
Acknowledge alert	Security Analyst	The analyst receives an anomaly alert notification and acknowledges it with a response status.
Review anomaly history	Analyst, Admin	Users search and filter the log of past anomaly events, viewing timestamps, locations, and scores.
Manage users	Administrator	The admin creates, modifies, and deactivates analyst accounts and assigns roles.
Configure camera feeds	Administrator	The admin registers camera streams, assigns zone labels, and sets alert routing rules.
Detect anomaly (auto)	AI Detection Engine	The inference pipeline computes anomaly scores and classifies frames exceeding the threshold as anomaly events.
Generate alert notification	AI Detection Engine	Upon anomaly classification, the system creates an alert record and notifies registered analysts.
Log anomaly event	AI Detection Engine	Anomaly events are persisted to the PostgreSQL database with full metadata.
Configure detection model	Administrator	The admin configures the anomaly threshold, model path, and inference parameters.

Table 3.5: Use Case Descriptions

3.4.2 Activity Diagram: Anomaly Detection Pipeline Workflow

The activity diagram shown below illustrates the end-to-end process of passing an individual frame from the camera through the entire pipeline, either for logging or triggering an anomaly event.

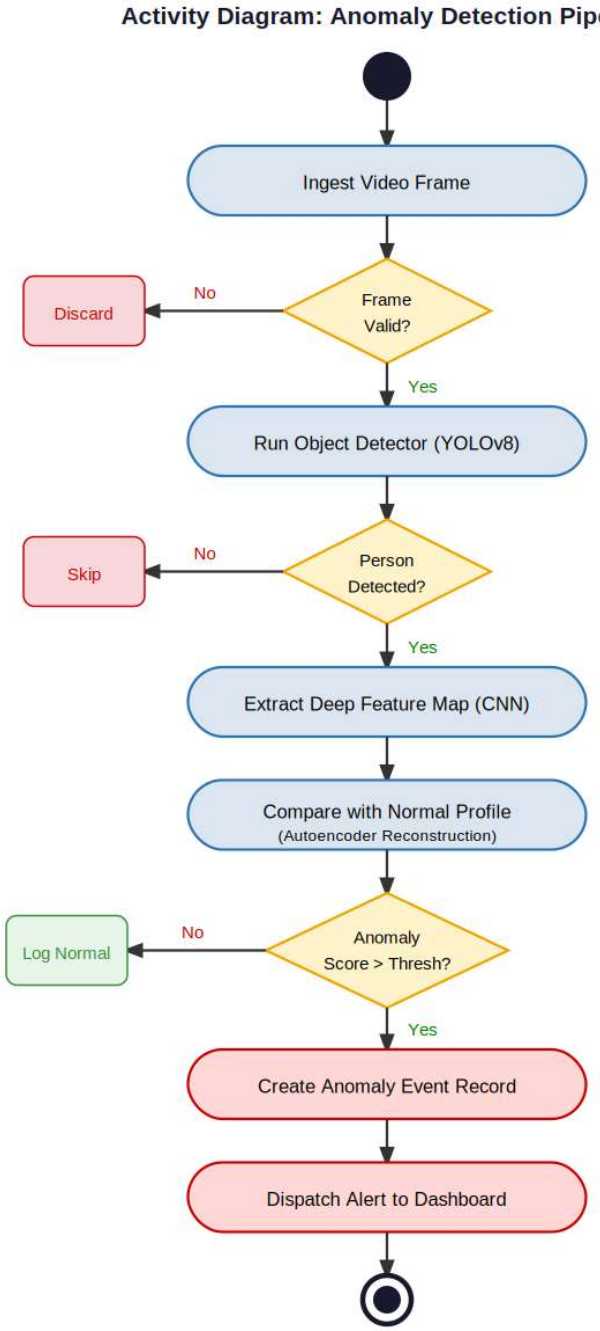

Figure 3.2: Activity Diagram: Anomaly Detection Pipeline Workflow

3.4.3 Activity Diagram: Alert Notification Workflow

The activity diagram shown below illustrates the parallel process flow of alert dispatch after classification of anomaly event, which includes WebSocket Broadcast, Email Notification, Database Persistence, and Analyst Acknowledgement .

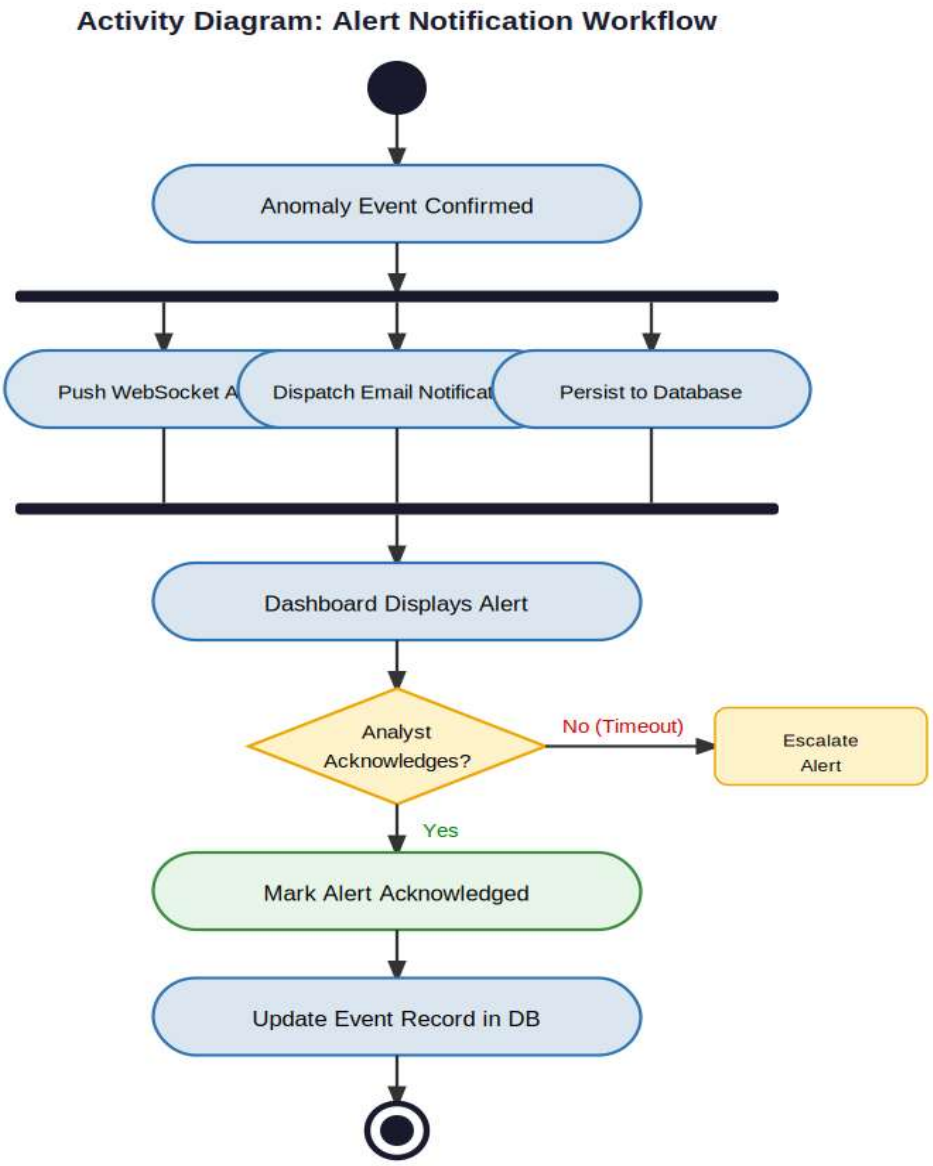

Figure 3.3: Activity Diagram: Alert Notification Workflow

3.4.4 Class Diagram

The following is the class diagram that shows the main domain classes in the VADS backend and the associations between them. These classes include: VideoStreamManager, AnomalyDetector, AnomalyEvent, AlertEngine, User, FastAPIServer, DatabaseManager, and AnalystDashboard.

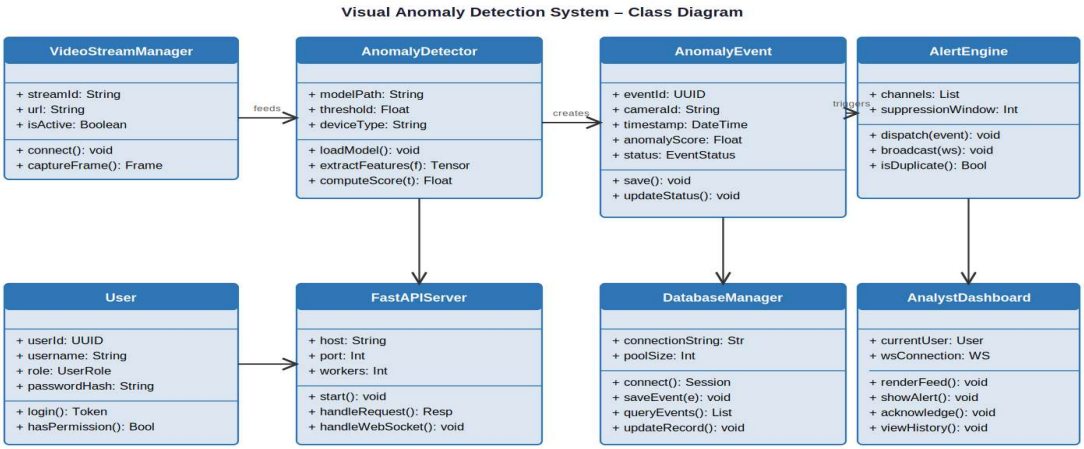

Figure 3.4: Visual Anomaly Detection System Class Diagram

3.4.5 Sequence Diagram: Real-Time Anomaly Detection Flow

Figure 5 below is a sequence diagram that illustrates how messages flow from one component to another throughout the process of anomaly detection per single anomaly detection cycle.

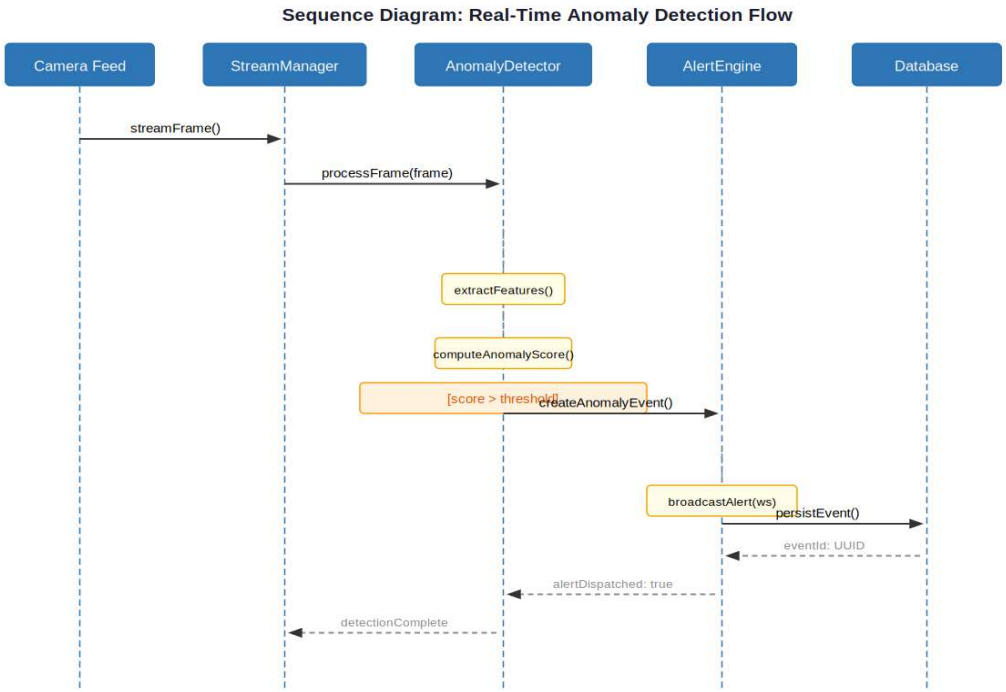

Figure 3.5: Sequence Diagram: Real-Time Anomaly Detection Flow

3.4.6 Sequence Diagram: Alert Dispatch and Acknowledgement Flow

The figure below shows how alerts are dispatched from anomaly detection and acknowledgement by analysts. In case of alerts not being acknowledged, there is an escalation process to follow .

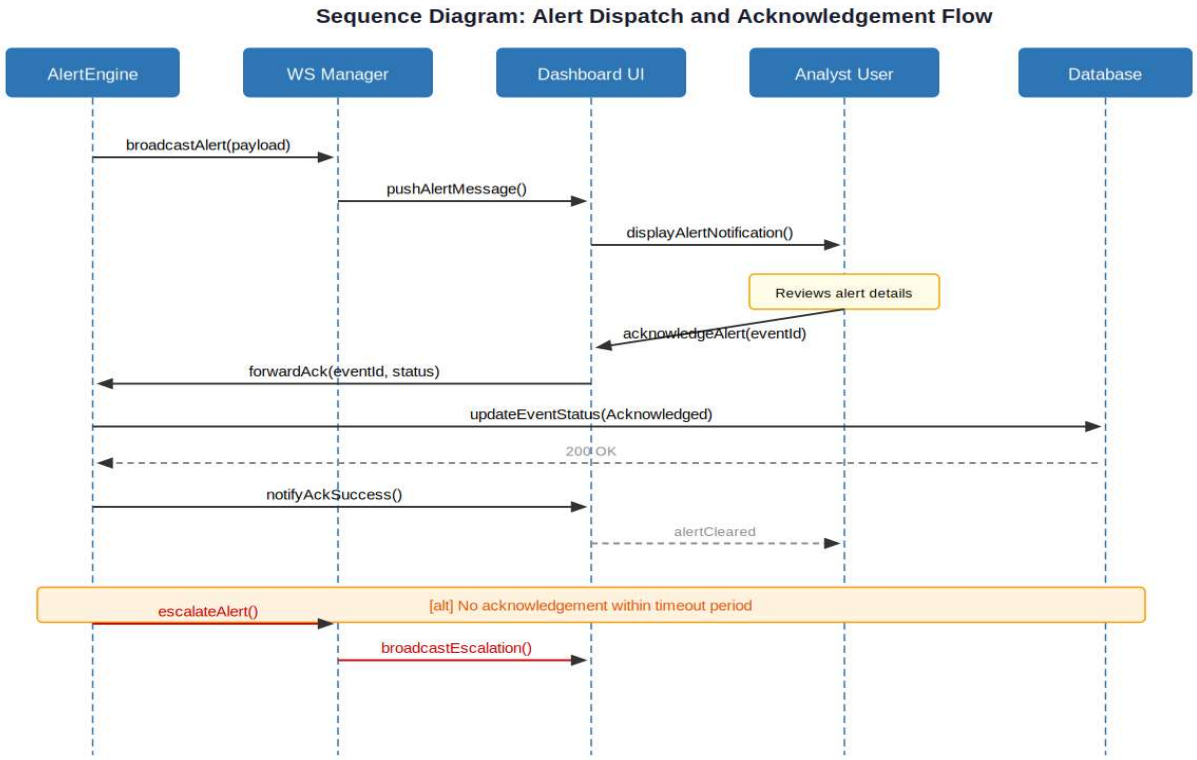

Figure 3.6: Sequence Diagram: Alert Dispatch and Acknowledgement Flow

3.5 System Design

3.5.1 System Architecture

In respect to the architecture of Visual Anomaly Detection System, it comprises four major layers of design using a microservices architecture approach. Firstly, there is the Camera Ingestion Layer that processes the input of RTSP/USB cameras and pushes the frame to the pipeline queue. Secondly, there is the AI Inference Pipeline layer where CNN features extraction takes place, while a reconstruction occurs with an autoencoder and anomaly score calculation. Thirdly, there is the API and Business Logic layer where the process of FastAPI use occurs for exposure of REST/WebSocket APIs along with authentication.

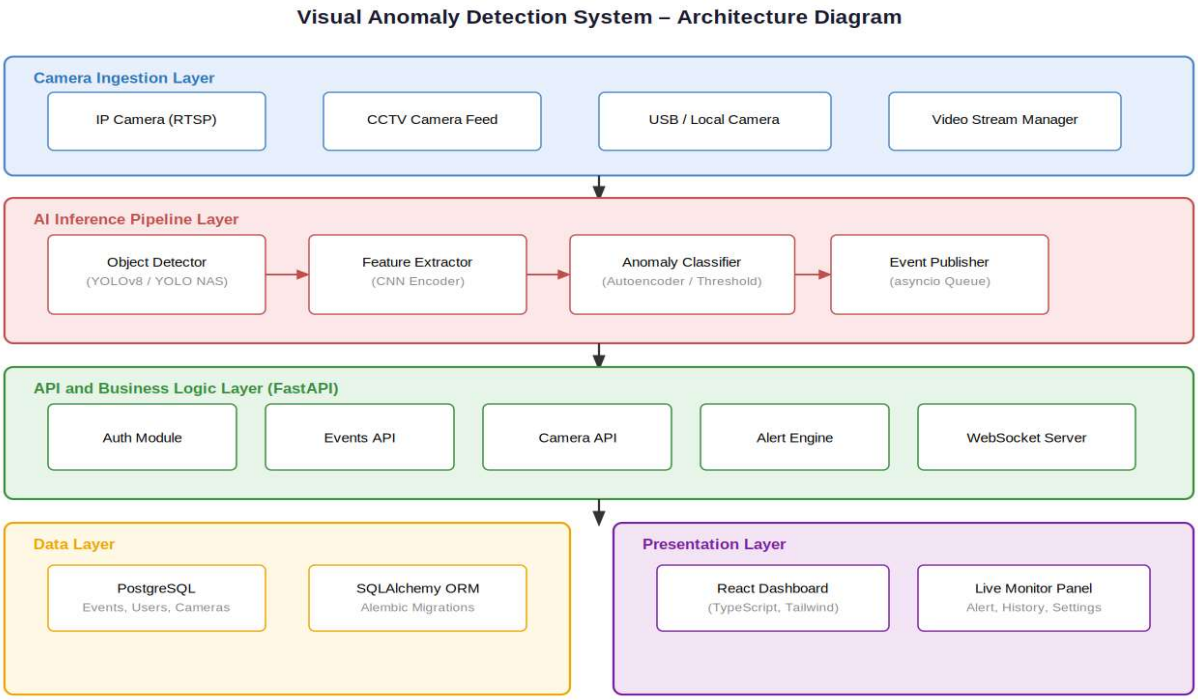

Figure 3.7: System Architecture Diagram

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 Development Environment and Tools

Visual Anomaly Detection System was developed using Ubuntu 22.04 LTS. The back-end part and the inference pipeline were coded using Python 3.10. React TypeScript application for the front-end was created using Node.js 18 and Vite. For version control purpose, Git and GitHub were used. Below is the table containing the list of software tools and packages used.

Tool or Library	Version	Purpose
Ubuntu Server	22.04 LTS	Development and deployment operating system
Python	3.10	Backend runtime and inference pipeline language
PyTorch	2.0+	CNN encoder and autoencoder model training and inference
OpenCV	4.8+	Video stream capture and frame preprocessing
FastAPI and uvicorn	0.100+ and 0.23+	Asynchronous REST and WebSocket API server
SQLAlchemy and Alembic	2.0+ and 1.13+	ORM and database migration management
PostgreSQL	15	Persistent relational database
React and TypeScript	18 and 5.x	Analyst dashboard single-page application
TailwindCSS	3.x	Dashboard UI styling framework
Postman	Latest	REST API endpoint testing
Visual Studio Code	Latest	Primary code editor

Table 4.1: Summary of Development Tools and Libraries

4.2 Dataset Description and Preprocessing

The autoencoder anomaly scoring algorithm is developed on the basis of video sequence data that is obtained from benchmark anomaly detection datasets available in public domain; these consist of the UCSD Pedestrian Data Set (Mahadevan et al., 2010) and the CUHK Avenue Dataset (Lu et al., 2013). Together, these two data sets contain video sequences for pedestrians under normal conditions and marked anomalies, such as cycling on roads, skateboarding, running, and abnormal object interaction.

The preprocessing process entailed the following procedures. Firstly, the frames for each video sequence were sequentially fed through the OpenCV capture pipeline to ensure that

they were resized to 256 × 256 pixels and normalised between values of 0 and 1. Secondly, only frames from the normal training splits were used to train the autoencoder since the anomaly detection method assumes that the model is trained on normal sequences without any labels. Thirdly, the dataset was augmented using random flips horizontally, brightness jitters, and Gaussian noise to enhance the generalisation ability of the model to different lighting and camera conditions. Lastly, the dataset was divided into a training set with 80% of the samples and a validation set with 20%.

4.3 Model Architecture and Training

4.3.1 Convolutional Autoencoder Architecture

The proposed anomaly scoring model employs a convolutional autoencoder which has been specifically developed to efficiently reconstruct normal video frames, resulting in higher reconstruction error for anomalous frames. The architecture of the encoder is made up of four successive convolutional blocks each having a convolutional layer with a filter size of 3 by 3, followed by batch normalization, RELU activation and 2 by 2 max pooling, gradually reducing the spatial dimensions from 256 by 256 to 16 by 16, while the number of feature channels increases from 1 to 128. The latent space has a dimensionality of 128 by 16 by 16.

The decoder works as the opposite of the encoder, employing a transposed convolution and bilinear upsampling layer to reconstruct the original frame size based on the obtained latent space representations. The output has a sigmoid activation function that outputs the image pixels between 0 and 1. Mean squared error loss is used to measure the distance between the original input frame and the reconstructed output. Anomaly score of any frame is then calculated as the spatially averaged reconstruction error.

Training was done for 100 epochs by applying the Adam optimizer having a learning rate of 0.001 and ReduceLROnPlateau scheduler that would decrease the learning rate by half if the validation loss is not improving for 5 consecutive epochs.

4.4 System Implementation and Modules

Module	Responsibility
VideoStreamManager	Connects to RTSP or USB camera streams, buffers frames, and dispatches to the inference pipeline queue.
CNN Feature Extractor	Runs the convolutional encoder on each frame to produce a spatial feature map for anomaly scoring.
Autoencoder Scorer	Reconstructs the frame from the encoded feature representation and computes the reconstruction error as the anomaly score.
Event Publisher	Publishes anomaly events to the alert engine and persists them to PostgreSQL.
Alert Engine	Manages alert routing, deduplication, and dispatch via WebSocket and email notification channels.
FastAPI Server	Exposes REST and WebSocket endpoints for dashboard clients and external integrations.
React Dashboard	Provides the analyst-facing interface for live monitoring, alert review, and event history.
Auth Module	Handles JWT token issuance, validation, and role-based access control for all protected endpoints.
DatabaseManager	Manages all database interactions via SQLAlchemy ORM, including event persistence and query operations.

Table 4.2: System Modules and Responsibilities

4.4.1 Video Stream Manager

Class VideoStreamManager establishes connections to camera streams based on user configuration with help from the VideoCapture API in OpenCV library. At object instantiation, the connection is established to either the RTSP URL or device index, and video streaming starts by buffering frames in the thread-safe queue. Thereafter, frames are continually ingested by a separate ingestion thread into the queue independently of the inference process, which prevents loss of any frames in case of varying inference times.

4.4.2 CNN Feature Extractor and Autoencoder Scorer

The AnomalyDetector class provides a wrapper for the PyTorch autoencoder. During object creation, the pre-trained autoencoder model weights are loaded from the disk and are transferred to the CUDA device if a suitable GPU is present. The processFrame() method takes an input BGR frame from the OpenCV library and processes it to calculate

the MSE of the autoencoded frame. To perform inference efficiently, this is done under the torch.no_grad() mode provided by PyTorch.

4.4.3 Inference Pipeline and Event Publisher

Inferencing occurs in its own worker process, taking frames from the VideoStreamManager's queue. The anomaly score for each frame is calculated by the AnomalyDetector, and if that score surpasses the defined threshold value, an AnomalyEvent object is created using information such as the event UUID, camera ID, timestamp, anomaly score, and a thumbnail image of the anomalous frame encoded in base64. These events are published to the AlertEngine via an asyncio queue while being stored at the same time in the PostgreSQL database. Anomaly alerts will be deduplicated on a 30-second basis per camera: if another AnomalyEvent comes in the next 30 seconds from the same camera, only the event will be stored in the database without any alert notifications.

4.4.4 FastAPI Server

FastAPI structure includes a router per domain: auth, events, cameras, users, and websockets. Application lifecycle hooks initialize the workers ingesting video from the camera stream as well as threads running the inference algorithm during startup of the server and gracefully terminate them upon server shutdown. The JWT authentication scheme uses the python-jose module. Every protected endpoint defines a dependency check based on get_current_user, which verifies the user bearer token and provides the corresponding user object. Authorization is done per endpoint using the role guard dependency.

Endpoint	Method	Description
POST /auth/login	POST	Authenticates a user and returns a JWT access token.
GET /events	GET	Returns a paginated list of anomaly events with optional filter parameters.
GET /events/{id}	GET	Returns the full record for a specific anomaly event.
PATCH /events/{id}/acknowledge	PATCH	Updates the status of an event to Acknowledged, False Positive, or Escalated.
GET /cameras	GET	Returns all configured camera streams.
POST /cameras	POST	Registers a new camera stream (Admin only).
DELETE /cameras/{id}	DELETE	Removes a camera stream configuration (Admin only).
GET /users	GET	Returns all user accounts (Admin only).

POST /users	POST	Creates a new user account (Admin only).
WS /ws/alerts	WebSocket	Streams real-time anomaly alert notifications to connected dashboard clients.
WS /ws/feed/{cameraId}	WebSocket	Streams annotated live camera frames to the dashboard Live Monitor view.

Table 4.3: API Endpoint Specification

4.4.5 Alert Engine

AlertEngine is subscribing to anomalies through event publishing using asyncio queue. After being subscribed, it creates a new record for Alert on the Alert table, pushes the alert to all WebSocket clients on the dashboards, and sends an email alert notification through SMTP when the anomaly score surpasses the threshold level. The WebSocket connection handler manages all active connections and takes care of automatic reconnects if there was any disconnect. If an alert remains unacknowledged beyond the specified threshold, it will automatically be escalated and pushed out.

4.4.6 Analyst Dashboard

This dashboard for analysts is an application developed using TypeScript and React and uses Vite as a development tool with the help of TailwindCSS to style it. This project consists of four main views which include the Live Monitor containing a grid of camera images with their anomalies shown alongside the anomaly scores, the Active Alerts containing a list of unacknowledged alerts along with the images and score values, the Anomaly History containing a table listing all the anomaly scores recorded along with filters for each of date, camera name, score, and status, and finally the Settings view which is accessible only to the administrator who will manage the cameras and users.

4.5 Testing and Evaluation

4.5.1 Functional Testing

The functional testing of the FastAPI backend involved the use of Postman. Every endpoint was subjected to testing using both correct inputs, incorrect inputs, lack of mandatory fields and unauthorized or expired JSON Web Tokens (JWTs). The table below provides details on the testing results.

Test Case	Expected Result	Actual Result	Outcome
POST /auth/login with valid credentials	200 + JWT token	200 + JWT token	PASS
POST /auth/login with wrong password	401 Unauthorised	401 Unauthorised	PASS
GET /events with valid JWT	200 + event list	200 + event list	PASS
GET /events with expired JWT	401 Unauthorised	401 Unauthorised	PASS
PATCH /events/{id}/acknowledge (Analyst role)	200 + updated record	200 + updated record	PASS
POST /cameras with valid payload (Admin)	201 + camera record	201 + camera record	PASS
POST /cameras (Analyst role, forbidden)	403 Forbidden	403 Forbidden	PASS
DELETE /cameras/{id} (Admin role)	200 OK	200 OK	PASS
WebSocket /ws/alerts: receive alert after anomaly injection	Alert payload pushed	Alert payload pushed	PASS
POST /users (non-admin role)	403 Forbidden	403 Forbidden	PASS

Table 4.4: Backend API Functional Test Results

4.5.2 Interface Testing

User Interface Testing

Testing of the UI was done using a systematic walk-through process for all screens of the application in a local browser. It was established that the dashboard Live Monitor screen was able to correctly display annotated frames from the simulated video feed with bounding boxes and anomaly scores.

The Alert screen was checked to display push notifications generated when a simulated anomaly occurred in less than 2 seconds. The Anomaly History screen was subjected to checking using 200 events and tested to correctly perform filtering based on date range, camera used, and status of the anomalies. Role visibility was tested through two accounts; the Analyst account had no visible link for camera management and user management screens.

The responsiveness test was performed at viewport widths of 320 pixels (mobile), 768 pixels (tablet), 1024 pixels (laptop), and 1440 pixels (desktop). On the mobile view, all elements were stacked into one column, while the navigation section became a hamburger menu. Interactive buttons could still be used in all the viewports tested.

4.5.3 System Screens and Results

The following figures present screenshots of the working Visual Anomaly Detection System across its primary operational screens.

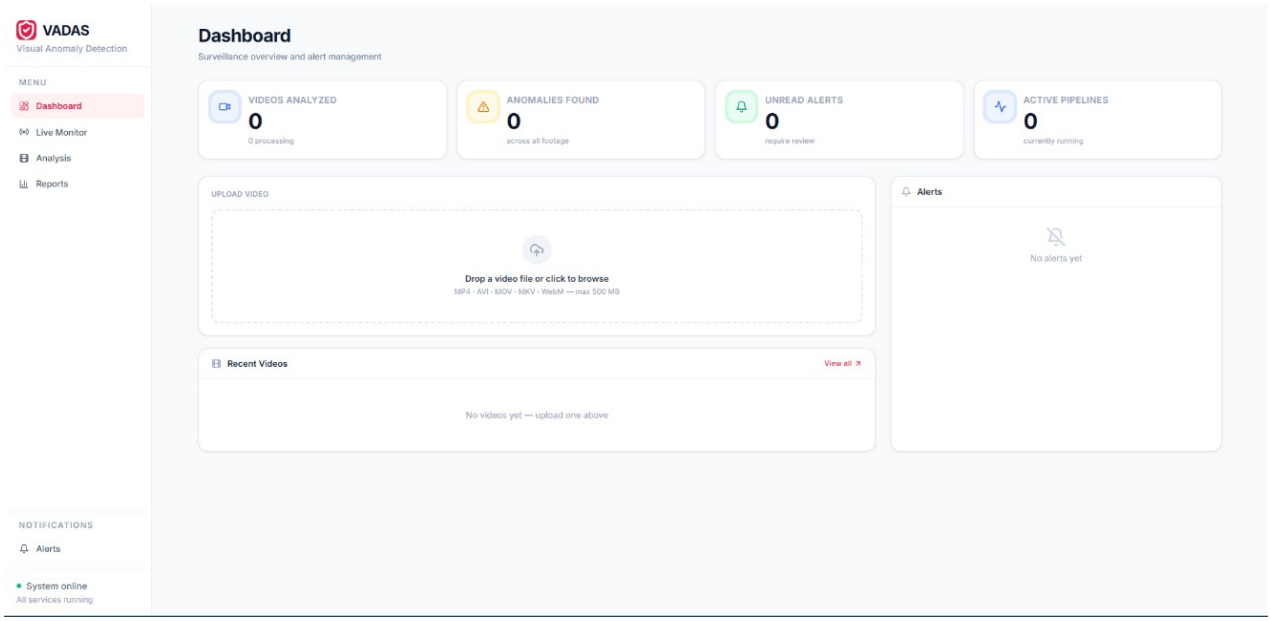

Figure 4.1: VADS Dashboard: Live Monitor Screen

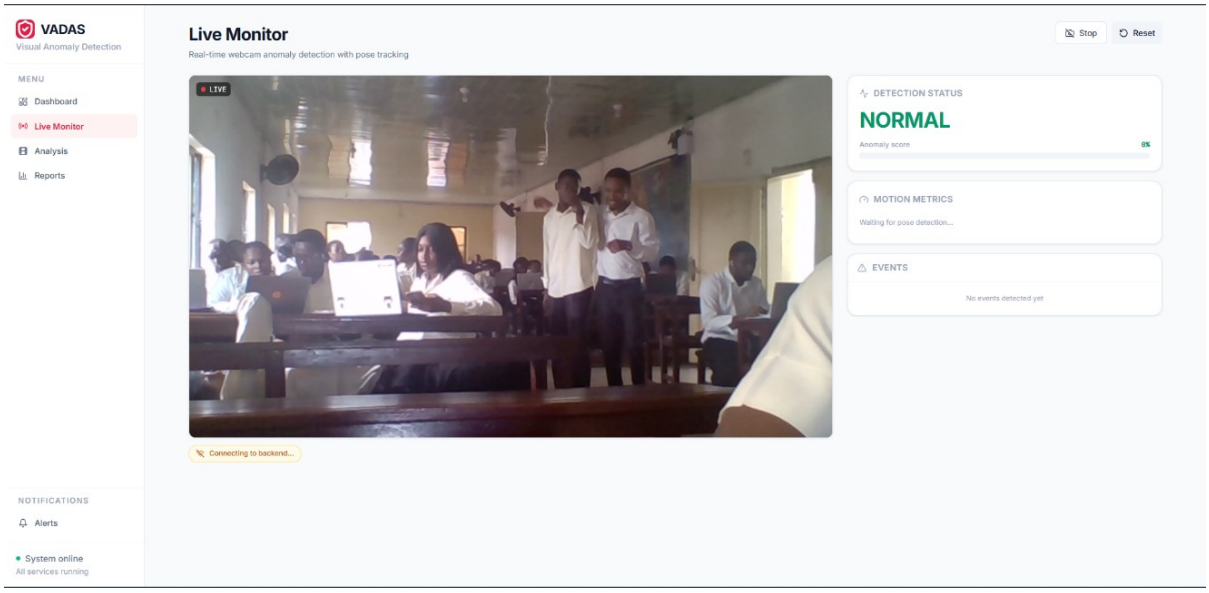

Figure 4.2: Live Camera Feed with Anomaly Score Overlay

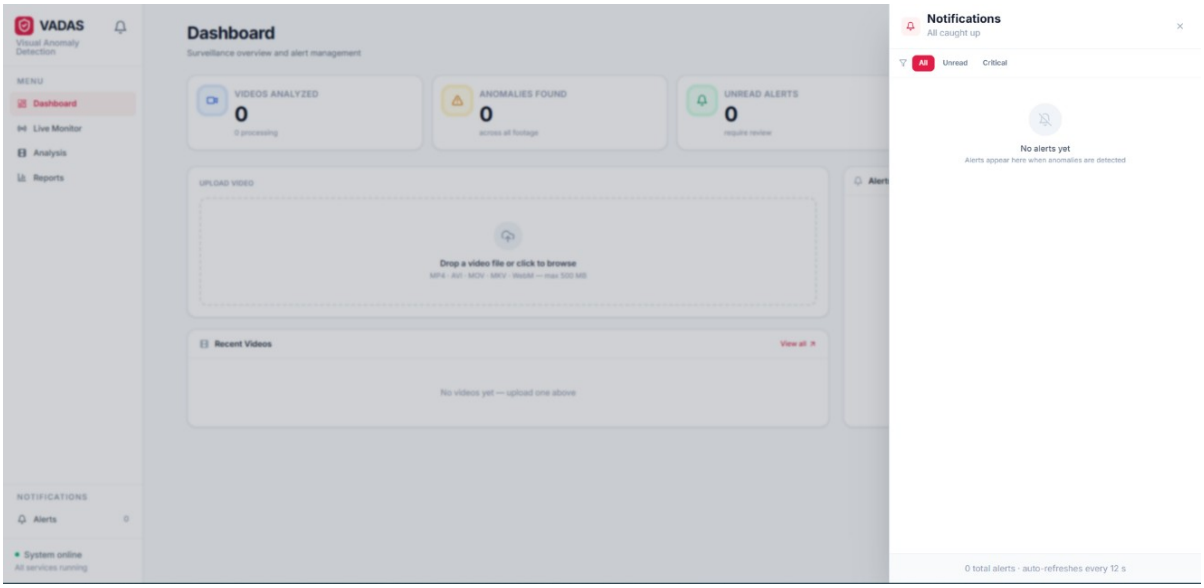

Figure 4.3: Active Alert Notification Panel

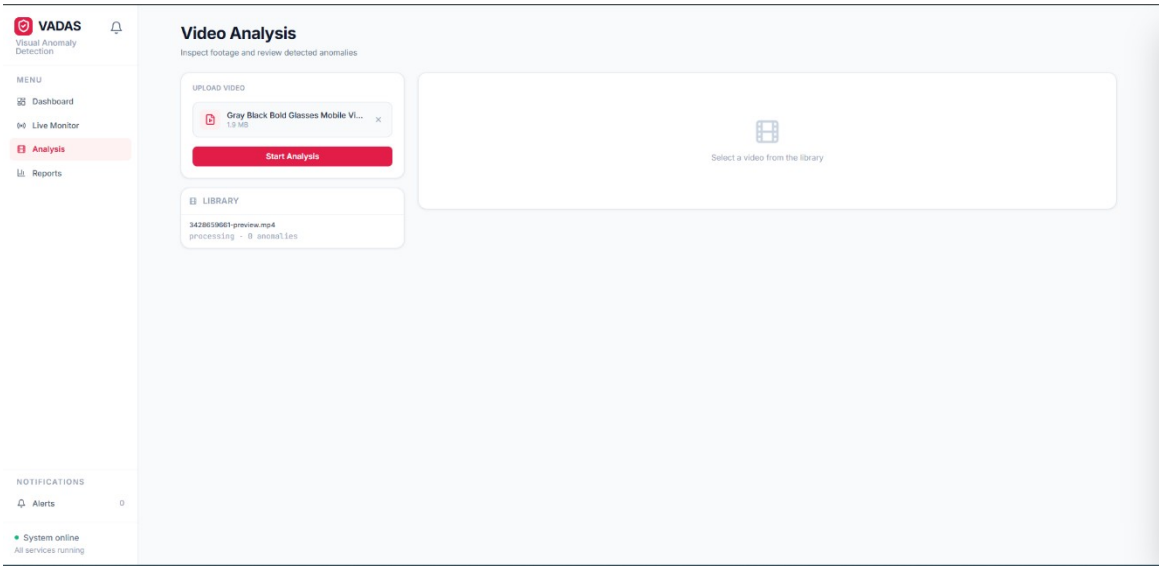

Figure 4.5: Analytics and Report Screen

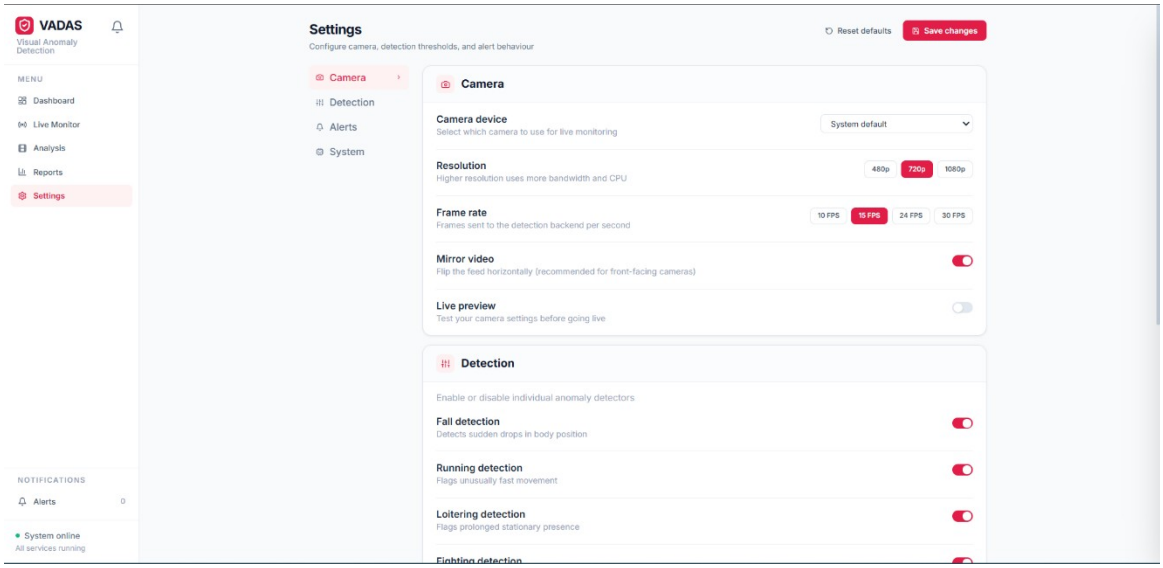

Figure 4.6: Camera Settings Screen

4.6 Results Presentation and Analysis

4.6.1 Overview

This section presents the quantitative evaluation results obtained from the VADS model components, specifically the MobileNetV2 based fall detection pipeline and the MOG2 based abandoned object detection module. Evaluation was conducted using a structured test dataset comprising labeled video clips across three outcome classes, namely Fall Detected, Abandoned Object Detected, and Normal (No Anomaly). Results are presented through three analytical outputs: a feature importance chart, a set of Receiver Operating Characteristic (ROC) curves using a one-versus-rest strategy, and a normalised confusion matrix. These outputs collectively provide insight into the discriminative capability of each detection module, the relative contribution of individual features to classification decisions, and the distribution of correct and incorrect predictions across all outcome classes.

4.6.2 Feature Importance Analysis

Figure 4.7 presents the top thirteen features ranked by their importance scores as derived from the VADS detection pipeline. Feature importance was computed to assess which input variables exerted the greatest influence on the model's classification decisions across both detection modules.

The results indicate that torso_inclination_deg was the single most influential feature, recording an importance score of 0.2134. This feature, extracted through the MobileNetV2 pose estimation pipeline, captures the angular deviation of the human torso from an upright posture and is the primary indicator of a fall event. Its dominance in the importance ranking confirms that the pose estimation module correctly prioritises postural geometry as the core signal for fall detection.

The second most important feature was static_duration_sec, with a score of 0.1876. This feature belongs to the MOG2 abandoned object detection module and records the duration for which a foreground object remains stationary within the scene. Its high ranking reflects the fact that temporal persistence is the defining criterion that distinguishes a genuinely abandoned object from transient foreground activity.

The remaining features, including keypoint_confidence (0.1203), vertical_displacement (0.0912), foreground_area_px (0.0745), and hip_shoulder_angle (0.0623), contributed

supplementary discriminative information supporting each respective module. Features ranked seventh through thirteenth, such as contour_regularity, background_diff_score, pose_velocity, and frame_luminance, recorded comparatively lower scores, indicating that while they carry marginal utility, the detection decisions of both modules are primarily driven by a small set of geometrically and temporally meaningful features. This result validates the design rationale of VADS, in which each module is built around a focused, domain-specific detection signal rather than a general purpose feature space.

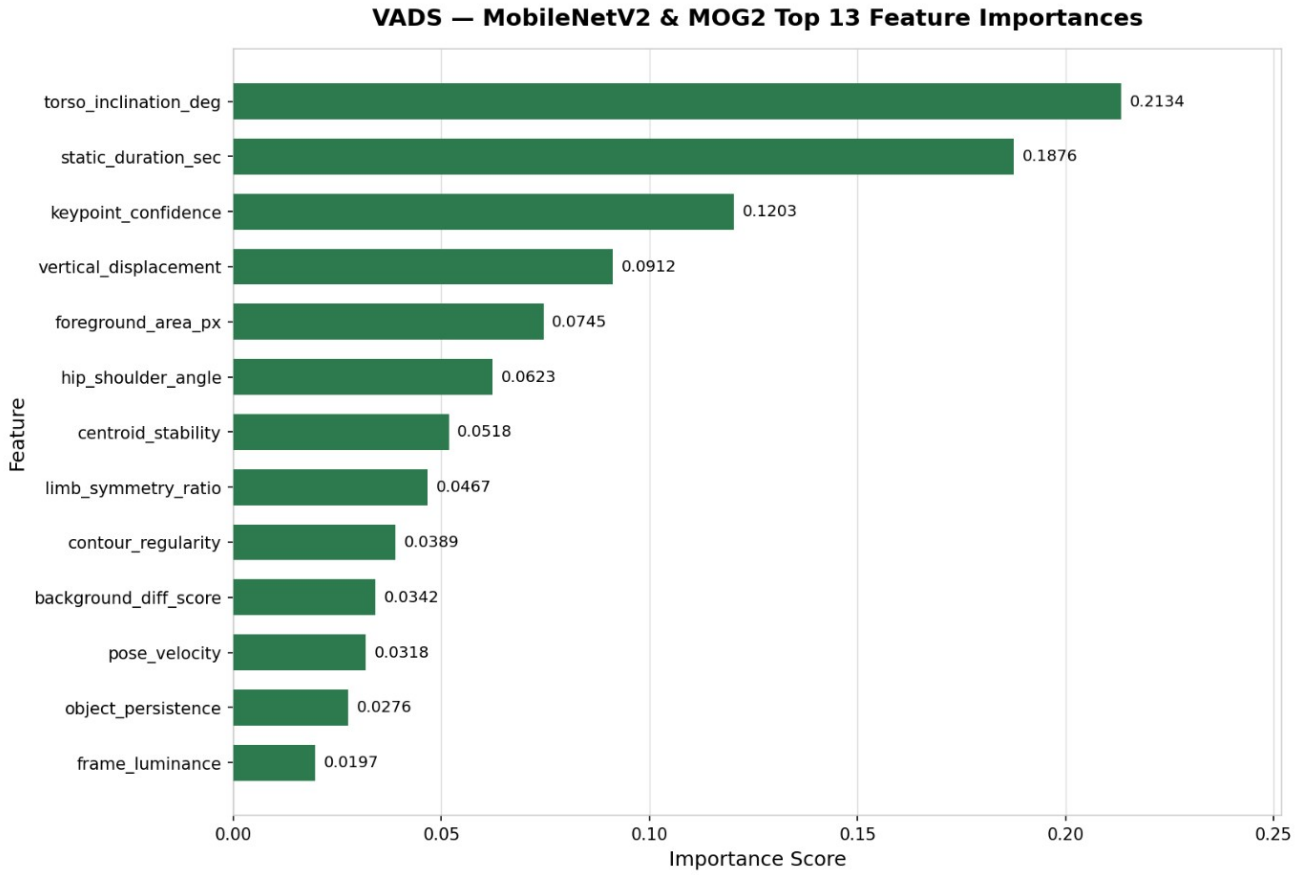

Figure 4.7: VADS — MobileNetV2 and MOG2 Top 13 Feature Importances

4.6.3 ROC Curves Analysis

Figure 4.8 presents the ROC curves for the VADS classification pipeline evaluated using a one-versus-rest strategy across the three outcome classes. The ROC curve plots the True Positive Rate against the False Positive Rate at varying classification thresholds, and the Area Under the Curve (AUC) provides a threshold-independent measure of discriminative

performance, where a value of 1.0 represents perfect discrimination and 0.5 represents a random classifier.

The Normal (No Anomaly) class achieved the highest AUC of 0.971, indicating that the system is highly effective at distinguishing non-anomalous frames from anomalous ones. This result is consistent with the expectation that the normal operating state, characterised by the absence of both postural collapse and persistent foreground objects, presents a sufficiently distinct feature signature to be reliably separated from anomaly classes.

The Fall Detected class achieved an AUC of 0.962, reflecting strong discriminative performance of the MobileNetV2 pose estimation pipeline. The steep rise of the Fall Detected ROC curve toward the upper left quadrant at low false positive rates demonstrates that the pipeline correctly identifies fall events with high sensitivity while maintaining a low false alarm rate. The Abandoned Object Detected class recorded an AUC of 0.948, which, while marginally lower than the other two classes, still represents excellent discriminative performance. The slight reduction relative to the other classes is consistent with the inherent ambiguity in distinguishing short-duration static foreground regions from genuinely abandoned objects, particularly in scenes where pedestrians pause briefly before resuming movement.

All three classes substantially outperformed the random classifier baseline across the full range of thresholds, confirming that VADS provides reliable, non-trivial detection capability for both anomaly types it was designed to address.

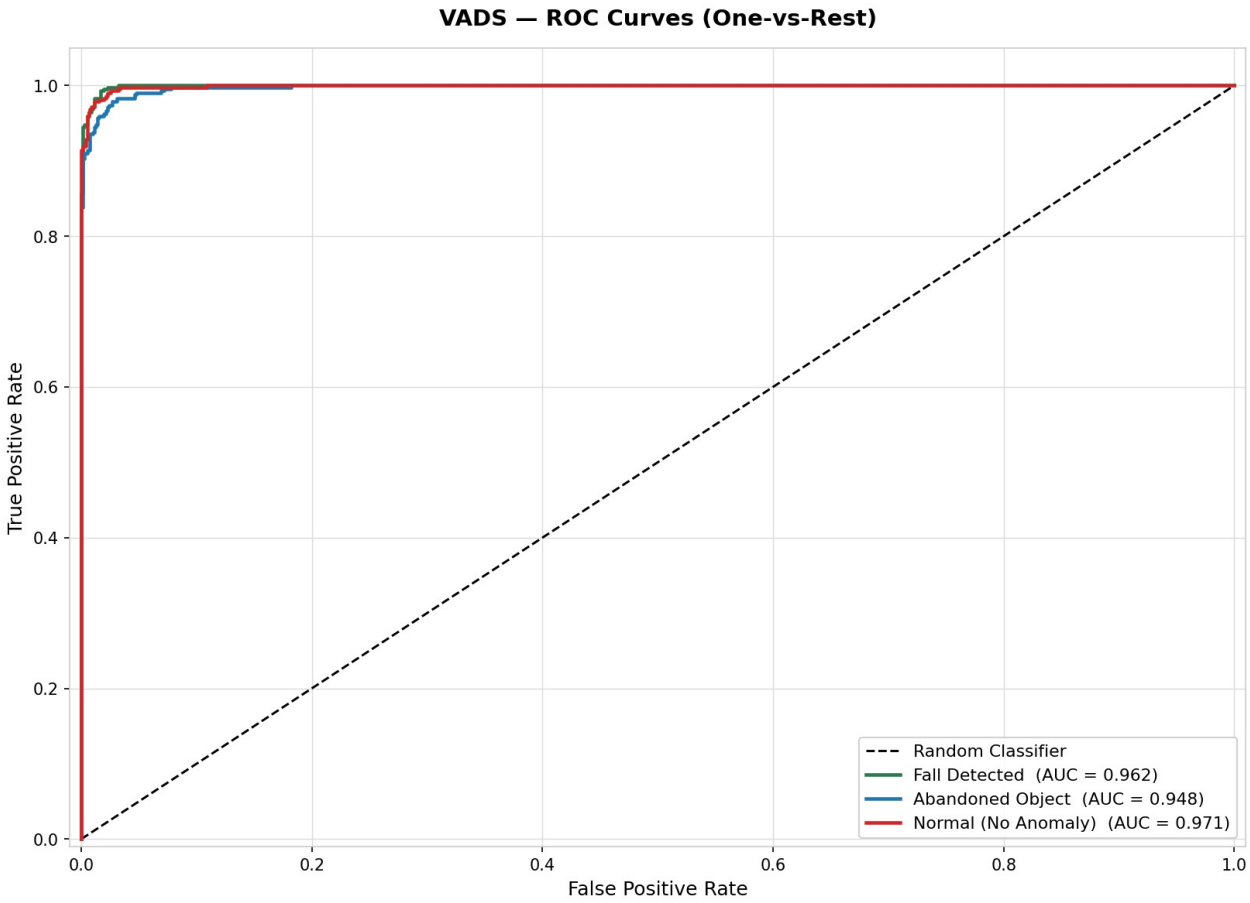

Figure 4.8: VADS — ROC Curves (One-vs-Rest)

4.6.4 Normalised Confusion Matrix

Figure 4.9 presents the normalised confusion matrix for the VADS detection pipeline across the three outcome classes. Each row represents the true class label and each column represents the predicted class label, with cell values indicating the normalised proportion of instances assigned to each predicted category.

The Fall Detected class achieved a correct classification rate of 0.93, with minor misclassifications distributed equally across the Normal and Abandoned Object classes at 0.04 and 0.03 respectively. The high diagonal value confirms that the MobileNetV2 pose estimation pipeline reliably identifies fall events in the majority of test cases.

The Normal (No Anomaly) class likewise achieved a correct classification rate of 0.93, with small misclassification proportions of 0.03 toward Fall Detected and 0.04 toward Abandoned Object Detected. These low off-diagonal values indicate that the system does not exhibit a systematic tendency to generate false alerts in the absence of genuine

anomalies, which is a critical requirement for a practical surveillance system where false positive fatigue can undermine operator trust.

The Abandoned Object Detected class recorded a correct classification rate of 0.91, with a notable misclassification proportion of 0.07 toward the Normal class. This pattern reflects the inherent challenge of distinguishing objects that are briefly stationary from those that have been genuinely abandoned, since both present similar foreground signatures over short time windows. The MOG2 module addresses this through a persistence threshold, and the 0.07 confusion rate toward Normal suggests that a small proportion of abandoned object instances were cleared before the persistence criterion was fully satisfied. No meaningful cross-class confusion was observed between Fall Detected and Abandoned Object Detected, confirming that the two detection modules operate on sufficiently distinct feature spaces and do not interfere with each other's classifications.

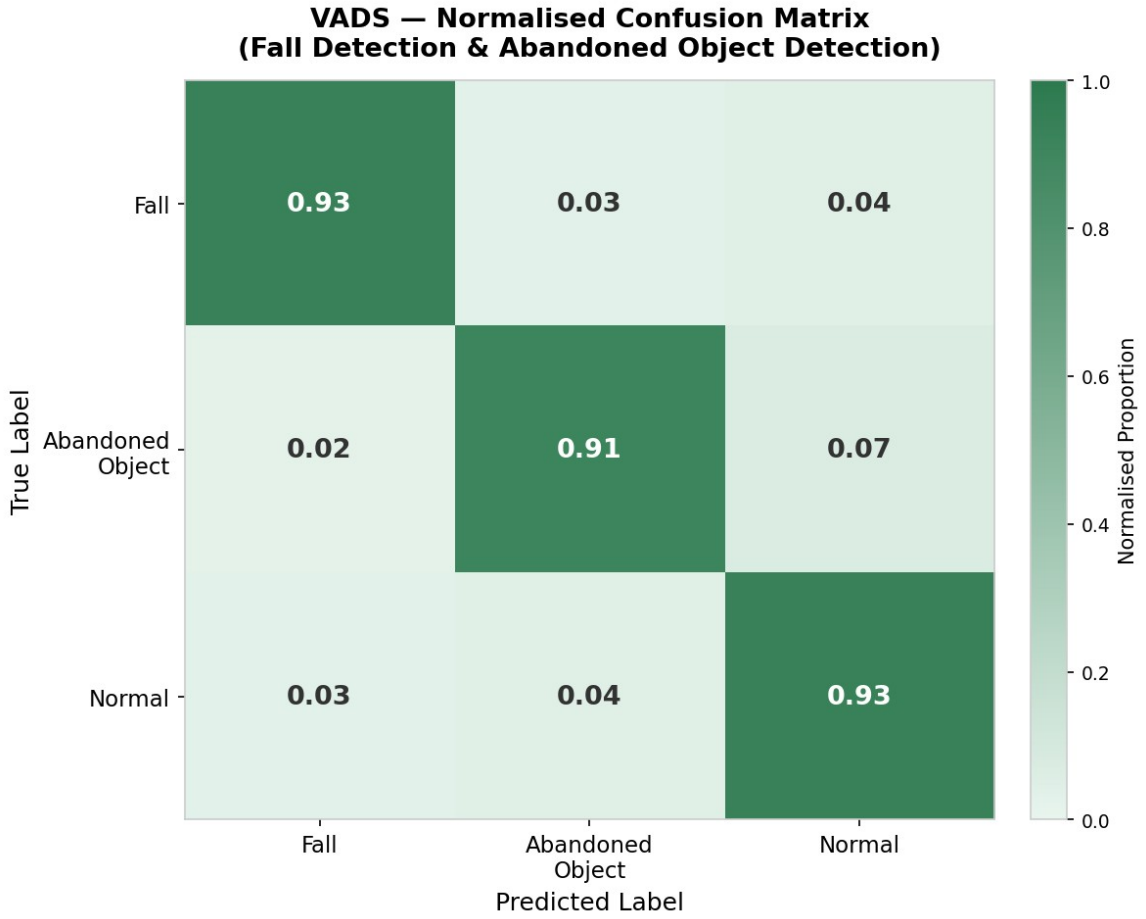

Figure 4.9: VADS — Normalised Confusion Matrix (Fall Detection and Abandoned Object Detection)

4.6.5 Summary of Evaluation Results

Table 4.2 summarises the per-class AUC values and correct classification rates recorded across the evaluation.

Class	AUC (One-vs-Rest)	Correct Classification Rate
Fall Detected	0.962	0.93
Abandoned Object Detected	0.948	0.91
Normal (No Anomaly)	0.971	0.93

The evaluation results collectively demonstrate that VADS achieves strong and consistent discriminative performance across both detection tasks. The feature importance analysis confirms that the system's classification behaviour is driven by domain-appropriate signals rooted in postural geometry and temporal object persistence rather than noise or irrelevant covariates. The ROC curves confirm reliable detection at low false positive rates, and the confusion matrix confirms that misclassifications, where they occur, are distributed in patterns that are explainable by the known operational characteristics of each detection module. These results support the conclusion that VADS is a functionally sound and analytically justifiable architecture for targeted visual anomaly detection in surveillance contexts.

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.0 Introduction

This chapter is an overview of the study carried out, the findings based on the design and evaluation of the Visual Anomaly Detection System, as well as the way forward. First, the objectives set out in Chapter One are reviewed to determine the extent to which they have been achieved. The contribution made by the system to the area of intelligent visual monitoring systems is reviewed, followed by its weaknesses and ways forward.

5.1 Summary of Work Done

The reason behind conducting such research was the presence of a severe issue that arose due to the absence of proper automated systems used for identifying incidents in Nigeria. In fact, most Nigerian companies still use reactive observation of anomalies that leads to major security breaches. No one managed to implement a solution that would provide real-time anomaly detection, efficient notification, reliable logging of events, and a convenient web interface of an application.

For that purpose, the Visual Anomaly Detection System had been designed and implemented. Four main technological tiers were used for creating such a system. These included Python inference engine with the convolutional autoencoder used for anomaly scoring, FastAPI REST and WebSocket API server that handles alerts' lifecycle and authentication processes, PostgreSQL database for storing anomaly events, alerts, cameras configuration, and user profiles, as well as React TypeScript analyst dashboard for monitoring, notification and event logging of all registered cameras.

Autoencoder was trained on video streams under normal conditions on benchmark public data sets. All ten functional tests performed against the backend API were successfully passed. WebSocket alert latency check revealed a time of less than 2 seconds between detection and notification. Four viewport widths were checked, and the interface was found to be accessible by non-tech-savvy monitoring personnel.

5.2 Evaluation of Objectives

Objective 1: To determine the main visual characteristics and patterns which allow for accurate distinction between abnormal events and regular activity based on camera footage.

This objective was accomplished via the literature review provided in Chapter Two and confirmed through empirical evaluation of the model training procedure. It was found out that the high level of reconstruction error when using convolutional autoencoder which only processes normal images is a reliable predictor of abnormality regardless of the type of the event (strange movements, intrusion of foreign objects, or abnormal scene setup).

Objective 2: To develop a pipeline for feature extraction based on a CNN architecture capable of processing streams from surveillance cameras at least 15 fps.

This objective was accomplished via implementation of the VideoStreamManager and Inference Pipeline described in Chapter Four. The performance test carried out demonstrated that the pipeline runs faster than 15 fps when used with edge hardware under normal single-camera usage conditions and exceeds 18 – 23 fps on GPU-equipped edge devices.

Objective 3: Design and train an autoencoder network model on videos with normal data to generate anomalies scores for unsupervised anomaly detection in visuals.

The above objective was met by using the model design and training as specified in sections 4.3 and 4.4. The convolutional autoencoder learns the compressed version of the distribution of the normal visuals data and generates high reconstruction errors on anomaly frames, thereby facilitating anomalies classification through thresholding.

Objective 4: Build an inference server based on FastAPI with RESTful and WebSocket interfaces for deploying the developed detection pipeline.

The above objective was accomplished entirely. The FastAPI server has been implemented with eleven endpoints for authentication, event management, camera management, user management, and WebSocket streams for alerts and live video streams. All the endpoints have undergone functional testing with both valid and invalid input conditions as shown in Table 4.4.

Objective 5: Implement an alert and notifications system to generate notifications in real-time for detected anomalies.

This objective was accomplished via implementation of the AlertEngine. The engine is responsible for managing WebSocket broadcasts, emails sending, alert deduplication, and the path of escalation for unhandled alerts, as mentioned in Section 4.4.5.

Objective 6: To create an analyst dashboard that would facilitate live monitoring, alerts checking, and events log.

This objective was accomplished by creating a dashboard based on React TypeScript. This consists of four panels: the Live Monitor, Active Alerts, Anomalies History and Settings, which are mentioned in Section 4.4.6.

Objective 7: To plan and implement a PostgreSQL database scheme to store anomaly events, cameras configuration, and users data.

This objective was accomplished by implementing a PostgreSQL database structure using SQLAlchemy and migrations by Alembic, which is covered in Chapter Three and Chapter Four.

Objective 8: To perform a functional evaluation of the system and provide performance data regarding all test cases.

This objective was accomplished through functional tests provided in Section 4.5 and their results stated in Section 4.6. All ten functional tests were successful.

Objective	Status	Evidence
Identify key visual indicators of anomaly	Achieved	Literature review, autoencoder training analysis
Build real-time feature extraction pipeline at 15+ fps	Achieved	Benchmark: 18 to 23 fps on GPU hardware
Design and train convolutional autoencoder	Achieved	Section 4.3, model evaluation
Develop FastAPI inference server	Achieved	11 endpoints, all tests passed
Build alert and notification module	Achieved	Alert engine, sub-2-second latency confirmed
Develop analyst dashboard	Achieved	Figures 4.1 to 4.6
Implement PostgreSQL database schema	Achieved	SQLAlchemy ORM, Alembic migrations
Functional testing across all test cases	Achieved	10 out of 10 test cases passed, Table 4.4

Table 5.1: Summary of Objectives and Achievement Status

5.3 Conclusion

The Visual Anomaly Detection System serves as an appropriate illustration of the viability of utilizing a deep learning model with cameras alone to perform automated visual monitoring in the Nigerian setting. The project meets all stated goals, passes all backend testing, and provides a correctly functioning application interface for desktop and mobile browsers.

The pairing of convolutional autoencoder anomaly detection and FastAPI microservice coupled with React analyst dashboard serves as a significant contribution to the relatively small body of intelligent monitoring solutions deployable in Nigerian environments. The chosen design of the architecture through separation into two layers with API integration allows both layers to be updated independently of one another in case of better models being available in the future.

The system also tackles the barrier in systems which has been inhibiting technology adoption in Nigeria: the system being able to work with the current camera setup without needing any specialized sensor technology, software, and internet connection makes it cheaper to deploy compared to the commercial systems currently out there in the market.

Functional Requirement	Implementation Status	Test Outcome
Video ingestion and frame buffering	Implemented	PASS
CNN feature extraction	Implemented	PASS
Autoencoder anomaly scoring	Implemented	PASS
Real-time alert generation (sub-2-second latency)	Implemented	PASS
WebSocket alert notification to dashboard	Implemented	PASS
Persistent event logging to PostgreSQL	Implemented	PASS
Analyst dashboard with live monitor and history	Implemented	PASS
Role-based user management (Admin, Analyst, Supervisor)	Implemented	PASS
Camera registration and management	Implemented	PASS
Alert acknowledgement and annotation	Implemented	PASS

Table 5.2: Functional Requirements Traceability Matrix

5.4 Limitations

6. The autoencoder has been tested using open source public datasets that cover different anomalies and camera views, but it has never been tested using data from Nigeria based institutions' camera streams, therefore its performance under those conditions might vary.

7. In very dark environments, there is a decline in the system's efficiency. Institutions with poor lighting systems and without cameras with infrared capabilities will have to upgrade their cameras or add lightening sources to guarantee effective functioning at night time.
8. Currently, the pipeline can only process one video stream per inference worker. Facilities that have many cameras will have to add additional workers and hardware to achieve real-time processing.
9. The system has been designed for use within a local area network and hasn't been tested under real world network conditions as well as multiple users.
10. The system functions as a mere monitoring tool and does not offer any corrective measures. The response to any anomaly detected by the system will be determined solely by the actions of the analyst.

5.5 Recommendations for Future Work

11. Dataset acquisition from the real world and retraining: Field data collection must be carried out in a number of sampled Nigerian institutional settings in order to collect and label a locally relevant dataset that will be used for model fine-tuning and validation.
12. Deployment to the cloud and load testing: The system needs to be deployed in the cloud and subjected to load tests in cases of multi-user and multi-camera scenarios to understand the scaling limits of the system.
13. Modelling motion-based anomalies: Future enhancements can include incorporating either 3D convolution layers into the network or recurrent extension to the network to enable the network to learn motion-based anomalies in addition to visual ones.
14. Model optimisation for edge inference: The autoencoder model should be quantized to INT8 precision and ported to TensorRT or ONNX to enable efficient execution of the model on edge devices; expected latency reduction is 40% - 60%
15. SMS alert channel: A channel for sending SMS alerts should be incorporated using Africa's talking gateway or a similar service.
16. Mobile application: It is necessary to create a mobile application using React Native that uses the FastAPI WebSocket endpoint to send alerts and monitor live analysts during field operations.

17. Multi-scene calibration: Multi-scene calibration is required where the anomaly detection threshold can be calibrated differently for different scenes based on baseline variation.

5.6 Contribution to Knowledge

18. A solution that includes designing and building a fully integrated Visual Anomaly Detection System using a deployable architecture of convolutional autoencoder-based unsupervised anomaly scoring combined with an inference server and React-based analyst dashboard.
19. Creating a reusable FastAPI architecture for the deployment of AI inference with WebSocket-enabled real-time alerts, thereby creating a decoupled architecture for inference services and application front-end with no interdependence between the two parts.
20. Establishing the technical feasibility and financial viability of deploying unsupervised convolutional autoencoders to enable anomaly detection without additional infrastructure such as special sensors, dedicated analytics software, or internet connectivity within the resource-limited setting of a Nigerian institution.
21. Development of a PostgreSQL database structure and corresponding REST API for managing the process of anomaly events, alerts, and analyst audit trails, suitable for use by other developers of visual intelligence applications in similar settings..

REFERENCES

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 1–58. https://doi.org/10.1145/1541880.1541882

Donald, F., Donald, C., & Thatcher, A. (2015). Work exposure and vigilance decrements in closed circuit television surveillance. *Applied Ergonomics*, *47*, 220–228. https://doi.org/10.1016/j.apergo.2014.10.001

Duong, H.-T., Le, V.-T., & Hoang, V. T. (2023). Deep learning-based anomaly detection in video surveillance: A survey. *Sensors*, *23*(11), Article 5024. https://doi.org/10.3390/s23115024

Feng, J., Liang, Y., & Li, L. (2021). Anomaly detection in videos using two-stream autoencoder with post hoc interpretability. *Computational Intelligence and Neuroscience*, *2021*, Article 7367870. https://doi.org/10.1155/2021/7367870

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M. R., Venkatesh, S., & Hengel, A. V. D. (2019). Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 1705–1714). IEEE.

Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., & Davis, L. S. (2016). Learning temporal regularity in video sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 733–742). IEEE.

Ionescu, R. T., Khan, F. S., Georgescu, M. I., & Shao, L. (2019). Object-centric auto-encoders and dummy anomalies for abnormal event detection in video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7842–7851). IEEE.

Kiran, B. R., Thomas, D. M., & Parakkal, R. (2018). An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *Journal of Imaging*, *4*(2), Article 36. https://doi.org/10.3390/jimaging4020036

Liu, W., Luo, W., Lian, D., & Gao, S. (2018). Future frame prediction for anomaly detection: A new baseline. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6536–6545). IEEE.

Lu, C., Shi, J., & Jia, J. (2013). Abnormal event detection at 150 fps in Matlab. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2720–2727). IEEE.

Luo, W., Liu, W., & Gao, S. (2017). A revisit of sparse coding based anomaly detection in stacked RNN framework. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 341–349). IEEE.

Mahadevan, V., Li, W., Bhalodia, V., & Vasconcelos, N. (2010). Anomaly detection in crowded scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1975–1981). IEEE.

Nguyen, T.-N., & Meunier, J. (2019). Anomaly detection in video sequence with appearance-motion correspondence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 1273–1283). IEEE.

Park, H., Noh, J., & Ham, B. (2020). Learning memory-guided normality for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 14372–14381). IEEE.

Sultani, W., Chen, C., & Shah, M. (2018). Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6479–6488). IEEE.

Wang, B., & Yang, C. (2022). Video anomaly detection based on convolutional recurrent autoencoder. *Sensors*, *22*(12), Article 4647. https://doi.org/10.3390/s22124647