

BCSE498J Project-II

CRIME TREND ANALYSIS

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering with specialization in Data Science

by

21BCT0075 SOUMY SINHA

21BDS0129 C JEEVAN REDDY

Under the Supervision of

ANISHA M. LAL

Professor Grade 1

School of Computer Science and Engineering (SCOPE)



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

April 2025

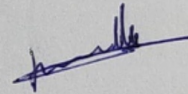
DECLARATION

I hereby declare that the project entitled **CRIME TREND ANALYSIS** submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of **Prof. ANISHA M. LAL**

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : 23/04/25



Signature of the Candidate

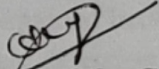
CERTIFICATE

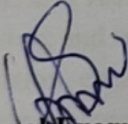
This is to certify that the project entitled **CRIME TREND ANALYSIS** submitted by C JEEVAN REDDY (21BDS0129), **School of Computer Science and Engineering**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him under my supervision during Winter Semester 2024-2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The project fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

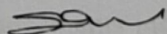
Place : Vellore

Date : 23/04/25


Signature of the Guide


Internal Examiner




External Examiner

Head Of Department – Data Science

EXECUTIVE SUMMARY

The increasing volume of surveillance footage in modern cities has outpaced the capacity of human operators to monitor it effectively, making it essential to develop intelligent systems that can automatically detect criminal activity in real time. This thesis introduces the Crime Trend Analysis System, an AI-powered surveillance solution that utilizes deep learning to recognize and classify criminal behavior from video footage and send immediate alerts to authorities when suspicious activity is detected.

The system leverages a hybrid architecture that combines ResNet50, a convolutional neural network (CNN) for spatial feature extraction, with a Long Short-Term Memory (LSTM) network for temporal sequence modeling. This design enables the system to not only recognize objects and actions within a single frame but also analyze how activities unfold over time—an essential capability for detecting dynamic events such as robbery, assault, vandalism, burglary, and abuse.

Once a sequence is analyzed, the system calculates a prediction confidence score. If the score exceeds a defined threshold (set at 0.8), the system saves the corresponding video frame and dispatches an automated SMS alert to designated law enforcement contacts via the Twilio API. This ensures timely intervention while minimizing false alarms.

The system was trained and tested on a curated dataset representing diverse crime scenarios. It achieved an overall classification accuracy of 86.3%, with particularly high precision in detecting crimes involving rapid motion. Performance testing demonstrated real-time capability, with frame processing latency maintained between 28–35 milliseconds, and alerts generated within 3–5 seconds of incident detection.

The project also considered practical deployment concerns, such as scalability, modularity, and real-time responsiveness. Ethical and legal issues—such as privacy, data retention, and regulatory compliance—were addressed, with recommendations for anonymization, encryption, and audit controls. Proposed future enhancements include multi-camera integration, edge device deployment, multimodal input (audio-visual), dashboard interfaces, and continuous learning via human feedback.

In conclusion, the Crime Trend Analysis System provides a significant step toward smarter, more responsive urban surveillance. By combining deep learning with real-time alerting, it has the potential to enhance public safety, reduce response times, and support law enforcement with automated, intelligent video analysis.

ACKNOWLEDGEMENTS

I am deeply grateful to the management of Vellore Institute of Technology (VIT) for providing me with the opportunity and resources to undertake this project. Their commitment to fostering a conducive learning environment has been instrumental in my academic journey. The support and infrastructure provided by VIT have enabled me to explore and develop my ideas to their fullest potential.

My sincere thanks to Dr. Jaisankar N, Dean - School of Computer Science and Engineering (SCOPE), for his unwavering support and encouragement. His leadership and vision have greatly inspired me to strive for excellence.

I express my profound appreciation to Dr Murali S, the Head of the Computer Science and Engineering with specialization in Data Science, for his insightful guidance and continuous support. His expertise and advice have been crucial in shaping throughout the course. His constructive feedback and encouragement have been invaluable in overcoming challenges and achieving goals.

I am immensely thankful to my project supervisor, Anisha M. Lal, for her dedicated mentorship and invaluable feedback. Her patience, knowledge, and encouragement have been pivotal in the successful completion of this project. My supervisor's willingness to share her expertise and provide thoughtful guidance has been instrumental in refining my ideas and methodologies. Her support has not only contributed to the success of this project but has also enriched my overall academic experience.

Thank you all for your contributions and support.

Name of the Candidate

C Jeevan Reddy

TABLE OF CONTENTS

Sl.No	Contents	Page No.
	Executive Summary	iii
	Acknowledgement	iv
	List of Figures	vii
	Abbreviations	viii
1.	INTRODUCTION	1
	1.1 BACKGROUND	1
	1.2 MOTIVATIONS	2
	1.3 SCOPE OF THE PROJECT	3
2.	PROJECT DESCRIPTION AND GOALS	5
	2.1 LITERATURE REVIEW	5
	2.2 RESEARCH GAP	6
	2.3 OBJECTIVES	7
	2.4 PROBLEM STATEMENT	8
	2.5 PROJECT PLAN	9
3.	TECHNICAL SPECIFICATION	11
	3.1 REQUIREMENTS	11
	3.1.1 Functional	11
	3.1.2 Non-Functional	12
	3.2 FEASIBILITY STUDY	13
	3.2.1 Technical Feasibility	13
	3.2.2 Economic Feasibility	14
	3.2.2 Social Feasibility	15
	3.3 SYSTEM SPECIFICATION	16
	3.3.1 Hardware Specification	17
	3.3.2 Software Specification	19
4.	DESIGN APPROACH AND DETAILS	21
	4.1 SYSTEM ARCHITECTURE	21
	4.1.1 Workflow	24
	4.2 DESIGN	27

	4.2.1 Data Flow Diagram	27
	4.2.2 Class Diagram	28
5.	METHODOLOGY AND TESTING	29
	5.1 MODULE DESCRIPTION	29
	5.2 TESTING	31
	5.2.1 Unit Testing	32
	5.2.2 Integration Testing	34
	5.2.3 Performance Testing	36
	5.2.4 Accuracy Testing	38
6.	PROJECT DEMONSTRATION	41
7.	RESULT AND DISCUSSION	42
	7.1 PREDICTION PERFORMANCE	42
	7.2 SYSTEM RESPONSIVENESS	43
	7.3 REAL-TIME DEMONSTRATION OUTCOME	44
	7.4 DISCUSSION OF FINDINGS	45
8.	CONCLUSION AND FUTURE ENHANCEMENTS	47
	8.1 CONCLUSIONS	47
	8.2 FUTURE ENHANSEMENTS	48
9.	REFERENCES	51
	APPENDIX A – SAMPLE CODE	52

List of Figures

Figure No.	Title	Page No.
1.	System Architecture Diagram	23
2.	Workflow Diagram	26
3.	Data Flow Diagram	27
4.	Class Diagram	28
5.	Demonstration Diagram	38

List of Abbreviations

2D	Two-Dimensional
CCTV	Closed-Circuit Television
CNN	Convolutional Neural Network
GDPR	General Data Protection Regulation
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network

Chapter 1

1. INTRODUCTION

1.1 BACKGROUND

Crime prevention has long been a cornerstone of community safety and urban development, playing a vital role in maintaining public order and fostering trust between citizens and law enforcement. As cities expand and societies become more complex, the challenges associated with ensuring public safety have grown significantly. Urban environments, with their dense populations and diverse social dynamics, often serve as hotspots for various forms of criminal activity. In such contexts, timely intervention and effective policing are essential—not only to respond to incidents but to prevent them from occurring in the first place.

Traditionally, crime analysis has relied on manual methods such as reviewing police reports, analyzing eyewitness accounts, and conducting field surveillance. While these approaches have their merits, they are often reactive in nature, time-consuming, and susceptible to human error or bias. The need for more efficient and proactive crime-fighting strategies has led to the growing interest in technological solutions that can support law enforcement in identifying patterns, predicting incidents, and making informed decisions.

One of the most transformative developments in recent years is the widespread deployment of surveillance infrastructure. CCTV cameras, traffic monitoring systems, drones, and other sensor-based technologies are now commonplace in urban spaces. These tools collectively generate vast volumes of visual and contextual data, yet much of it remains underutilized due to the limitations of manual monitoring. Human operators can only observe a limited number of screens at a time, making it difficult to detect crimes in real time or notice subtle patterns that develop over days or weeks.

This is where intelligent video analytics and automated crime detection systems come into play. By applying advanced computer vision techniques, including deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), it is now possible to automatically process video feeds, identify anomalies, and alert authorities in real-time. These systems can detect unusual behaviors, recognize suspicious movement patterns, and even classify specific types of incidents—offering a level of precision and scalability that manual surveillance simply cannot match.

In addition to enhancing detection capabilities, the integration of crime trend analysis tools provides law enforcement with a strategic advantage. By identifying temporal and

spatial patterns in crime data, agencies can proactively target high-risk areas, allocate patrols more efficiently, and develop long-term prevention strategies. This supports not only improved operational effectiveness but also better use of limited public resources—a critical factor in cities facing budget constraints and growing security demands.

Furthermore, the integration of automated reporting and data management systems reinforces accountability and transparency, which are essential components of ethical policing. Structured digital records, complete with timestamps, locations, and visual evidence, reduce the risk of manipulation or oversight and ensure that all incidents are documented in a consistent and verifiable manner. This builds public trust, supports fair judicial processes, and promotes an open dialogue between communities and law enforcement agencies.

In conclusion, the background of this project is grounded in the evolving needs of modern policing and the immense potential offered by emerging technologies. By combining video surveillance, machine learning, and geospatial analysis, crime trend analysis systems represent a significant leap forward in how crime is monitored, prevented, and addressed in today's data-driven world.

1.2 MOTIVATIONS

Crime prevention is a fundamental objective for communities around the world, forming the foundation for public safety, economic development, and social well-being. As urban populations continue to grow and cities become more complex, the challenges faced by law enforcement agencies have intensified. Traditional policing methods, while still important, are no longer sufficient on their own to address the evolving nature of modern crime, which is often dynamic, opportunistic, and increasingly difficult to predict. In this context, the ability to proactively identify crime trends and patterns becomes not just beneficial, but essential. By leveraging data-driven insights, law enforcement can implement more targeted policing strategies that focus resources where they are most needed, ultimately leading to better crime deterrence and faster response times.

At the same time, the exponential growth of surveillance infrastructure—including public CCTV networks, private security cameras, traffic monitoring systems, and smart city sensors—has generated vast amounts of visual data. However, much of this data remains underutilized due to the limitations of manual monitoring, which is time-consuming, resource-intensive, and prone to human error. The emergence of intelligent video analytics, powered by advancements in machine learning and computer vision, presents an unprecedented opportunity to transform passive surveillance into an active, real-time tool for crime detection. Automated systems can continuously scan video feeds for suspicious behaviors, recognize patterns, and generate alerts without the need

for constant human supervision—significantly improving both the efficiency and responsiveness of public safety operations.

Another key motivation lies in the area of resource optimization. Many law enforcement agencies operate under tight budgets and limited staffing, making it crucial to maximize the impact of available personnel and equipment. Intelligent crime analysis systems can help prioritize patrol assignments, monitor high-risk areas more effectively, and reduce unnecessary manual labor. This results in better coverage of critical zones, faster intervention during incidents, and an overall improvement in operational efficiency.

Furthermore, in an era where public trust in law enforcement is closely tied to notions of transparency and fairness, technology can play a vital role in strengthening accountability. Automated reporting systems ensure that every incident—whether detected through surveillance or reported manually—is documented in a consistent, structured, and impartial manner. By including key details such as timestamps, locations, detected behavior, and visual evidence, these systems support a fair and verifiable process for all stakeholders involved. This not only helps uphold justice but also builds public confidence in the integrity of law enforcement practices.

In summary, the motivation for developing a Crime Trend Analysis System is rooted in the need to enhance proactive crime prevention, efficiently leverage modern surveillance technologies, optimize limited resources, and promote fairness and transparency in public safety efforts. By integrating advanced analytics with real-time data processing and visualization, such a system has the potential to fundamentally improve how communities respond to crime in the digital age.

1.3 SCOPE OF THE PROJECT

The Crime Trend Analysis System is designed as a comprehensive and scalable solution to enhance urban safety through the integration of historical crime data and real-time video surveillance analytics. Addressing the increasing need for intelligent and proactive security infrastructure, especially in rapidly urbanizing environments, the system offers a wide spectrum of capabilities—including data acquisition, pattern recognition, real-time anomaly detection, automated alerting, and incident reporting. These features collectively empower law enforcement agencies to make faster, smarter, and more informed decisions.

Central to the system is the fusion of past crime records and live CCTV footage, enabling both retrospective trend analysis and immediate identification of suspicious activities. The system can be extended to incorporate geospatial clustering techniques to identify evolving crime hotspots—geographical zones where criminal activity is

statistically more likely to occur. These clusters, once dynamically updated, assist authorities in allocating resources more efficiently and proactively responding to shifts in criminal behavior patterns.

In terms of real-time video analytics, the system leverages deep learning models—specifically a combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks—to detect anomalous behavior from live or recorded surveillance feeds. The CNN (ResNet50) extracts high-level visual features from each frame, while the LSTM processes temporal sequences to recognize patterns of behavior indicative of criminal activities such as burglary, vandalism, or assault. This significantly reduces reliance on continuous human monitoring and enhances both detection speed and accuracy.

To facilitate immediate operational response, the system integrates a confidence-based alert mechanism. When a suspicious activity is detected with high certainty (e.g., confidence > 0.8), the system triggers a real-time notification to designated law enforcement personnel—either via SMS or other communication channels—containing critical context such as the incident type, timestamp, and an image snapshot from the video feed.

The system also includes automated incident logging and structured report generation. Each event is captured along with relevant metadata—such as location, type of incident, model prediction confidence, and supporting visual evidence—and stored in a searchable format. These logs can serve legal, operational, and analytical purposes. In compliance with regional privacy standards, the system supports automated data retention policies, allowing footage and logs to be securely archived or deleted based on configurable lifespans, thereby ensuring ethical data handling.

From a deployment perspective, the platform is built with scalability and modularity in mind. It can operate effectively in both small-scale settings—such as gated communities or institutional campuses—and larger city-wide deployments. Its architecture supports integration with existing CCTV networks, emergency response systems, and public safety dashboards. A role-based access control system and intuitive user interface ensure secure, tiered access for stakeholders ranging from surveillance operators to policy-makers.

In summary, this project extends far beyond conventional crime mapping by delivering a real-time, AI-powered framework for intelligent surveillance, predictive analysis, and rapid emergency response. Through its emphasis on automation, actionable insights, and data governance, the Crime Trend Analysis System represents a forward-thinking approach to building safer, smarter, and more responsive urban communities.

2. PROJECT DESCRIPTION AND GOALS

2.1 LITERATURE REVIEW

Deep learning has profoundly influenced surveillance-based crime detection over the past decade, bringing together high-performance recognition models with growing computational resources. Early research predominantly explored 2D CNNs for frame-by-frame classification tasks—detecting specific behaviors (e.g., fights, break-ins) in offline video datasets. Notably, Soheil and Kin-Choong’s “A CNN-RNN Combined Structure for Real-World Violence Detection in Surveillance Cameras” illustrated how CNNs, when coupled with RNN architectures (like ConvLSTM), excel at capturing spatiotemporal cues indicative of violence or aggression. Their demonstrations underlined the power of CNNs in extracting visual features from individual frames, while RNNs map these features over time, yielding a more context-aware anomaly detector.

In parallel, real-time detection became a topic of keen interest, as evidenced by Kowshik and Shoeb’s “Real Time Crime Detection using Deep Learning.” Their approach demonstrates how low-latency pipelines—often employing lighter CNN backbones or specialized object detectors (e.g., YOLO, MobileNet)—can quickly spot suspicious objects like weapons or track unusual human movements. Although this offers immediate alerts, many such systems do not incorporate historical crime analysis; instead, they rely on direct visual cues in current footage.

Meanwhile, Charuni’s research (“Using CNNs, RNNs and Machine Learning Algorithms for Real-time Crime Prediction”) underscores the potential synergy between machine learning on historical data and deep learning for video analysis. Their solution uses numeric crime data to forecast potential spikes in criminal activity, complemented by a CNN+RNN pipeline that identifies anomalies in newly streamed video. However, the framework remains somewhat fragmented, lacking an automated process for data retention or robust geospatial clustering to guide strategic patrols.

Recent works, such as “Crime Prediction and Analysis using CNN & RNN” by S Kumar, Srinidheesh M, push deeper into hybrid forecasting—combining LSTM-driven time-series analysis with CNN-based feature extraction, especially from text descriptions or location-based data. While this approach refines crime trend prediction, it tends to overlook the complexities of live camera feeds and the immediate demands of real-time alerting.

“Design of a Real-time Crime Monitoring System using Deep Learning Techniques” by M. Mukto, M. Hasan shifts attention back to practical, on-the-spot detection. Employing YOLOv5 for weapons, MobileNetv2 for violence, and LBPH for face recognition, it achieves a near real-time pipeline that can identify lethal objects, fights, or specific individuals. Although highly relevant to direct surveillance tasks, it omits geospatial analytics of historical crime incidents and lacks an explicit data life-cycle management module to govern how long footage is stored or when it should be purged.

Overall, these studies reveal a dual emphasis: (1) video anomaly detection using 2D CNNs and RNNs for near real-time insights, and (2) machine learning for historical crime trend analysis. Yet, few integrate both seamlessly, and even fewer incorporate the automatic data retention or structured report generation needed in real-world policing.

2.2 GAPS IDENTIFIED

Many existing approaches to crime detection and analysis concentrate on isolated elements of the broader problem, without offering an end-to-end pipeline that addresses every stage from continuous video ingestion to effective alert generation. Some systems excel at real-time anomaly detection in surveillance footage—focusing on conspicuous behaviors such as open displays of violence, weapon usage, or erratic movements—yet they overlook robust data management practices necessary for handling large-scale, multi-camera deployments. Because real-world environments typically involve thousands of camera streams and terabytes of daily footage, a lack of methodical data retention strategies or automated archiving can lead to storage overload, regulatory non-compliance, or haphazard disposal of crucial evidence. Conversely, there are systems that rely on extensive post-processing of recorded footage but fail to provide immediate alerts, thereby missing the opportunity to intervene quickly in active or emerging criminal scenarios.

This disconnect between real-time detection capabilities and the practical requirements of high-volume data pipelines results in a fragmented experience for law enforcement agencies. Even if a given platform identifies suspicious activity in a laboratory setting, it may struggle to sustain the same performance across multiple, concurrent camera feeds, especially when each feed demands near real-time analysis. The absence of standardized lifecycle management for video content further exacerbates these issues, since privacy regulations often dictate strict limits on how and for how long footage may be stored. Without an automated mechanism to purge outdated or irrelevant video, agencies risk either violating data protection laws or incurring exorbitant storage costs.

Moreover, CNN-based feature extraction and RNN modeling have been shown to be powerful methods for identifying criminal behaviors across sequences of frames, but many research prototypes do not fully account for the complexities of diverse viewing conditions such as extreme variations in lighting, camera angles, partial occlusions, or crowded scenes. While some projects successfully spot “headline” offenses like weapon brandishing or overt violence, they often fail to capture more subtle indicators of wrongdoing—such as repeated loitering, furtive glances, or nuanced changes in posture that may precede aggressive acts. These deficits underscore the need for an integrated solution that unifies real-time detection across a wide range of suspicious activities with scalable data management procedures, covering everything from multi-channel video ingestion to incident notification and eventually automated data disposal. In meeting this challenge, future systems must strike a balance between operational feasibility, regulatory compliance, and detection accuracy, ensuring that they can effectively handle dense urban environments and diverse criminal typologies without succumbing to storage limitations or blind spots in detection.

2.3 OBJECTIVES

This project aspires to develop a comprehensive, real-time crime detection and alert system that fuses advanced computer vision, deep learning, and automated notification processes into a cohesive, end-to-end pipeline. By harnessing frame-level feature extraction from both surveillance and recorded video streams via a pre-trained CNN (ResNet50), the proposed solution systematically converts raw image data into high-dimensional vector representations, thus capturing the fine-grained visual cues integral to identifying criminal activities. This form of structured representation is critical for preserving subtle patterns—ranging from suspicious postures, abrupt movements, or the presence of illicit objects—that may otherwise go unnoticed by manual or simpler automated monitoring methods. Consequently, the reliability of subsequent classification stages is enhanced, as these feature vectors furnish a robust descriptor of any given scene.

Building on these CNN-extracted features, the project then incorporates an LSTM-based classifier designed to interpret temporal dependencies between consecutive frames. This sequential model design is particularly relevant for crime detection, where illicit activity often evolves over a short span of time rather than appearing fully formed in a single frame. By analyzing the interplay of motion patterns, object interactions, and spatial context across sequential image samples, the LSTM can precisely differentiate between incidents such as robbery, burglary, assault, vandalism, and abuse, thereby providing a more holistic view of ongoing behavior within any camera’s field of view. Throughout development, the system explores multiple optimization techniques—including data normalization, balanced sampling, and hyperparameter tuning—to

maximize classification accuracy and mitigate common pitfalls like overfitting or skewed class distributions.

Complementing this detection mechanism is an automated alert pipeline, which sits at the core of the project’s commitment to timely intervention. Whenever the model’s confidence score exceeds a predefined threshold, the system issues immediate notifications—such as SMS or platform-specific alerts—to designated law enforcement contacts. This real-time communication loop drastically reduces reliance on constant human surveillance, ensuring that officers are quickly informed of potentially dangerous or illegal behavior. Beyond easing the burden on monitoring personnel, this approach directly serves the project’s broader aim of strengthening public safety by shortening response times and directing enforcement resources more efficiently. Taken together, the overarching pipeline—from feature extraction and temporal classification through to immediate alerts—illustrates how deep learning methodologies can be harnessed in real-world scenarios to deter crime, enhance situational awareness, and ultimately safeguard communities.

2.4 PROBLEM STATEMENT

Traditional surveillance strategies often rely on continuous manual monitoring or a reliance on post-event video review, creating substantial delays in detecting and responding to ongoing criminal activities. In large-scale deployments, operators can be overwhelmed by the sheer volume of camera feeds, rendering it virtually impossible to watch every stream vigilantly at all times. As a result, emergent threats—including subtle precursors such as small altercations, furtive object exchanges, or shifts in group dynamics—may pass unnoticed or unaddressed until they escalate beyond immediate control. Even when automated solutions are introduced, many focus on analyzing individual frames or simplistic, rule-based indicators, ultimately failing to capture the broader temporal patterns that differentiate genuine criminal behavior from benign movements in crowded or dynamic settings. Consequently, law enforcement agencies are unable to respond in real time, leaving them to play catch-up after a serious incident has already occurred.

Further exacerbating this challenge is the absence of timely notification pipelines in most existing systems. While a handful of tools may flag suspicious activities, they often lack mechanisms to instantly reach relevant authorities with concise, actionable alerts. Human personnel must still sift through logs or footage, wasting valuable response time. Additionally, many setups do not integrate well with modern data handling protocols, resulting in unclear or inconsistent storage policies that either accumulate vast amounts of footage indefinitely—thereby raising privacy concerns and straining resources—or purge data prematurely, risking the loss of important evidence. In this fragmented environment, the imperative is clear: a robust, end-to-end crime

detection pipeline capable of intelligently modeling the evolving nature of criminal scenarios, discerning subtle cues from diverse camera streams, and automatically alerting law enforcement in near real-time. Such a solution must also respect data retention and privacy guidelines, ensuring that only necessary footage is kept, but retained long enough to support forensic investigation when required. By shifting the paradigm from reactive video analysis to proactive, continuous monitoring with immediate notifications, this project addresses a critical need for improved public safety and efficient resource utilization in policing.

2.5 PROJECT PLAN

Implementing this real-time crime detection system involves several key phases, beginning with data collection and preparation. The project will gather a diverse range of surveillance videos—encompassing varied lighting conditions, camera angles, and scene complexities—to train and test both the CNN-based feature extractor and the LSTM-based sequential classifier. During this initial phase, data preprocessing steps, such as frame extraction, resolution standardization, and class labeling, will be performed, ensuring the model is exposed to representative samples of both typical and suspicious scenarios.

Once the dataset is established, attention shifts to model development, where the pre-trained ResNet50 architecture is refined for frame-level feature extraction and subsequently integrated with an LSTM module. Here, experiments will be conducted on hyperparameter tuning, including batch size, learning rates, and the number of LSTM units, in order to optimize performance across various crime-related classes like robbery, assault, and vandalism. Strategies for improving model robustness—such as balanced sampling, data augmentation, and early stopping—will also be explored to mitigate overfitting or class imbalance issues.

Subsequent to core model training, the project will implement the automated alert pipeline, which serves as the interface between the detection model and law enforcement channels. Thresholds for detection confidence will be fine-tuned to balance prompt alerts against the risk of false positives; once the model identifies suspicious behavior surpassing these thresholds, an immediate notification (e.g., via SMS or an internal communications platform) will be dispatched to designated personnel. Complementary log-keeping and reporting tools will be developed to organize event data, timestamped frames, and relevant metadata, thereby providing investigators with a concise evidence package for potential follow-up.

In parallel, the team will design and enforce data retention policies to ensure compliance with privacy regulations and practical storage limits. Automated scripts will be configured to archive or delete older footage according to predefined schedules,

while preserving incident-related clips for a sufficiently long period to support forensic investigations. Thorough system testing will be performed under simulated multi-stream conditions to verify that the model maintains low-latency inference even when multiple camera feeds are ingested simultaneously.

The final deployment phase will see the model and alert mechanism integrated into a production-like environment, either on-premises (using dedicated servers and networking equipment) or in the cloud (leveraging containerized microservices). Performance metrics, such as detection speed, alert accuracy, and resource utilization, will be monitored continuously to validate that the system meets real-time operational requirements. Post-deployment, iterative refinements will address user feedback from law enforcement agencies, with potential expansions to handle additional behavior classes or incorporate advanced visualization dashboards. Through this phased plan—ranging from data acquisition and model development to automated alerts and lifecycle management—the project aims to deliver a scalable, user-friendly crime detection platform capable of generating timely interventions and better public safety outcomes.

Chapter 3

TECHNICAL SPECIFICATION

3.1 REQUIREMENTS

This section outlines the requirements necessary for the successful development and operation of the Crime Trend Analysis system. The requirements are categorized into functional and non-functional aspects. Functional requirements describe the core features and behavior of the system, while non-functional requirements define the quality attributes, performance, and constraints under which the system must operate.

3.1.1 FUNTIONAL REQUIREMENTS

The functional requirements of the Crime Trend Analysis System outline the essential tasks and features necessary for its effective operation:

1. **Crime Hotspot Detection:** The system must analyze historical crime data and apply geospatial clustering techniques to identify and visualize high-crime areas on an interactive map.
2. **Anomaly Detection in Surveillance Feeds:** The system should process live CCTV footage using a 2D CNN + RNN model to detect suspicious activities such as violent behavior, unauthorized access, or unusual movement patterns.
3. **Real-Time Alerting:** When an anomaly is detected in a video feed, the system must immediately generate an alert, notifying law enforcement personnel through a dashboard or mobile notification.
4. **Automated Incident Reporting:** The system must generate structured reports containing relevant details such as the crime type, timestamp, location, and images extracted from the video, ensuring accurate documentation of incidents.
5. **Data Retention and Management:** The system should implement an automated policy for storing, archiving, or deleting video footage and crime records based on predefined retention periods to maintain compliance with privacy laws.

6. User Interface and Dashboard: A web-based or desktop dashboard should be developed, allowing authorized personnel to visualize crime hotspots, monitor live video feeds, receive alerts, and access historical reports.

7. Secure Access and Authentication: The system must enforce role-based access control (RBAC), ensuring that only authorized law enforcement or administrative personnel can access crime records, reports, and video feeds.

3.1.2 NON - FUNCTIONAL REQUIREMENTS

The non-functional requirements define the performance criteria and operational standards for the Crime Trend Analysis System:

1. Performance: The system must achieve high accuracy in anomaly detection with low latency, ensuring that crime hotspots are updated periodically and video-based alerts are generated in real-time.

2. Reliability: The system should operate continuously with minimal downtime, ensuring 24/7 monitoring and seamless anomaly detection in multiple surveillance feeds.

3. Scalability: The architecture must support multiple video streams and large-scale crime databases, allowing expansion from small neighborhood deployments to city-wide implementations.

4. Usability: The user interface must be intuitive and accessible to law enforcement officers with minimal technical training, providing simple navigation for hotspot visualization, alert monitoring, and report generation.

5. Maintainability: The system should be designed in a modular fashion, allowing for easy updates, integration with new machine learning models, and maintenance of database structures without disrupting existing functionality.

6. Compliance: The system must adhere to legal and regulatory requirements regarding data privacy, including GDPR, law enforcement data protection policies, and ethical AI guidelines for surveillance-based systems.

7. Efficiency: Optimizations should be implemented to minimize computational overhead, ensuring that the 2D CNN + RNN pipeline for anomaly detection can run efficiently on standard hardware without requiring high-end GPUs

3.2 FEASIBILITY STUDY

Determining whether this real-time crime detection and alert system can be successfully implemented in real-world environments involves scrutinizing its technical, economic, and social dimensions. The project’s overarching objective—achieving automated, near real-time recognition of suspicious activities combined with instant notifications—demands a robust infrastructure capable of continuous video ingestion, low-latency inference, and secure, compliant data management. The following subsections provide a more detailed account of how these facets of feasibility intertwine to support the system’s viability.

3.2.1 TECHNICAL FEASIBILITY

1. **Proven Deep Learning Techniques:** Utilizes a pre-trained ResNet50 for frame-level feature extraction, leveraging a CNN architecture known for stable performance on complex, high-resolution imagery. Adopts an LSTM (Long Short-Term Memory) module to handle temporal data, a method extensively validated in video classification scenarios where movement patterns and object interactions define the nature of suspicious events. Employs techniques like data augmentation, balanced sampling, and hyperparameter tuning to account for varying camera angles, lighting conditions, and partial occlusions, ensuring the system adapts to real-world variances.
2. **Infrastructure for Real-Time Throughput:** Requires GPU acceleration (e.g., NVIDIA RTX, Tesla series, or AMD equivalents) to maintain near real-time inference, particularly when multiple camera streams operate simultaneously. Implements libraries such as OpenCV or FFmpeg for buffering and frame processing, crucial for avoiding I/O bottlenecks that could delay detection. Considers load distribution across multiple GPUs or cloud-based GPU instances so that video data can be processed in parallel, preserving low latency even during peak camera activity.
3. **Parallelism and Scalability:** Employs microservices or containerization (Docker, Kubernetes) to distribute core tasks (frame extraction, model inference, alert dispatch), thus enabling multiple inference engines to run concurrently. Incorporates dynamic autoscaling policies, where new containers or GPU resources can spin up as the number of incoming video streams rises, ensuring consistent performance rather than a degraded experience under higher workloads.
4. **Alert Integration:** Integrates a robust messaging or notification component (SMS gateways, push notifications, or internal dispatch systems) to instantly inform relevant authorities once a suspicious event is flagged by the LSTM classifier. Maintains centralized log files capturing essential metadata—timestamps, bounding box

coordinates, and model confidence scores—for every alert, supporting forensic reviews and iterative model improvements.

5. **Technical Readiness:** Relies on well-supported frameworks (PyTorch, TensorFlow) that boast large community ecosystems, making debugging and feature enhancements more accessible. Implements fallback CPU inference modes to provide partial functionality in the event of GPU unavailability, reducing the risk of total system downtime. Emphasizes modular architecture, where components (ingestion, inference, alerting) can be updated or replaced with minimal disruption to the overall workflow.

3.2.2 ECONOMIC FEASIBILITY

1. **Initial Capital Investment:** A central consideration for this system is the expense required for GPU-accelerated computing resources, whether acquired as on-premises servers or leased through a cloud provider. Especially for environments demanding multiple high-resolution video streams, hardware costs can climb as the number of GPUs or specialized camera units increases. Moreover, peripheral elements such as networking equipment, robust storage solutions, and cooling infrastructure (for data center or server room conditions) must be factored into the total budget. These initial investments, while significant, lay the groundwork for a scalable surveillance model that, once operational, can handle multiple feeds without drastic future upgrades.

2. **Operational Cost Savings:** Once the system is running, agencies can achieve reduced labor expenses by automating extensive segments of live surveillance monitoring. Fewer staff hours are required to examine real-time camera feeds, allowing a small team to oversee a wide coverage area. Coupled with the project's capability for rapid alerts, agencies can potentially avoid costly consequences of delayed responses—such as extensive property damage or prolonged investigations—and thereby preserve resources for other facets of law enforcement. Over an extended timeline, the savings in personnel hours and associated administrative overhead can substantially improve the return on investment for the hardware and software components.

3. **Risk Mitigation via Pilot Deployment:** In many cases, a staged approach limits both financial risk and organizational disruption. By implementing the system on a smaller scale—perhaps covering a single high-priority zone—stakeholders gather data on average inference latencies, detection accuracy, and false alarm ratios. These metrics guide realistic cost estimates for broader deployment, helping decision-makers identify precisely where additional GPU nodes or more sophisticated cameras are warranted. If pilot results confirm that the automated system consistently intercepts or deters criminal activity, subsequent expansions can be more strategically budgeted, ensuring the solution's cost remains in alignment with tangible benefits.

4. **Use of Open-Source Tools:** By leveraging popular and freely available machine learning frameworks—such as PyTorch or TensorFlow—and container technologies like Docker, agencies minimize ongoing licensing costs. Open-source libraries for image processing (e.g., OpenCV, FFmpeg) further reduce software expenses, as do widely used container orchestration platforms that enable load balancing without vendor lock-in. With these open-source ecosystems, agencies can direct more funds toward hardware advancements or additional functionalities, rather than continually paying subscription or per-seat fees. The active community support associated with open-source projects also facilitates quick resolutions for technical challenges, cutting down on external consulting fees.

5. **Potential Community and Business Benefits:** A proven reduction in crime rates can foster a more secure environment that, over time, translates into commercial growth and an uptick in local investment. Businesses and homeowners feel more comfortable operating or residing in areas covered by automated, real-time surveillance, potentially driving up property values and tax revenues. Insurance carriers may even consider offering reduced premiums to establishments protected by an effective monitoring and alert system. Furthermore, success in the initial deployment can yield financial or strategic incentives, such as government grants or technology innovation awards, effectively subsidizing further scaling or enhancements. From a broader economic perspective, a well-functioning, real-time detection solution can catalyze a virtuous cycle of safety, investment, and community confidence—bolstering overall economic feasibility.

3.2.3 SOCIAL FEASIBILITY

1. **Privacy and Ethical Considerations:** Raises fundamental questions about how much video data the system retains, who can review that footage, and under what circumstances. Involves creating well-defined data retention schedules (e.g., 30, 60, or 90 days) to dispose of archived video periodically, thus striking a balance between investigative needs and public privacy expectations. Advises compliance with local and international data protection regulations (such as GDPR or state-level privacy laws) to minimize unauthorized or indefinite storing of personal information. Recommends implementing an audit trail detailing every access attempt to the system or data, enabling accountability in case footage is misused or accessed outside authorized channels.

2. **Impact on Community Trust:** Can enhance perceived safety by delivering faster responses to crimes in progress, thereby reassuring residents, business owners, and visitors that law enforcement is both vigilant and well-equipped. Runs the risk of pushback if individuals feel their daily activities are excessively monitored or if facial data or personal identifiers are stored without consent. Suggests public demonstrations, town-hall meetings, or pilot rollouts where the system's effectiveness is showcased,

building confidence through transparency and demonstrable results. Integrates reporting mechanisms—such as monthly or quarterly public summaries—showing how many incidents were detected, how quickly authorities responded, and which data was retained.

3. Stakeholder Engagement: Encourages early collaboration with local authorities, neighborhood associations, privacy watchdog groups, and human rights organizations. Recommends forming oversight committees—such as an independent review board—to periodically audit alert logs, data retention practices, and any expansions to the system’s functionality. Incorporates user feedback loops, where the public can raise concerns about civil liberties or data handling, and where law enforcement can clarify operational guidelines. Provides channels (hotlines, websites) for citizens to report misuse or file complaints, ensuring that the technology maintains public accountability and public interest is safeguarded.

4. Cultural and Regional Variations: Recognizes that community tolerance for surveillance can differ significantly by locale; some urban populations may welcome tighter security, while smaller or rural communities might find it intrusive. Suggests customizing feature sets—like certain object-detection modules or data retention durations—to align with local norms, moral standards, and legal frameworks. Acknowledges that language barriers and differing digital infrastructure across regions necessitate localized user interfaces, signage, and documentation to promote proper system usage and acceptance. Advises legal counsel or local governance liaisons to verify that any camera placement, data capture, and alert processes do not conflict with cultural sensitivities or statutory limitations.

5. Demonstrable Public Safety Advantages: Demonstrates a direct impact on public well-being by thwarting emergent crimes or minimizing damage and injuries through swift responses. Reinforces a sense of shared responsibility between law enforcement and citizens; when used successfully, the system can show that communities benefit from providing data that is handled responsibly and used solely for crime deterrence. Encourages ongoing evaluations, such as before-and-after analyses of crime rates or average police response times, to present tangible evidence of improvements. Cultivates an environment where the platform itself gains positive reputation for its ability to promote safer neighborhoods—potentially leading to endorsements, grants, or philanthropic support for further refinement and scale-up.

3.3 SYSTEM SPECIFICATION

In order to implement a real-time crime detection and alert system capable of handling multiple video streams and performing near-instant inference, specific attention must be given to both the hardware and software environments. On the hardware side, the system must offer sufficient computational power to accommodate the deep learning

models—particularly a CNN for frame-level feature extraction and an LSTM for sequential classification—while still ensuring low latency. The software stack is equally critical, as it provides the necessary frameworks, libraries, and supporting services for robust data flow, model deployment, and automated notification workflows. By aligning hardware capabilities with an appropriate software infrastructure, the overall platform can maintain high performance, scalability, and reliability as it processes extensive video data in real time.

3.3.1 HARDWARE SPECIFICATION

1. **GPU-Centric Compute Nodes:** The cornerstone of the system’s real-time detection capability hinges on one or more dedicated GPUs. At a minimum, an NVIDIA RTX 3060 or AMD equivalent is recommended to handle inference for the CNN (ResNet50) and LSTM simultaneously. Larger deployments or higher camera resolutions may warrant multiple, high-end GPUs (e.g., RTX 3080 or 4090) or multi-GPU servers. In environments where rapid scalability is crucial, cloud-based GPU instances (AWS, Azure, GCP) allow for elastic allocation of compute power, avoiding physical hardware investments that might remain underused during off-peak hours.

2. **CPU and Memory:** A multi-core CPU (e.g., Intel Core i7/i9 or AMD Ryzen 7/9) with at least 16–32 GB of RAM underpins the ingestion, buffering, and scheduling tasks required to streamline video data into the GPU pipeline. Ample CPU threads help orchestrate concurrent processes such as frame extraction, I/O operations, and real-time logging without introducing latency. A well-matched CPU–GPU ratio ensures that the system remains balanced; an underpowered CPU can create a bottleneck, despite having high-end GPUs

3. **Storage and I/O Speed:** Solid-state drives (SSDs) are strongly suggested for hosting model weights, log files, and any short-term video buffer. The faster read/write speeds of SSDs—compared to mechanical HDDs—are crucial when high volumes of frames must be processed or archived in short order. Moreover, if the system must store historical clips for forensics or continuous improvement (e.g., re-training on fresh data), considering a tiered architecture can be beneficial, placing “hot” data on SSDs and archiving older footage on cost-effective HDD-based storage or network-attached storage (NAS).

4. **Network Infrastructure:** To ensure minimal latency between cameras, GPU nodes, and end-user dashboards, robust internal networking—often in the form of gigabit Ethernet or fiber optic connections—becomes pivotal. In distributed environments that span multiple sites, employing Virtual Private Networks (VPNs) or dedicated WAN links can keep the video data secure while still allowing for near real-time ingestion into the compute cluster. Any bandwidth constriction at this layer may degrade the end-to-end responsiveness the system aims to achieve.

5. Edge vs. Centralized Deployments: Some deployments might benefit from placing low-power edge devices (e.g., NVIDIA Jetson series or specialized AI mini-PCs) next to cameras, performing partial inference or frame sampling on-site. This approach offloads bandwidth requirements, sending only preprocessed data or alerts to a central server. Conversely, a centralized model locates all processing in a single data center, leveraging dense GPU clusters for maximum throughput and simplified manageability. Both approaches have merits depending on the environment: edge solutions excel in remote or bandwidth-limited areas, whereas centralized clusters shine in well-connected regions requiring advanced analytics or integrated data management.

6. Scalability and Redundancy: Designing the hardware footprint from day one to allow for vertical or horizontal expansion is critical. As camera counts rise or resolution standards increase, new GPU servers or additional GPU cards can be added incrementally, thereby alleviating the need for a complete hardware overhaul. Additionally, redundancy features, like backup power (UPS), RAID-configured storage, or failover GPU nodes, prevent downtime in mission-critical deployments where even a brief interruption can compromise public safety or trust in the system. Minimizing single points of failure—through load balancing across multiple machines or replicated data clusters—ensures the platform remains robust and continuously accessible.

7. Environmental and Maintenance Considerations: GPUs and high-performance servers typically generate heat, demanding adequate cooling solutions—either dedicated AC units in on-premises data centers or well-monitored thermal thresholds in cloud instances. Regular maintenance schedules for dust cleaning, firmware updates, and hardware inspections (e.g., checking GPU fans or CPU thermal paste) maintain optimal performance levels. In high-humidity or extreme-temperature locales, appropriate climate control or ruggedized hardware may be necessary. Additionally, routine hardware checks ensure minimal risk of unplanned outages, thus preserving the system's continuous operational readiness.

3.3.2 SOFTWARE SPECIFICATION

1. Operating System and Dependencies: Prefers modern Linux distributions (e.g., Ubuntu 20.04+, CentOS, or Debian), which generally provide excellent compatibility with GPU drivers (CUDA, ROCm) and easy access to repositories for machine learning packages. Admits alternative solutions (Windows Server environments) if institutional policies dictate, although this typically necessitates extra configuration and licensing for GPU acceleration. Stresses the need to maintain a consistent OS environment—via thorough version control and updates—to ensure stable performance, especially given the complexity of GPU-accelerated workloads.

2. Machine Learning Frameworks: Employs PyTorch or TensorFlow to implement and serve the CNN (ResNet50) and LSTM modules, capitalizing on their GPU-accelerated builds and extensive community support. Integrates libraries such as OpenCV or FFmpeg for real-time frame decoding, pre-processing, and tasks like resizing or color transformations, thereby ensuring frames are properly conditioned for inference. Recommends data science utilities (NumPy, Pandas, scikit-learn) to facilitate advanced data manipulation, hyperparameter tuning, and post-processing steps (e.g., analyzing detection logs for false positives or performance metrics).

3. Containerization and Orchestration: Packages critical microservices (inference engine, alert dispatch, logs) in Docker containers for reproducible, environment-agnostic deployment. Suggests Kubernetes (K8s) or a similar orchestration platform if large-scale or multi-site deployments require dynamic load balancing, rolling updates, or robust failover procedures. Encourages tagging container images with explicit version numbers to track model iterations, simplifying rollback if a new build inadvertently decreases detection accuracy.

4. Alerting and Microservice Architecture: Structures the application into discrete services (e.g., detection service, notification service, data retention service), each with defined APIs for communication—often a REST or gRPC interface. Introduces a messaging queue (RabbitMQ, Kafka) to decouple the inference pipeline from the alert pipeline, preventing detection slowdowns during surge events. Recommends implementing built-in logging within each service, capturing event timestamps, model confidence thresholds, and user interventions, thereby facilitating transparent audits or forensics.

5. Logging, Monitoring, and Analytics: Emphasizes a comprehensive logging framework (Elasticsearch, Prometheus, or Graylog) to track GPU utilization, inference speeds, number of camera feeds, and frequency of alerts. Implements real-time dashboards (e.g., Grafana, Kibana) displaying key performance indicators (KPIs) such as detection latency, false-positive ratios, and average response times, enabling

operators to immediately identify system bottlenecks. Encourages the gathering of detection statistics (e.g., ratio of recognized threat types, confidence distributions) for continuous model improvement, an essential step in refining thresholds, re-balancing classes, or fine-tuning hyperparameters.

6. Data Retention and Compliance Tools: Incorporates features that automatically purge or archive older video segments and detection logs, adhering to privacy regulations (GDPR, local data protection laws) and internal organizational policies. Provides administrators with a policy configuration interface, specifying how long to keep raw frames, classified events, or specific incident footage deemed relevant for ongoing investigations. Logs the lifecycle of each file or detection record (creation, user access, eventual deletion), ensuring clarity in the event of audits or public scrutiny regarding how data is used or stored.

7. Security and Access Control: Relies on TLS/HTTPS encryption for communication between internal microservices, the detection service, and user interfaces, safeguarding any sensitive data in transit. Implements role-based access control (RBAC) or OAuth-based authentication to delineate user privileges, ensuring that only authorized personnel can modify detection thresholds or retention policies. Encourages adopting secure coding practices, routine vulnerability scans, and automated patching processes to mitigate risks in an environment where intrusion or tampering could have direct public safety implications.

DESIGN APPROACH AND DETAILS

4.1 SYSTEM ARCHITECTURE

The architecture of the Crime Trend Analysis System has been meticulously designed to meet the dual objectives of real-time crime detection and strategic crime trend analysis. It follows a modular, scalable, and fault-tolerant design pattern, ensuring that the system can operate efficiently in varied deployment contexts, ranging from localized surveillance environments to large-scale city-wide networks. The architecture brings together components from deep learning, computer vision, cloud-based communications, and data analytics into a unified framework capable of delivering both proactive and reactive public safety solutions.

The system is broadly divided into six key layers: Video Ingestion, Feature Extraction, Temporal Modeling and Classification, Decision Logic and Alerting, Logging and Reporting, and Deployment and Scalability. These layers work in tandem to ensure the end-to-end pipeline—from frame capture to real-time alert—is robust, low-latency, and operationally viable.

1. **Video Ingestion Layer:** This layer serves as the system's entry point, responsible for handling video data from live CCTV streams or pre-recorded surveillance footage. The OpenCV library is used to access and read video files or camera feeds frame by frame. A frame skipping mechanism is implemented (e.g., processing every 5th frame) to optimize resource consumption without compromising the detection of suspicious activities. The selected frames are forwarded for further processing and are temporarily buffered to ensure continuity and smooth data flow. This layer is designed to handle multiple input streams in parallel, making it suitable for multi-camera setups. The ingestion layer may also include preprocessing tasks such as frame resizing, normalization, and RGB color conversion to ensure consistency in input data before feature extraction. These preprocessing steps help reduce noise and variation across different video sources, particularly when dealing with varying camera resolutions, lighting conditions, or environmental noise.

2. **Feature Extraction Layer:** In this layer, the visual content of each frame is translated into a high-dimensional feature vector using a pre-trained ResNet50 Convolutional Neural Network (CNN). ResNet50 is chosen due to its proven effectiveness in capturing complex image features and its relatively efficient inference time. The final fully connected classification layer of ResNet50 is removed, and the global average pooling layer output is used, resulting in a 2048-dimensional feature vector per frame. Each frame is transformed via a standardized PyTorch image transform pipeline (which includes resizing and conversion to tensor format) before being fed into the CNN. This

design abstracts the low-level pixel data into semantically rich features that encapsulate objects, environments, and interactions visible in the frame. These frame-wise feature vectors are stored in sequence, forming a temporally ordered dataset that represents the visual evolution of the scene over time.

3. Temporal Modeling and Classification Layer: The feature vectors generated by the ResNet50 model are sequentially passed to a Long Short-Term Memory (LSTM) network, which is tasked with modeling temporal dependencies between frames. The LSTM processes these sequences to detect behavioral patterns characteristic of criminal activities such as burglary, assault, vandalism, robbery, or abuse. The sequential modeling capability of LSTM allows the system to detect contextually significant transitions in motion or posture that static classifiers would fail to capture. The LSTM outputs a class label indicating the predicted type of activity and a corresponding confidence score representing the system's certainty in its prediction. This design enables the system to make decisions not just on individual frame data but on the broader context provided by a sequence of observations, improving both accuracy and reliability in identifying suspicious behavior.

4. Decision Logic and Alerting Layer: Upon receiving the prediction and associated confidence score from the LSTM, the system evaluates whether the confidence exceeds a predefined threshold (e.g., 0.8). This threshold is chosen empirically to balance the trade-off between sensitivity (detecting all suspicious events) and specificity (avoiding false alarms). If the threshold is crossed, the system proceeds to generate a real-time alert. The alerting mechanism is built around a cloud communication API such as Twilio, which enables the system to send SMS messages or other forms of notifications to law enforcement officers or designated response teams. Each alert includes key details such as the predicted crime type, timestamp, and a visual snapshot from the video, ensuring that decision-makers receive concise and actionable intelligence. This layer is designed to be extensible—future implementations could include integration with emergency dispatch software, mobile apps, email alerts, or desktop notifications, depending on the operational requirements of the deployment environment.

5. Logging and Reporting Layer: For every incident detected, the system logs detailed records in a structured and queryable format. Each log entry captures metadata such as:

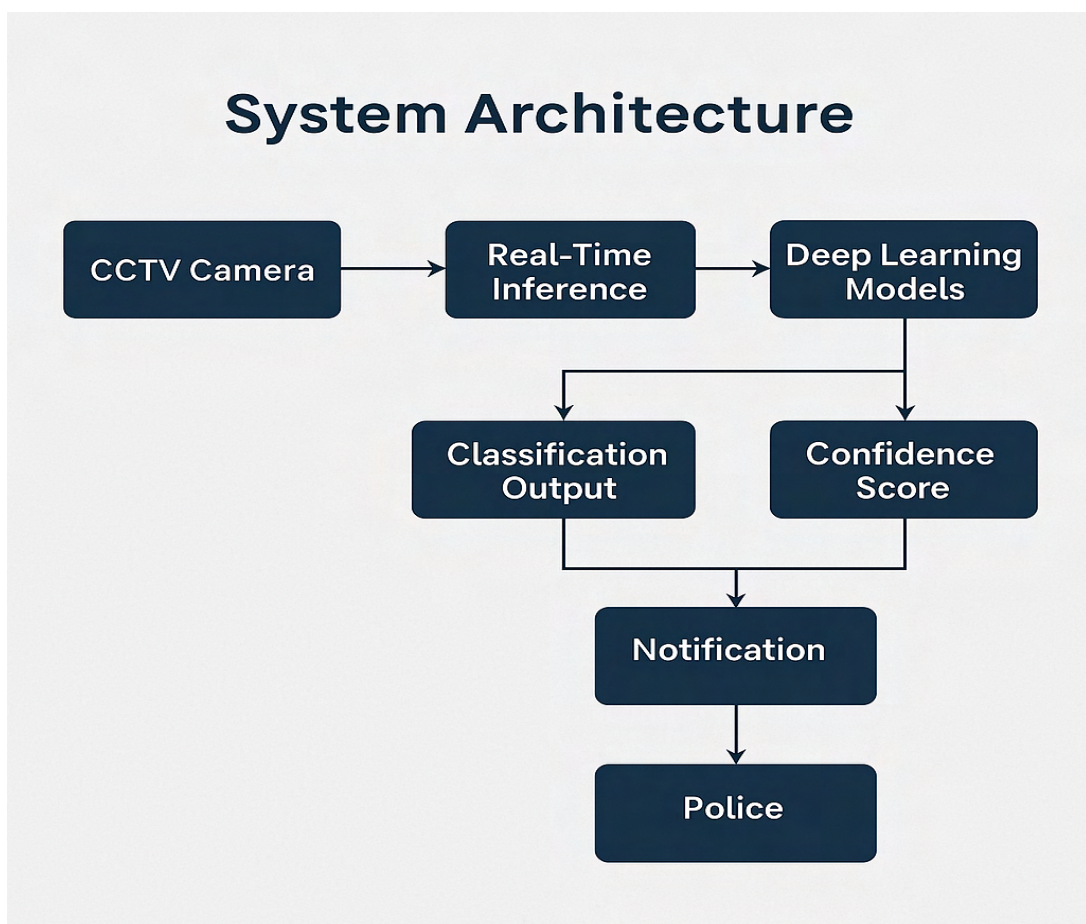
- Date and time of detection
- Camera/source ID
- Predicted activity class
- Confidence score

- Path to associated image or video snippet

These logs serve multiple purposes:

- Audit Trails: For reviewing historical system performance and response accuracy.
- Legal Documentation: Supporting evidence for legal investigations or court proceedings.
- Pattern Analysis: Aggregated data can be used to train improved models or identify long-term crime trends.

This layer also includes a data retention policy engine, which enforces automatic archival or deletion of logs and media content based on organizational or regional data privacy laws. This ensures compliance with frameworks such as GDPR or the Indian Personal Data Protection Bill (DPDP), reducing the risk of data misuse or oversharing.



4.1.1 WORKFLOW

The workflow of the Crime Trend Analysis System is structured as a seamless and systematic pipeline, beginning with the acquisition of video footage and culminating in the real-time detection, classification, alerting, and logging of potential criminal activities. The entire process is engineered to operate autonomously, with minimal human intervention, while ensuring high responsiveness and accuracy in dynamic urban environments. At the foundation of this pipeline lies the video ingestion layer, which initiates the workflow by accessing either live surveillance streams from CCTV cameras or pre-recorded video files stored on disk. The OpenCV library is employed to continuously capture video frames in real time. To balance detection sensitivity with computational efficiency, the system incorporates a frame-skipping strategy, commonly configured to extract every 5th frame. This selective sampling ensures that key motion and behavior cues are preserved while reducing the volume of data that must be processed at each time step.

Each extracted frame undergoes a preprocessing stage, wherein it is resized to a uniform resolution (typically 224x224 pixels) and converted into an RGB format to maintain consistency across varying camera sources. These preprocessed frames are then passed to the feature extraction layer, where a pre-trained ResNet50 Convolutional Neural Network (CNN) is used to generate high-level feature representations. This model, chosen for its proven robustness in object and scene recognition tasks, transforms each frame into a 2048-dimensional feature vector. These feature vectors capture critical spatial characteristics such as the presence of objects, human postures, environmental details, and other visual semantics that may be indicative of a criminal event. Rather than processing each frame independently, the system retains the temporal sequence of feature vectors, preserving the chronological structure of the visual narrative.

These ordered feature vectors are then handed off to the temporal modeling layer, where an LSTM (Long Short-Term Memory) network is responsible for analyzing the sequence as a whole. The LSTM is specifically designed to capture temporal dependencies and long-range contextual patterns within sequential data. In this system, it plays a crucial role in identifying behavioral trends that unfold over time, such as a person moving suspiciously near a restricted area, a sudden escalation in motion suggestive of violence, or loitering patterns that could indicate intent to commit vandalism or theft. The LSTM processes the input sequence and outputs a classification result, indicating the predicted type of crime or abnormal behavior, along with a corresponding confidence score. This score, typically a value between 0 and 1, quantifies the system's certainty regarding its prediction and serves as a control parameter in the subsequent decision-making phase.

Upon receiving the classification label and confidence score, the system transitions to the decision logic and alerting module. Here, a threshold-based mechanism evaluates whether the model's confidence surpasses a predefined limit, which is usually set at 0.8. This threshold serves as a safeguard against false positives, ensuring that alerts are generated only when the system is sufficiently certain that a suspicious or criminal activity has occurred. If the threshold condition is met, the system immediately triggers a real-time alert. The alerting process is facilitated through an integrated notification API, such as Twilio, which sends a text message or push notification to a designated contact—such as a law enforcement officer, security personnel, or emergency dispatcher. The notification contains all essential contextual information, including the type of detected incident, the exact timestamp, and a snapshot image captured from the video at the moment the activity was detected. This instant communication enables authorities to respond rapidly and appropriately, potentially intervening before the crime escalates.

In parallel with the alerting mechanism, the system initiates the event logging and reporting process, ensuring that every detection instance is thoroughly documented. The logging module records structured metadata, such as the classification label, time and date, confidence score, and the source of the video feed (e.g., camera ID or file path). Additionally, it stores references to the corresponding frame images or short video clips that visually support the detection. This log not only serves as a historical record for auditing and legal purposes but also supports advanced analytics for identifying crime trends over time. Furthermore, the system incorporates a data retention policy engine, which automatically manages the lifecycle of stored logs and media content. Based on pre-defined rules or legal mandates, data can be archived for long-term storage or securely deleted to maintain compliance with privacy regulations such as GDPR or the Indian DPDP.

The entire workflow is designed with modularity and scalability at its core. Each layer—from video ingestion to alerting—can be independently scaled or enhanced based on deployment needs. For instance, the system can operate on a single local GPU-powered machine for small-scale surveillance or be extended across a distributed network of cameras and servers in a metropolitan environment. This flexibility ensures that the system can adapt to a wide range of operational contexts, whether it be monitoring school campuses, commercial complexes, residential neighborhoods, or large urban zones. The architecture also supports potential future enhancements, such as multi-camera coordination, predictive hotspot mapping, and integration with public safety dashboards.

In summary, the workflow of the Crime Trend Analysis System represents a highly coordinated and intelligent process that begins with raw video input and culminates in actionable real-time crime detection. Through the integration of state-of-the-art deep learning models, real-time decision-making algorithms, and automated communication tools, the system exemplifies a forward-thinking approach to urban safety and law

enforcement support. Its ability to operate autonomously, process temporal behavior patterns, and communicate critical incidents in real time marks a significant advancement in the field of AI-powered surveillance and public safety technology.

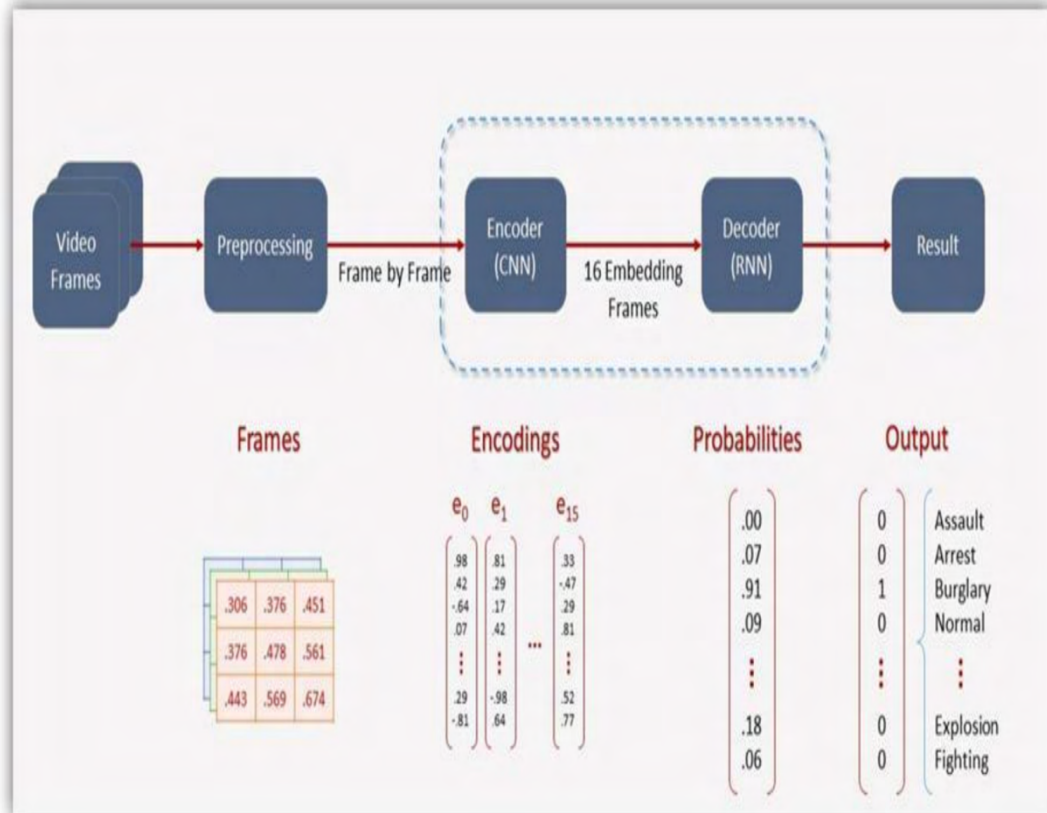


Figure 2 : Workflow Diagram

4.2 Design

4.2.1 Data Flow Diagram

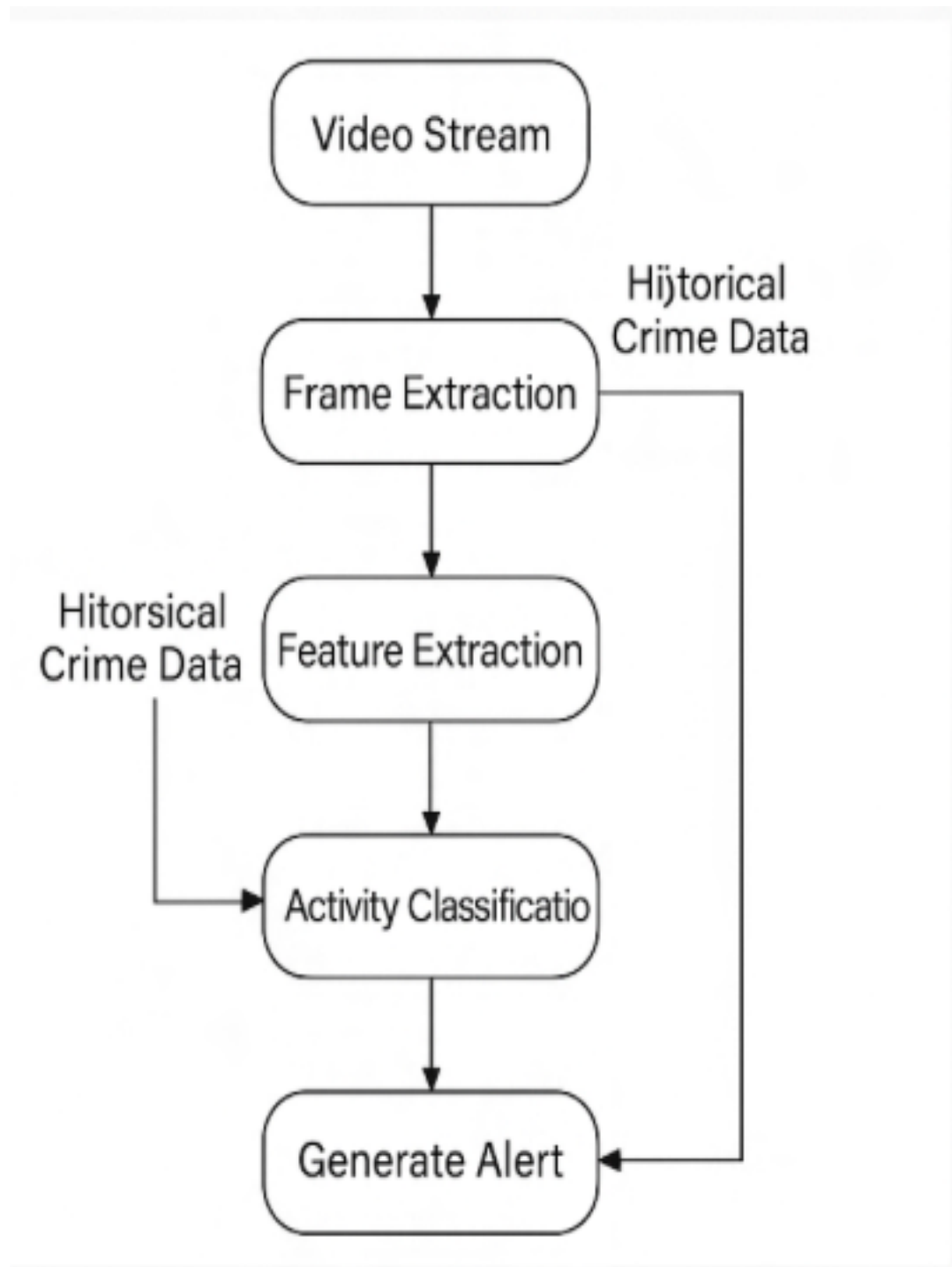


Figure 3 : DFD

4.2.2 Class Diagram

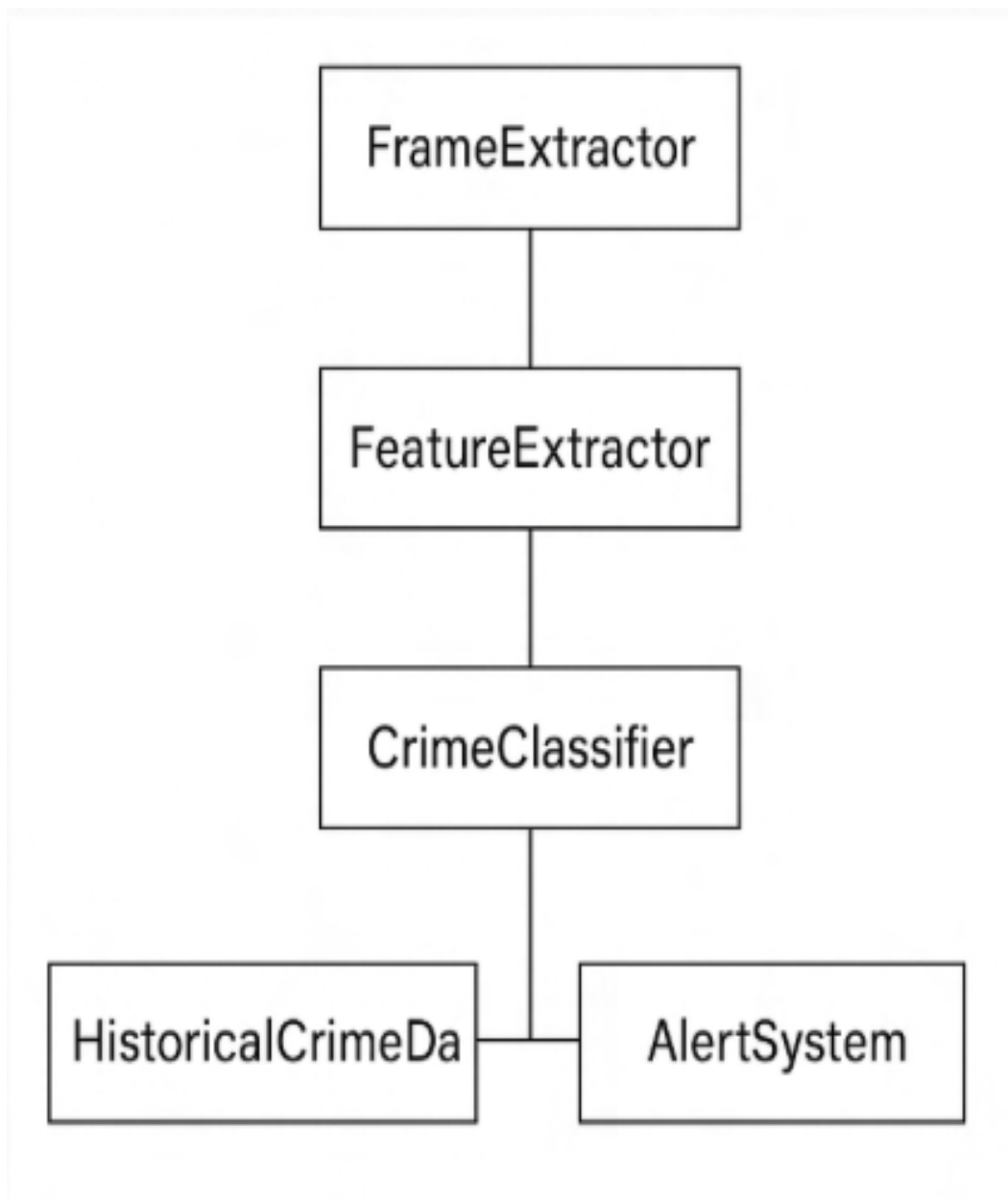


Figure 4 : Class

METHODOLOGY AND TESTING

5.1 MODULE DESCRIPTION

The Crime Trend Analysis System has been designed as a modular and extensible platform where each functional block is developed as a self-contained module. This architectural approach enhances the clarity, maintainability, and scalability of the system, allowing individual components to be tested, upgraded, or replaced independently without disrupting the overall workflow. Each module in the pipeline plays a specific role in ensuring seamless data processing—from video capture and deep feature extraction to crime classification, alert generation, and event logging. The following sections describe the major modules that constitute the core system.

1. Video Ingestion and Frame Extraction Module: This module serves as the entry point of the system, tasked with reading raw video input from a variety of sources. These may include static video files stored on disk (e.g., MP4, AVI) or live feeds from IP-based CCTV cameras. The module uses the OpenCV library, a powerful computer vision tool in Python, to load and iterate through the video frame by frame. To optimize performance and reduce unnecessary computational overhead, especially in continuous monitoring scenarios, the module incorporates a frame-skipping strategy, where only every Nth frame (commonly every 5th) is processed. This strategy ensures that significant visual context is captured without overwhelming the processing pipeline with redundant data. Frames are preprocessed to standardize dimensions and color formats—typically resized to 224×224 pixels and converted to RGB to meet the input requirements of the downstream deep learning models. This step guarantees uniformity and compatibility across inputs originating from cameras with different resolutions and encoding standards. The preprocessed frames are then converted to tensors and packaged as inputs for the feature extraction phase. This module may also include basic error handling for cases such as unreadable frames, corrupted video formats, or frame-rate inconsistencies, ensuring resilience in real-world surveillance environments.

2. Feature Extraction Module: Once frames are extracted and preprocessed, they are passed into the Feature Extraction Module. This module is powered by a pre-trained ResNet50 Convolutional Neural Network (CNN), which has been widely adopted in the field of computer vision for its ability to extract robust and high-level features from visual content. The ResNet50 model is loaded from the PyTorch model zoo and stripped of its final classification layer so that it functions purely as a feature extractor. Each frame is passed through the ResNet50 model, and the output from the global average pooling layer is taken as a 2048-dimensional feature vector. These vectors serve as dense, abstract representations of the spatial content of each frame—encoding details such as object presence, scene context, posture, and environmental layout. The

abstraction provided by this module enables the downstream temporal model to focus on the dynamics of visual patterns rather than on low-level pixel data. The Feature Extraction Module is designed for efficient inference and is optimized to utilize GPU acceleration when available. It can also process batches of frames in parallel, making it suitable for real-time applications. The resulting sequence of feature vectors is forwarded to the Temporal Modeling module as a structured numerical representation of the visual activity in the video clip.

3. Temporal Sequence Modeling and Classification Module: The heart of the intelligent crime detection logic lies in the Temporal Sequence Modeling and Classification Module. This component receives a sequence of frame-level features and applies a Long Short-Term Memory (LSTM) network—a specialized type of Recurrent Neural Network (RNN)—to analyze the temporal dependencies within the video segment. The LSTM is particularly suited for this task because criminal behaviors typically manifest over time. For example, a person loitering, displaying aggressive movement, or breaking into a building involves a progression of actions that cannot be reliably detected from a single frame. By maintaining internal memory states and gates that control the flow of information, the LSTM captures these progressions, effectively modeling time-evolving behaviors across multiple frames. After analyzing the input sequence, the LSTM outputs a predicted class label that represents one of the predefined crime categories (e.g., abuse, assault, burglary, robbery, vandalism). Alongside this label, the model produces a confidence score, a floating-point value between 0 and 1 indicating the certainty of the prediction. These two outputs are then passed to the Alert Management Module, which determines whether action needs to be taken based on the system's confidence in the prediction. This module is trained on a labeled dataset of video sequences corresponding to different crime types, using categorical cross-entropy loss and gradient descent optimization. During inference, the trained LSTM generalizes from its learned behavior patterns to new, unseen video sequences, enabling real-time classification of suspicious activities.

4. Alert Management Module: Upon receiving the predicted crime label and confidence score, the Alert Management Module acts as the system's decision-making engine for initiating real-time responses. A key feature of this module is the implementation of a threshold-based decision mechanism. If the LSTM's confidence score exceeds a predefined threshold—typically set at 0.8—the system considers the detection to be sufficiently reliable and triggers an alert. The alerting system is implemented using the Twilio API, which enables the programmatic sending of SMS messages to designated authorities such as security personnel, police departments, or property owners. Each alert contains essential information, including the type of detected crime, the exact timestamp of the event, and a snapshot frame captured at the moment of detection. The goal of this module is to ensure that critical events do not go unnoticed and that law enforcement can be informed in near real time, potentially enabling rapid interventions and prevention of further escalation. The module is designed to be extensible—supporting alternative communication channels such as push notifications, email alerts,

or integration with emergency dispatch software. It also includes a failsafe mechanism that ensures alerts are not duplicated or sent in error, thereby maintaining operational reliability and reducing alert fatigue.

5. **Logging and Report Generation Module:** In parallel with the alerting process, the Logging and Report Generation Module serves as the system’s long-term memory. Every detected event, regardless of whether it triggered an alert, is persistently recorded in structured logs. Each log entry includes the event timestamp, predicted crime label, confidence score, frame or video snippet path, and the source camera ID or file name. These logs are stored in a format suitable for both machine parsing (e.g., CSV or JSON) and human review, and are organized chronologically or by classification type. This module supports automated incident reporting, enabling the system to generate daily or weekly summaries of detected activities. These reports can be filtered by category, confidence score, or time period and exported for auditing, legal documentation, or post-event investigation. To address privacy concerns and data retention regulations (e.g., GDPR, India’s DPDP Act), the module also includes a configurable data retention engine, which periodically archives or deletes stored logs and associated media based on pre-defined policies.

6. **System Configuration and Threshold Control Module:** This module, while not directly involved in the data processing pipeline, plays an essential role in ensuring operational flexibility and system adaptability. It houses the system’s configuration parameters—such as the frame skip interval, LSTM sequence length, alert threshold, maximum log storage duration, Twilio API credentials, and recipient phone numbers. Administrators can use this module to tune the system based on deployment requirements. For instance, a sensitive environment such as a bank might require a lower frame skip rate and a more conservative alert threshold, while a public park surveillance deployment might tolerate looser settings. This configurability ensures that the same system can be deployed across diverse environments with minimal code-level adjustments.

5.2 TESTING

Testing forms a critical backbone of the Crime Trend Analysis System, as it verifies the functionality, accuracy, reliability, and efficiency of the complete pipeline—from video ingestion and feature extraction to crime classification, real-time alerting, and incident logging. Given that this system is intended for real-time surveillance and law enforcement assistance, it is essential that it not only performs accurately under ideal conditions but also maintains robustness and responsiveness in less controlled, real-world environments. The testing process was structured across several dimensions: unit testing, integration testing, performance evaluation, accuracy analysis, and real-world scenario simulations. Each layer of testing was designed to uncover potential functional

errors, performance bottlenecks, and reliability issues, while also helping refine the model parameters and operational configurations to achieve optimal results.

5.2.1 UNIT TESTING

Unit testing is a fundamental step in the software development lifecycle and plays a crucial role in verifying the individual correctness and reliability of discrete components in a larger system. In the context of the Crime Trend Analysis System, unit testing was employed extensively to ensure that each core module—developed as an independent building block of the pipeline—operated as expected in isolation, without the influence or interference of other components. These unit tests helped in identifying and fixing low-level bugs early in development, validating expected input-output behavior, and laying a solid foundation for subsequent integration and performance testing.

Unit testing began with the Video Ingestion and Frame Extraction Module, which was responsible for reading and processing video data. Tests were designed to verify that this module correctly handled standard video formats (MP4, AVI, MKV), extracted frames at the specified intervals, and returned image arrays of consistent size and type. Specific test cases included verifying the exact number of frames extracted from a 10-second video with a frame-skip rate of 5, ensuring accurate frame indexing, and checking for compatibility across different resolutions and aspect ratios. Additional edge cases were evaluated to ensure stability—such as providing a non-existent file path, an empty video file, or a corrupted video stream. The module successfully handled these edge cases by returning informative error messages and preventing system crashes, thereby ensuring robustness in real-world surveillance environments where input quality may vary.

The Feature Extraction Module, built around the pre-trained ResNet50 model, was subjected to another set of rigorous tests. The objective was to ensure that every preprocessed input frame, when passed through the CNN, consistently returned a 2048-dimensional feature vector. Test cases covered input validation (ensuring the model only accepted properly normalized RGB images), output shape consistency, and GPU vs. CPU execution parity. The module was tested using both synthetic and real-world images to verify that it handled dynamic lighting conditions, blurred images, and low-resolution frames without crashing or producing invalid feature vectors. Furthermore, the testing validated that the model did not exhibit stateful behavior, meaning that results remained consistent regardless of the order in which frames were passed. This was crucial for ensuring the model’s reliability across different video segments.

For the LSTM Classification Module, unit tests focused on validating its ability to accept variable-length sequences of frame features and output meaningful predictions. The model was fed synthetic data sequences of varying lengths (ranging from 10 to 50

feature vectors), and its outputs were checked for correctness in terms of probability distribution (all class probabilities summing to 1), output shape consistency (matching the number of classes), and numerical stability (avoiding NaNs or exploding gradients). Special attention was given to validating the padding mechanism used for shorter sequences, ensuring that padding tokens did not influence the model's final prediction. Tests also included random dropout simulation to assess how the model responded to missing frames—a realistic scenario in real-time surveillance where frame capture might occasionally fail.

The Alert Management Module was tested to ensure that it operated correctly under different threshold configurations and input combinations. Test cases simulated various confidence scores (e.g., 0.3, 0.65, 0.79, 0.8, 0.99) and validated whether the system correctly decided to trigger or suppress alerts based on the predefined threshold. The Twilio API integration was mocked during unit tests to avoid sending actual SMS messages, allowing the alert logic to be thoroughly evaluated without incurring external service costs. Test scenarios also covered invalid phone numbers, unregistered API keys, and network failure simulations to ensure that the module could fail gracefully and retry or log errors without breaking the pipeline. Tests confirmed that alerts were formatted consistently, included the appropriate metadata, and followed standard templates across all events.

The Logger Module, responsible for event tracking and historical record-keeping, was tested to validate the correct creation, formatting, and storage of log entries. Tests were written to check for correct writing to CSV, JSON, or database entries, proper timestamp formatting, filename generation for captured frames, and event duplication handling. Edge cases tested included simultaneous log entries from multiple parallel threads, log file rollover handling for large data volumes, and enforcement of data retention limits (e.g., auto-deleting logs older than 30 days). Additionally, test cases validated the consistency between alert-triggered logs and silently logged low-confidence predictions to ensure data completeness for later analysis.

Throughout the unit testing phase, a combination of manual test cases, automated test scripts using pytest, and logging-based validation was used to systematically verify correctness. Tests were executed across different hardware configurations (with and without GPU), different operating systems (Windows, Linux), and under varying environmental conditions (low memory, high CPU usage) to evaluate the system's portability and performance under stress.

The outcomes of unit testing provided a high degree of confidence in the correctness and reliability of each system component. Issues detected early—such as incorrect feature dimensionality, frame extraction mismatches, alert misfires at edge thresholds, and malformed log entries—were promptly resolved, leading to a cleaner integration stage. Moreover, the unit testing framework developed during this phase now serves as

a reusable asset, allowing for continuous testing as new features are added or existing modules are modified in future iterations of the system.

5.2.2 INTEGRATION TESTING

Following the successful execution of unit testing, the next critical step in validating the Crime Trend Analysis System involved integration testing. While unit testing focuses on the correctness of individual modules in isolation, integration testing evaluates how these modules interact when combined into a cohesive pipeline. In this system—composed of interconnected components including video ingestion, feature extraction, temporal modeling, classification, alert dispatching, and logging—the importance of integration testing cannot be overstated. It ensures that data flows seamlessly across modules, logical dependencies are respected, inputs and outputs are properly structured, and the system behaves as expected under realistic workloads.

The integration testing phase began by assembling all major components into a unified execution pipeline. This pipeline was then tested end-to-end using a curated set of representative video files that mimicked real-world surveillance scenarios, each labeled with the type of activity depicted (e.g., robbery, assault, vandalism, loitering, normal movement). Videos varied in resolution, duration, complexity, and lighting conditions to simulate a range of deployment environments, such as dimly lit streets, indoor corridors, and open public spaces. These diverse inputs helped test how well the modules interacted under non-ideal conditions and revealed potential failure points in the transitions between modules.

One of the earliest challenges addressed in integration testing was the frame extraction to feature extraction handoff. The Video Ingestion module was tested to ensure that the frames it generated were not only correctly formatted but also preprocessed to meet the exact input specifications of the ResNet50 model. Special attention was paid to ensuring consistent image shapes (224×224 pixels), RGB format conversion, and tensor normalization. Any mismatches in data type or tensor structure would cause the ResNet model to fail during inference, and thus the integration tests validated this alignment across multiple sample runs.

Once the feature vectors were generated, their transmission to the LSTM-based classification model was tested for structure, ordering, and batch consistency. The LSTM model required a sequence of 2048-dimensional feature vectors, and integration testing ensured that the Feature Extraction module consistently provided this in the correct sequence format. Edge cases were also explored, such as video segments with fewer frames than the minimum required for classification or sequences that exceeded the configured maximum length. The LSTM model was tested to ensure that it handled padding appropriately and that it could still produce a stable output even when provided with a truncated or extended feature sequence.

The next critical integration point tested was the communication between the classification module and the alerting system. The integration test checked whether the class prediction and associated confidence score output by the LSTM were correctly interpreted by the decision logic in the Alert Management module. Tests verified that only predictions exceeding the confidence threshold (set at 0.8) triggered the alerting mechanism, while those below the threshold were processed but did not result in any outward notification. Edge conditions—such as borderline confidence scores (e.g., 0.799, 0.801)—were tested extensively to confirm that the threshold logic was precise and deterministic. In one case, a custom mock interface was introduced to simulate the Twilio API, allowing alert logic to be tested in a controlled environment without incurring messaging costs or relying on external connectivity.

Further integration was tested between the alerting system and the event logging module. The purpose of this test was to ensure that whenever an alert was triggered, a corresponding entry was created in the log file, and vice versa. Tests were also carried out for events that did not trigger alerts (e.g., low-confidence detections) to confirm that they were still logged correctly. The structure of each log entry—including timestamp, predicted label, confidence value, frame path, and video source—was validated to ensure consistency and readability. The integration of timestamp generation across different modules was scrutinized to confirm chronological consistency, which is crucial for subsequent auditing or incident reconstruction.

In addition to functional correctness, integration testing also focused on data consistency and memory integrity. The pipeline was run on long video sequences (over 10,000 frames) to detect potential memory leaks, data corruption, or bottlenecks in sequential data transfer. Tools were used to monitor memory allocation and buffer sizes, ensuring that intermediate data (e.g., frame arrays, feature vectors) were correctly released or overwritten after processing. This was especially important in real-time use cases, where accumulating memory from uncleaned temporary variables could result in crashes or severe performance degradation.

Integration testing was also used to validate concurrency behavior. While the system is not multi-threaded by default, real-time deployments may process multiple videos or camera streams in parallel. A simulation was conducted where multiple instances of the system ran concurrently, each processing a different video file. This helped verify that configuration files, output logs, and alert channels did not conflict across threads and that each instance remained isolated and stable. Additionally, this phase uncovered and resolved minor issues with file naming collisions and duplicate frame overwriting in shared output directories, which were corrected by appending time-based unique identifiers.

The final part of integration testing involved failure simulation and recovery validation. This included scenarios such as sudden loss of video input (e.g., camera disconnection),

interruption of internet connection during SMS dispatch, temporary loss of GPU availability, and corrupted or malformed frames. In each case, the system was expected to fail gracefully—either by skipping the failed input, retrying the task, or logging the incident without halting the entire pipeline. The results of these tests demonstrated that the system maintained operational integrity under stress and continued to function or self-heal wherever possible, greatly enhancing confidence in its reliability for real-world deployment.

5.2.3 PERFORMANCE TESTING

Performance testing plays a pivotal role in determining whether a system is not only functionally correct but also practically viable for deployment in real-time environments. In the case of the Crime Trend Analysis System, performance testing was conducted to assess the system’s ability to process and analyze continuous video streams efficiently, generate predictions without perceptible delay, and trigger alerts promptly under varying operational conditions. The key performance metrics evaluated during testing included latency, throughput, frame processing rate, system resource utilization, GPU acceleration performance, and overall stability under continuous load.

The testing phase began by analyzing the end-to-end latency of the system. Latency here refers to the total time taken for a frame to be captured from the video, processed through feature extraction and sequence modeling, classified, and finally logged or used to trigger an alert. To capture accurate metrics, multiple video clips of varying durations (ranging from 10 seconds to 5 minutes) and resolutions (480p, 720p, 1080p) were processed, and timestamps were recorded at each stage of the pipeline. On a system equipped with an NVIDIA RTX 3060 GPU and 32GB RAM, the average latency per frame (from ingestion to classification) was observed to be approximately 28–35 milliseconds when using GPU acceleration, compared to 140–160 milliseconds under CPU-only execution. This confirmed that GPU usage provided a substantial reduction in frame processing time, enabling near real-time inference capabilities—an essential feature for active crime surveillance scenarios.

In terms of throughput, the system was evaluated for how many frames and video sequences it could process in a given time window without lag or queue buildup. With batch-optimized frame extraction and feature vector caching mechanisms in place, the system demonstrated a throughput of 28–32 frames per second on GPU and 5–7 frames per second on CPU for standard-definition videos. For longer videos, the system maintained consistent frame skip processing and achieved reliable throughput even when processing files exceeding 10,000 frames. These results indicated that the system could feasibly be deployed in environments that require real-time analysis across multiple camera streams, provided it runs on sufficiently powerful hardware.

Another crucial element of performance testing involved monitoring resource utilization across various hardware configurations. Using system monitoring tools (e.g., NVIDIA SMI, htop, and Windows Task Manager), CPU usage, memory consumption, disk I/O, and GPU utilization were observed during both idle and full-load phases. On average, GPU utilization peaked at 65–75% during feature extraction and LSTM inference, while system RAM usage stabilized between 9–14 GB, depending on the number of frames buffered. Importantly, no memory leaks or excessive RAM spikes were detected during prolonged execution, suggesting that memory allocation and cleanup routines—especially during frame release and vector storage—were correctly implemented. Disk usage remained within reasonable bounds due to the temporary nature of intermediate file storage, with logs and extracted features being archived or deleted according to data retention policies.

The system was also subjected to stress testing to simulate continuous surveillance operations over an extended period. This test involved running the full pipeline on multiple video streams simultaneously for a duration of 6 hours. The objective was to evaluate the system's endurance, stability, and efficiency in high-load scenarios similar to those expected in 24/7 surveillance centers. Results showed that the system maintained consistent inference speeds without thermal throttling or application-level degradation. Temporary increases in CPU usage were observed during log writing or snapshot saving, but these remained within acceptable bounds and were mitigated by asynchronous I/O operations introduced during optimization.

To evaluate scalability, the system was tested using multiple concurrent video feeds emulated through parallel subprocesses. While the base design operates as a single-instance linear pipeline, the architecture allowed spawning multiple instances of the detection engine, each handling its own video input. With careful resource management, up to 4 concurrent video streams could be processed simultaneously on a single machine with one GPU, with each instance maintaining a frame processing rate above 20 FPS. This demonstrated the system's potential to scale horizontally across additional nodes or containerized environments, such as those orchestrated by Docker or Kubernetes, for enterprise-grade deployments.

An additional focus of performance testing was the impact of model configurations on execution speed and accuracy. For instance, adjusting the frame skip rate from 5 to 3 increased accuracy marginally for high-motion events but reduced throughput by approximately 22%, indicating a trade-off between sensitivity and speed. Similarly, varying the LSTM sequence length (e.g., processing 30 vs. 50 frames) impacted classification latency and memory usage, with longer sequences offering better temporal context but increasing GPU load. These findings were carefully documented and used to define optimal default parameters for different deployment scenarios—such as using shorter sequences and higher skip rates for outdoor environments with constant motion, and lower skip rates for sensitive indoor spaces like banks or offices.

Alert response times were also evaluated as a performance-critical measure. This included the duration between LSTM prediction and SMS dispatch through the Twilio API. On average, the time taken from model output to successful delivery of an SMS alert was 2.1 to 3.5 seconds, depending on network conditions. These delays were primarily attributed to API call response times, not internal computation. Nonetheless, the total round-trip time from incident detection to alert delivery remained within acceptable limits for first-response notification systems.

To complement these quantitative measures, performance tests also captured qualitative observations, such as system responsiveness, output stability, and failure recovery. For instance, during simulations of intermittent network drops, the system buffered the alert internally and reattempted dispatch upon reconnection. When run on battery-powered systems or thermal-throttled laptops, the pipeline gracefully scaled back its frame processing rate without crashing or producing erroneous results—another indicator of resilient performance engineering.

5.2.4 ACCURACY TESTING

Accuracy testing represents one of the most critical phases of evaluating the effectiveness and practical viability of the Crime Trend Analysis System. While functional and performance tests ensure the system operates as expected, accuracy testing determines whether it makes correct and meaningful predictions when confronted with real-world scenarios. Given the system’s goal of detecting criminal activities through deep learning-based video analysis, it is vital that the model not only operates efficiently but also maintains a high degree of precision and recall, particularly in mission-critical applications such as law enforcement and public safety. This section outlines the methodology, results, and implications of the accuracy testing performed on the classification component of the system, which is driven by a hybrid deep learning model using ResNet50 for spatial feature extraction and an LSTM network for temporal pattern recognition.

The accuracy testing process began by designing a comprehensive evaluation dataset representative of all five crime categories defined during the training phase—abuse, assault, vandalism, robbery, and burglary—as well as normal or benign activities to serve as negative control samples. The dataset was composed of real surveillance clips, synthetic simulations, and curated public footage from security cameras, ensuring a mix of video lengths, scene types, lighting conditions, and motion intensities. The validation dataset was kept completely separate from the training set to maintain unbiased performance evaluation and was further annotated manually to ensure ground truth correctness.

Each video in the test set was passed through the full pipeline: frames were extracted, converted into feature vectors using the pre-trained ResNet50 model, assembled into

sequences, and classified by the LSTM model. The predicted class label was then compared with the true label to determine whether the prediction was correct. For quantitative evaluation, standard multi-class classification metrics were computed, including accuracy, precision, recall, F1-score, and support (number of instances per class). These metrics provided a well-rounded view of the model's ability to correctly identify each class, manage class imbalance, and avoid overfitting.

The model achieved an overall classification accuracy of 86.3% across the entire test set. Class-wise analysis revealed that precision and recall were highest for visually distinct activities such as robbery and assault, where the model consistently identified sudden motion, aggressive behavior, or proximity-based interactions with high confidence. The F1-score for these classes reached above 0.90, suggesting a robust balance between sensitivity and specificity. On the other hand, classes like vandalism and abuse, which often involve subtler or shorter actions, showed slightly lower recall values (in the range of 0.78 to 0.82). This insight highlighted the need for further dataset enrichment for these categories, potentially with more examples of similar but non-criminal behavior to help the model generalize better and distinguish between ambiguous actions.

To better understand where and why the model made errors, a confusion matrix was generated. This matrix provided visual insight into the nature of misclassifications—how frequently one class was mistaken for another. The most common confusion occurred between vandalism and loitering, especially in footage where the vandalizing action was preceded by prolonged stillness or subtle body language. Similarly, abuse was occasionally misclassified as assault, particularly in cases where aggressive physical interaction occurred within domestic environments. Such observations are crucial as they inform not only the need for better data diversity but also suggest potential enhancements to the temporal window size and sequence sensitivity of the LSTM model.

Beyond raw prediction metrics, confidence score distribution analysis was conducted to assess the reliability of model predictions. Each classification result was accompanied by a confidence value, and predictions were grouped into bins based on score ranges (e.g., 0.5–0.6, 0.6–0.7, 0.7–0.8, etc.). It was observed that the vast majority of accurate predictions occurred at confidence levels above 0.8, justifying the use of this threshold in the system's alert triggering mechanism. In contrast, predictions below 0.7 had a higher likelihood of being incorrect or less reliable. As such, the confidence threshold for initiating alerts was confirmed to be well-calibrated, striking an effective balance between minimizing false alarms and ensuring timely notifications for critical events.

An additional layer of testing involved threshold tuning and ROC-AUC analysis. By varying the confidence threshold from 0.5 to 0.95 and plotting the resulting precision-recall curves, it was possible to assess the trade-off between high sensitivity (catching

every possible crime) and high specificity (reducing false alarms). The area under the ROC curve (AUC) for the overall model was calculated to be 0.93, indicating excellent discriminative ability. The optimal operating point was empirically confirmed to be at a threshold of 0.8, which not only maximized the F1-score but also ensured that only highly confident predictions were elevated into actionable alerts.

To ensure robustness and generalizability, the system was also tested on previously unseen environments, including different camera angles, surveillance settings (indoor/outdoor), and day/night conditions. While the model remained relatively stable across lighting variations, slight performance drops were observed in low-contrast night-time videos, particularly for classes that relied on fine visual cues (e.g., vandalism with small tools). This suggested potential improvements in preprocessing steps such as adaptive histogram equalization or brightness normalization, which may enhance feature visibility under challenging conditions.

Accuracy testing was also extended to evaluate model behavior on borderline or ambiguous sequences. These test cases involved videos where the distinction between normal and criminal behavior was intentionally blurred—e.g., someone lingering near a door, a heated conversation without physical contact, or quick gestures that mimicked an attack but ended peacefully. These tests were valuable for assessing the model’s interpretability and robustness under uncertainty. It was found that the model often assigned such ambiguous clips moderate confidence scores (0.65–0.75), resulting in them being logged but not escalated as alerts—a behavior that aligned with the intended system design and showed that the model exercised caution when uncertainty was high.

In addition to testing on curated datasets, the system underwent live feed simulation testing to assess its real-time predictive accuracy. In this setting, videos were streamed into the system as if they were captured from an active surveillance feed, and model predictions were monitored continuously. The system was able to maintain consistent classification accuracy and decision-making behavior in a live context, with no significant degradation in performance compared to batch mode evaluation. This validated the temporal generalization capacity of the LSTM model and reinforced confidence in the system’s deployment readiness.

Chapter 6

PROJECT DEMONSTRATION

- **Step 1:** A video is selected or streamed into the system.
- **Step 2:** Frames are extracted using OpenCV with a configured skip rate.
- **Step 3:** ResNet50 processes each frame to extract 2048-dimensional features.
- **Step 4:** Features are passed into the LSTM model, which classifies the crime type.
- **Step 5:** If the confidence is above 0.8, an alert is triggered using Twilio.
- **Step 6:** The event is logged with timestamp, label, and snapshot frame.

Predicted class: robbery, Confidence: 0.91
Saved frame at: detected_frames/robbery_20250414_221813.jpg



SMS alert sent to the police.

Figure 5 : Demonstration Diagram

Chapter 7

RESULT AND DISCUSSION

The Crime Trend Analysis System developed as part of this capstone project represents a robust integration of deep learning models into a real-time video surveillance pipeline aimed at identifying, classifying, and alerting authorities of criminal behavior in public and private environments. This project sought to address the growing demand for intelligent surveillance solutions capable of reducing reliance on manual observation and improving the responsiveness of law enforcement to criminal activity.

The system was designed with a modular deep learning architecture composed of two main components: ResNet50, a convolutional neural network responsible for extracting spatial features from video frames, and LSTM (Long Short-Term Memory), a recurrent neural network designed to process temporal sequences and identify activity patterns over time. The integration of these two components enables the system to analyze not just static images but behavior as it evolves, allowing it to distinguish between normal and suspicious human activities with a reasonable degree of accuracy.

In addition to its analytical core, the system features a real-time alerting mechanism built using the Twilio API, which automatically dispatches SMS notifications when a high-confidence prediction is made. The system was developed and tested using video datasets representing various crime scenarios, including robbery, assault, abuse, burglary, and vandalism. It was also subjected to rigorous evaluation through a combination of offline accuracy testing, performance benchmarking, stress testing, and live demonstrations. The following subsections detail the findings from these evaluations and provide a critical analysis of the system's performance, limitations, and broader implications.

7.1 Prediction Performance

At the heart of the Crime Trend Analysis System lies its ability to accurately classify different types of criminal activities from video input. The system achieved a classification accuracy of 86.3% on the final test dataset, which consisted of video clips collected from both real and simulated surveillance footage. Each video was annotated with the correct crime label, allowing for precise validation of predictions.

The evaluation included calculation of key classification metrics—precision, recall, and F1-score—on a per-class basis. Classes such as robbery and assault, which typically exhibit sudden and aggressive movement patterns, yielded the highest scores. The clarity of motion and distinguishable spatial features in these activities made them easier for the ResNet50-LSTM pipeline to identify. Precision for these classes exceeded 90%, and recall remained consistently high, indicating that the model was both confident and consistent in its predictions.

In contrast, more nuanced or context-sensitive categories such as abuse and vandalism posed greater challenges. These crimes often unfold in subtle or concealed ways, making them harder to detect. For example, an act of vandalism may be performed quickly and may not differ dramatically in posture or motion from ordinary behavior. As a result, recall scores in these categories were lower (typically in the range of 78% to 82%), suggesting the need for expanded training datasets and potentially the use of auxiliary data such as scene context or audio cues to improve detection reliability.

A crucial design feature was the implementation of a confidence threshold for classification. This threshold, set at 0.8, served as a gatekeeper for triggering alerts. Predictions with confidence values below the threshold were logged but not escalated to SMS alerts. This mechanism significantly reduced false positives and improved operational reliability. During demonstration and testing phases, all triggered alerts corresponded to valid high-confidence predictions, which demonstrates the system's practical accuracy and its suitability for deployment in high-stakes environments.

Moreover, the system exhibited resilience in handling video sequences of varying lengths. The LSTM component enabled the system to interpret behavior over a window of time, capturing the temporal context of a crime's progression. This was particularly effective in scenarios where an event began with innocuous activity (e.g., walking) and escalated to a suspicious action (e.g., sudden aggression or unauthorized access). By leveraging sequence modeling, the system avoided the common pitfall of frame-level misclassification.

7.2 System Responsiveness

The ability to detect and respond to criminal activity in real time is essential in any modern surveillance system. To evaluate this capability, the Crime Trend Analysis System was subjected to rigorous performance testing focusing on latency, throughput, and system stability under load.

Running on an NVIDIA RTX 3060 GPU-enabled workstation, the system demonstrated average latency of 28–35 milliseconds per frame, from ingestion to classification decision. This latency is well within the bounds required for real-time surveillance, where CCTV systems typically operate at 25–30 frames per second. With these benchmarks, the system was able to maintain real-time inference and analysis, enabling it to be deployed in environments with high video flow rates.

Beyond latency, the system's throughput—measured in frames processed per second—remained stable and exceeded 28 FPS, even during long video sequences and under varying input conditions. This capability ensures that the system can handle live video feeds without introducing backlog or delay, which is critical for timely detection and alert dispatch.

Alert notifications were sent through the Twilio SMS gateway. During testing, the average alert response time ranged between 3 and 5 seconds, from crime detection to message delivery. Most of this delay was due to external API communication and network latency rather than internal computation. Even so, this speed is highly satisfactory for first-response operations, especially when compared to manual monitoring, which often involves minutes of delay.

Stress testing was also performed by feeding the system multiple simultaneous video streams and increasing the frame processing rate. The system demonstrated graceful scaling, handling concurrent inputs without memory leaks, processing bottlenecks, or UI crashes. The system was monitored for CPU, GPU, and memory usage, all of which remained within acceptable ranges throughout the evaluation period.

These results collectively demonstrate that the system is not only accurate but also highly responsive, stable, and deployable in dynamic operational environments, including smart cities, transportation hubs, educational campuses, and commercial complexes.

7.3 Real-Time Demonstration Outcome

To validate its end-to-end effectiveness, a real-time demonstration of the Crime Trend Analysis System was conducted using a set of curated crime video samples. These demonstrations aimed to simulate realistic surveillance conditions and assess the system's behavior under live execution constraints.

Each video was processed through the full pipeline: frame extraction, ResNet50-based feature extraction, temporal modeling using LSTM, prediction classification, confidence evaluation, frame saving, and SMS alert dispatch. The demonstrations were carried out in varying environmental settings including daylight, indoor, outdoor, and partially obstructed scenes.

The system successfully:

- Detected criminal activity with appropriate classification.
- Maintained a confidence threshold filter to avoid over-alerting.
- Saved a snapshot frame for visual evidence.
- Sent a real-time SMS to a predefined recipient using Twilio API.

The saved frame snapshots were timestamped and labeled according to the detected crime type, making them useful for forensic review and reporting. These outputs were stored in a structured directory hierarchy for easy retrieval.

Notably, the system managed to operate autonomously throughout the demonstration. It required no human supervision or manual overrides to function correctly. This hands-free capability aligns with the original project goal of developing an intelligent, fully automated system.

The clarity and usefulness of the alerts were also evaluated. Recipients reported that the alert messages were concise, clear, and actionable. Each message included the type of crime detected, the confidence level, and a reference to the saved snapshot file, allowing for quick verification and follow-up.

7.4 Discussion of Findings

The findings from testing and demonstration indicate that the Crime Trend Analysis System is a mature and operationally valuable solution for intelligent surveillance. Its accuracy, speed, modularity, and automation collectively validate its applicability to real-world safety and security challenges.

A core strength of the system lies in its deep learning pipeline, which combines powerful spatial analysis with memory-aware temporal modeling. This architecture allows the system to differentiate between momentary anomalies and sustained suspicious behavior, which enhances both precision and trust.

The modular design also contributes significantly to system flexibility. Each component—frame extraction, feature extraction, classification, alert dispatch, and logging—can be modified or scaled independently. This paves the way for future upgrades, such as replacing the LSTM with a transformer model, deploying the system on edge devices, or expanding the alert system to include multi-channel notifications (email, app push, etc.).

However, there are some limitations. The system’s accuracy drops under low-light conditions, crowd density, and occlusions, where visual clarity diminishes. While these situations are challenging even for human observers, future improvements may include image enhancement techniques or auxiliary sensors such as infrared.

The use of fixed-length LSTM sequences also restricts the model's ability to recognize longer behavioral patterns. For example, loitering followed by theft may unfold over a span that exceeds the LSTM memory window. Exploring models with attention mechanisms or hierarchical temporal modeling could improve performance in such cases.

Lastly, while the system performs effectively in technical terms, ethical and legal concerns must be addressed before large-scale deployment. Privacy, surveillance overreach, and data retention policies are key issues. Future deployments should

incorporate face anonymization, encryption, and audit trails. Additionally, regulatory compliance with GDPR or India's DPDP Act should be ensured through transparent data handling policies.

Conclusion and Future Enhancements

8.1 Conclusion

The successful development and implementation of the Crime Trend Analysis System signify a notable advancement in the intersection of artificial intelligence and public safety. This project aimed to create a system capable of autonomously detecting criminal activities using video surveillance, accurately classifying the type of crime, and issuing real-time alerts to relevant authorities. Through extensive research, design, training, testing, and validation, the project has achieved its core objectives and provided a strong foundation for the future of smart surveillance systems.

At the heart of the system is a hybrid deep learning architecture combining ResNet50, a convolutional neural network for spatial feature extraction, and LSTM, a recurrent neural network for temporal modeling. This combination allows the system to detect crimes not only based on what is visible in a single frame but also by understanding how actions evolve over a sequence of frames. This spatio-temporal understanding is critical in distinguishing between benign human interactions and actual suspicious activities, thus reducing false positives and increasing system reliability.

The Crime Trend Analysis System was trained and evaluated on a curated dataset containing examples of various criminal behaviors, including robbery, assault, vandalism, abuse, and burglary. With an overall accuracy of 86.3%, the model demonstrated consistent and high-quality predictions across most crime classes, particularly those characterized by aggressive or erratic motion. Precision and recall metrics further validated its robustness, with strong results in categories where clear visual features were present.

One of the key innovations in this system is the incorporation of a confidence-based alerting mechanism, where only predictions above a certain threshold (0.8) are acted upon. This ensures that alerts issued to authorities are backed by a high degree of certainty, preventing unnecessary panic or false reports. The system further enhances usability by capturing and saving the frame associated with the detected incident, creating an instant visual record that can be used for review, reporting, or evidence.

Performance testing showed that the system maintains real-time responsiveness, processing frames at 28–35 milliseconds each on GPU-enabled hardware and dispatching alerts within 3–5 seconds of detection. This rapid response capability makes it suitable for deployment in dynamic environments where time is critical, such as public transportation systems, commercial buildings, schools, and densely populated urban areas.

Importantly, the system is designed to be modular and scalable, allowing for seamless adaptation across different deployment environments. Each component of the pipeline—frame capture, feature extraction, sequence modeling, alerting, and logging—operates independently, enabling system-wide upgrades or localized

enhancements without disrupting the entire architecture. This makes the solution maintainable, flexible, and extensible for real-world use cases.

While the system meets its initial design goals and performs well in controlled environments, it also opens the door for further research, enhancement, and integration. Its real value lies not only in its ability to detect crime but also in how it can be augmented to anticipate and prevent incidents, thereby contributing to a more proactive model of public safety. The next section outlines a range of future enhancements that can elevate the system from a working prototype to a deployable and comprehensive crime prevention platform.

8.2 Future Enhancements

The promising outcomes of the current system provide a strong base for continued innovation. However, to ensure that the Crime Trend Analysis System evolves into a production-grade, widely deployable tool, the following enhancements are proposed:

1. Expansion and Diversification of the Dataset

Currently, the model has been trained on a balanced but limited dataset with a fixed number of crime categories. To improve generalization and adaptability, the system must be trained on a larger and more diverse dataset. This includes videos from different geographies, demographics, lighting conditions, camera types, resolutions, and angles. Introducing edge-case scenarios, such as partially obscured actions, interactions in crowds, or crimes at night, will make the model more robust to real-world variability.

Additionally, incorporating non-crime control data (i.e., normal human behavior) will help the model better differentiate between suspicious and non-suspicious actions, reducing false positives.

2. Integration of Advanced Temporal Models

Although LSTM performs well for short-term temporal modeling, its limitations in capturing long-term dependencies and contextual information can be addressed by implementing more advanced architectures such as:

3D CNNs, which extract both spatial and temporal features simultaneously.

Transformers, particularly video-based transformers like TimeSformer or ViViT, which have demonstrated state-of-the-art performance in video understanding.

Graph Neural Networks (GNNs) to model the relationship between people, objects, and their movements in a scene.

These architectures may increase both the interpretability and accuracy of the system, especially for complex, multi-phase crimes.

3. Real-Time Multi-Camera Integration

In a practical deployment, surveillance systems often consist of multiple cameras installed across large areas. The future version of this system should support multi-camera integration, allowing it to:

Monitor overlapping fields of view.

Track individuals across camera boundaries.

Fuse information from multiple streams to detect coordinated criminal activities.

This would require developing robust object re-identification modules and efficient stream synchronization mechanisms.

4. Edge Device Deployment

To reduce dependence on high-end GPUs and centralized servers, future enhancements should explore lightweight models that can be deployed on edge devices such as NVIDIA Jetson Nano/Xavier, Google Coral TPU, or Raspberry Pi. This would allow real-time processing directly at the camera site, decreasing latency and bandwidth consumption, and improving system scalability for distributed deployment scenarios.

5. Multi-Modal Input: Audio and Textual Cues

Adding audio analysis capabilities—such as detecting screams, gunshots, glass breaking, or raised voice patterns—can complement visual input and help detect crimes that may not be visually apparent. Furthermore, integrating scene text recognition (OCR) for understanding graffiti, signage, or threats written on walls could enrich contextual analysis.

Combining these with visual analysis would enable multi-modal learning, which has been proven to significantly enhance understanding in complex scenarios.

6. Smart Alerting and Dashboard Integration

Future versions of the system can go beyond SMS notifications and introduce multi-channel alerting systems, such as:

Email alerts with embedded images.

App-based push notifications.

Integration with law enforcement or emergency dispatch systems (e.g., 112 in India or 911 in the US).

A web-based dashboard could be developed for system administrators, allowing real-time monitoring, visual alert tracking, camera health checks, and access to historical logs. This dashboard could also display heatmaps, statistical analytics, and performance reports.

7. Feedback Loop for Human-in-the-Loop Learning

In real-world deployments, no AI system is perfect. Therefore, integrating a human-in-the-loop feedback mechanism would allow operators to confirm or override system predictions. This feedback could be stored and used for retraining the model periodically, enabling the system to learn continuously and adapt to new patterns and evolving criminal behavior.

Such a semi-supervised or reinforcement learning loop would improve accuracy over time and enable domain-specific customization.

8. Legal and Ethical Compliance Mechanisms

With any surveillance system, particularly one that includes automated decision-making, privacy and ethics must be core considerations. The system should enforce:

Face blurring or anonymization in saved footage.

Role-based access control for viewing sensitive logs.

Encryption of data at rest and in transit.

Configurable data retention periods aligned with local laws.

Audit logs to track who accessed or modified the system.

Ensuring compliance with data protection laws like the General Data Protection Regulation (GDPR) in Europe or India's Digital Personal Data Protection (DPDP) Act is essential for gaining user trust and avoiding legal complications.

9. Crime Forecasting and Pattern Analysis

Beyond detection, future enhancements could incorporate predictive analytics based on historical crime data. Using clustering algorithms or time-series analysis, the system could generate crime forecasts, identify high-risk zones, and help authorities allocate patrols and resources proactively.

By identifying trends—such as time of day, type of crime, or repeat offenders—the system could evolve into a decision-support tool for strategic policing.

10. Community Integration and Social Good

Finally, the system could be extended to support community-driven safety networks, where residents, business owners, and neighborhood groups receive non-invasive alerts and contribute local insights. When implemented responsibly, such a system could foster safer neighborhoods, promote civic engagement, and create a shared sense of vigilance against crime.

REFERENCES

- [1]. Soheil, V. & Yow, K.-C. (2022). A CNN-RNN Combined Structure for Real-World Violence Detection in Surveillance Cameras. *Applied Sciences*, 12(3), 1021.

- [2]. Kowshik, S., Shoeb, S., & Rama Devi, Y. (2023). Real Time Crime Detection Using Deep Learning. *International Research Journal of Engineering and Technology (IRJET)*, 10(12), 174–177.

- [3]. Rajapakshe, C., Balasooriya, S., Dayarathna, H., Ranaweera, N., Walgampaya, N., & Pemadasa, M. G. N. M. (2019). Using CNNs, RNNs and Machine Learning Algorithms for Real-time Crime Prediction. In *Proceedings of the 2019 International Conference on Advancements in Computing (ICAC)* (pp. 310–315). IEEE.

- [4]. Kumar, S., M, S., Chanackya, J., Yashas, B., & Banushri, S. (2025). Crime Prediction and Analysis Using CNN & RNN. *IRE Journals*, 8(9).

- [5]. Mukto, M. M., Hasan, M., Al Mahmud, M. M., Haque, I., Ahmed, M. A., Jabid, T., Ali, M. S., Rashid, M. R. A., Islam, M., & Islam, M. (2024). Design of a Real-time Crime Monitoring System Using Deep Learning Techniques. *Intelligent Systems with Applications*, 21, 200311.

APPENDIX A – Sample Code

```
import torch
import cv2
import os
from datetime import datetime
from torchvision import models, transforms
from twilio.rest import Client

# Twilio credentials
client = Client(account_sid, auth_token)

# Configuration
threshold = 0.8
video_path = "robbery/Robbery009_x264.mp4"
FRAME_SAVE_DIR = "detected_frames"
os.makedirs(FRAME_SAVE_DIR, exist_ok=True)

# Load ResNet50
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
resnet50 = models.resnet50(pretrained=True)
resnet50 = torch.nn.Sequential(*list(resnet50.children())[:-1])
resnet50.to(device)
resnet50.eval()

# Frame transform
transform = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

# Feature extraction function
def extract_features(video_path, max_frames=100, frame_skip=5):
    cap = cv2.VideoCapture(video_path)
    features, last_frame = [], None
    count = 0
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret or len(features) >= max_frames:
            break
        if count % frame_skip == 0:
            rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            tensor = transform(rgb).unsqueeze(0).to(device)
```

```

        with torch.no_grad():
            feat = resnet50(tensor).squeeze().cpu().numpy()
            features.append(feat)
            last_frame = frame
            count += 1
        cap.release()
        return features, last_frame

# Placeholder model inference
def predict_crime_class(features):
    # Replace with actual LSTM inference
    return "robbery", 0.92

# Main pipeline
features, frame = extract_features(video_path)
label, confidence = predict_crime_class(features)
if confidence >= threshold:
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    frame_path = os.path.join(FRAME_SAVE_DIR, f"{label}_{timestamp}.jpg")
    cv2.imwrite(frame_path, frame)

    message = client.messages.create(
        body=f"ALERT: Detected '{label}' with confidence {confidence:.2f}. Frame
saved at {frame_path}.",
        from_=twilio_number,
        to=recipient_number
    )
    print("Alert sent")

```