International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Smart Surveillance System using Machine Learning for Real-Time Abnormal Activity Detection

P. Rakesh[1], R. Praveena[2], Poorani S[3], Ragavi G[4], N. Nithesh[5], Prof. Sivaramakrishnan A[6]

[1,2,3,4,5]Department of Artificial Intelligence and Data Science, Chettinad College of Engineering and Technology, Anna University, Tamil Nadu, India
[6]Assistant Head of the Department & Associate Professor, Department of Artificial Intelligence and Data Science, Chettinad College of Engineering and Technology

Abstract
The increasing proliferation of surveillance infrastructure worldwide has created an urgent need for intelligent, automated video analysis systems capable of detecting security threats without human intervention. Traditional Closed-Circuit Television (CCTV) systems, while extensively deployed, remain fundamentally passive and offer no ability to autonomously detect or respond to incidents. This paper presents a Smart Surveillance System (SSS) that leverages Machine Learning (ML) and Computer Vision (CV) techniques to detect abnormal and violent human activities in real time. The proposed system accepts multiple input modalities including live webcam feeds, pre-recorded video files, and static images. Video frames are extracted and preprocessed using OpenCV, feature vectors are constructed using Histogram of Oriented Gradients (HOG) descriptors combined with optical flow motion magnitude, and activity classification is performed using a trained Random Forest classifier. Upon detection of abnormal activity, the system triggers automated multi-channel alerts via Email (SMTP) and SMS (Twilio API), while results are displayed on a Streamlit-based interactive dashboard. The system is designed for offline, edge-based deployment on standard consumer hardware without requiring cloud infrastructure or dedicated GPU acceleration. Experimental evaluation on a mixed benchmark and locally recorded dataset demonstrates 91.4% overall classification accuracy, a true positive rate of 93.2%, and an average detection latency of approximately 320 milliseconds per frame. Alert dispatch is completed within two seconds of confirmed detection. The proposed system reduces human operator response time by a factor of 5.2x compared to manual monitoring, establishing it as a practical, scalable, and cost-effective solution for surveillance in campuses, transportation hubs, commercial establishments, and smart city environments.

Keywords: Machine Learning, Computer Vision, Real-Time Surveillance, Abnormal Activity Detection, OpenCV, Random Forest, Streamlit, Alert System.

1. INTRODUCTION

Public safety and security monitoring have emerged as critical societal priorities in the context of rising

global urbanization and the expansion of high-density public spaces. Governments, institutions, and private organizations across the world invest heavily in surveillance infrastructure to protect people, assets, and critical facilities. The global video surveillance market was valued at over USD 45 billion in 2023 and continues to grow rapidly, driven by the expansion of smart city initiatives and increased security awareness following high-profile public safety incidents [10][19].

Despite this massive investment, the fundamental architecture of most deployed surveillance systems remains largely unchanged from systems designed decades ago. Traditional Closed-Circuit Television (CCTV) networks capture and store continuous video streams but lack any capability for automated content analysis. The entire burden of interpreting this footage falls on human operators, who must monitor multiple camera feeds simultaneously for extended periods. This approach suffers from well-documented cognitive limitations: human attention degrades significantly over monitoring sessions exceeding 20 minutes, and studies indicate that operators miss up to 45% of critical events during sustained monitoring tasks [14][15].

The consequences of these limitations are severe. Delayed detection of security incidents such as physical altercations, unauthorized intrusions, or suspicious behavior can result in significant harm to individuals and property before any response is mobilized. Furthermore, the labor costs associated with round-the-clock human monitoring make comprehensive surveillance economically prohibitive for many organizations, particularly educational institutions, smaller municipalities, and private businesses [3][12].

Recent advances in Artificial Intelligence (AI), particularly in the sub-fields of Machine Learning (ML) and Computer Vision (CV), offer a compelling solution to these challenges. ML-based systems can continuously analyze video streams without cognitive fatigue, detect predefined patterns of concern, and generate automated alerts within milliseconds of a triggering event. Computer vision techniques enable machines to extract meaningful spatial and temporal features from raw video data, forming the perceptual foundation for activity recognition and anomaly detection [1][13].

However, a significant gap exists between research-grade AI surveillance systems — which typically require powerful GPU-equipped servers, cloud connectivity, and specialized engineering expertise to deploy and maintain — and the practical needs of real-world security operators who require cost-effective, locally deployable, and easily operable solutions. Many existing approaches also lack integrated communication capabilities, requiring separate systems for alert generation and notification [6][9].

This paper bridges this gap by presenting a Smart Surveillance System (SSS) that integrates ML-based activity classification, computer vision-based video processing, and automated multi-channel alert generation within a single, unified, locally deployable application. The system is designed from the ground up to operate on standard consumer-grade hardware, requires no cloud services for core functionality, and provides a user-friendly Streamlit-based dashboard suitable for non-technical security personnel. The system architecture, implementation, experimental evaluation, and deployment considerations are described in detail in the sections that follow.

The remainder of this paper is organized as follows: Section 2 provides a comprehensive review of related literature. Section 3 formally defines the problem statement and motivation. Section 4 describes the proposed system architecture in detail. Section 5 presents the end-to-end system workflow. Section 6 discusses implementation technologies and design decisions. Section 7 presents experimental results and performance analysis. Section 8 enumerates key system features and application domains. Section 9 discusses current limitations and future research directions. Section 10 concludes the paper.

2. LITERATURE REVIEW

The field of automated video surveillance and abnormal activity detection has evolved considerably over the past two decades, progressing from early rule-based motion detection approaches through classical machine learning pipelines to modern deep learning architectures. This section reviews the most significant contributions to this field and identifies the gaps that the proposed system addresses.

Early approaches to automated surveillance relied on background subtraction and frame differencing methods to identify regions of motion, followed by rule-based heuristics to classify detected events. While computationally inexpensive, these methods were highly sensitive to environmental changes such as lighting variations, camera jitter, and background clutter, resulting in unacceptably high false positive rates in real-world deployments [14][17].

The introduction of machine learning approaches significantly improved classification robustness. Histogram of Oriented Gradients (HOG) features combined with Support Vector Machine (SVM) classifiers became a widely adopted baseline for human activity recognition. These methods demonstrated improved generalization compared to rule-based approaches but remained dependent on carefully engineered feature representations that failed to capture complex temporal dynamics in activity sequences [7][15].

Patel et al. [1] proposed a CNN-LSTM hybrid architecture for real-time violence detection that combines spatial feature extraction via convolutional layers with temporal sequence modeling via Long Short-Term Memory (LSTM) units. This approach achieved accuracy exceeding 90% on standard benchmark datasets and demonstrated the effectiveness of end-to-end deep learning for surveillance applications. Sharma et al. [13] further extended this framework, developing a fully integrated violence detection system using CNN and LSTM that incorporated preprocessing optimization and achieved strong results on diverse video datasets.

The IJSART study [2] evaluated a real-time violence detection system specifically designed for practical deployment scenarios, demonstrating that deep learning approaches can maintain high performance even under challenging real-world conditions including variable lighting and partial occlusion. The IJSRED series of publications [6][8] explored both supervised and self-supervised approaches to violence detection, with the self-supervised lightweight CNN-LSTM model [8] achieving competitive accuracy with significantly reduced computational requirements — an important step toward edge-deployable deep learning surveillance.

Omarov et al. [14] conducted a comprehensive state-of-the-art survey of violence detection techniques in video surveillance published in PeerJ Computer Science, cataloguing methods from classical motion analysis through deep learning pipelines and identifying key challenges including computational complexity, dataset bias, and the difficulty of defining violence in culturally diverse contexts. The PMC literature review [15] complemented this work with an extensive analysis of 67 papers on deep learning-based violence detection, providing a structured taxonomy of approaches and highlighting the persistent trade-off between model accuracy and inference speed.

The SUSAN architecture [16] introduced a novel deep learning framework specifically designed for violence detection in surveillance videos, combining spatial attention mechanisms with temporal pooling to achieve state-of-the-art results on multiple benchmark datasets including the Hockey Fight, Movies, and RWF-2000 datasets. The Expert Systems with Applications study demonstrated SUSAN's applicability to real surveillance scenarios. The ACM comprehensive review [17] surveyed vision-based

violence detection across 80 publications, concluding that hybrid CNN-LSTM models consistently outperform both purely spatial and purely temporal approaches.

The integration of Transformer-based architectures into video surveillance has emerged as a promising research direction. The IJECE study [20] investigated video violence detection using LSTM and Transformer networks, demonstrating that self-attention mechanisms can effectively capture long-range temporal dependencies in action sequences that LSTM architectures sometimes fail to model. While Transformer-based models achieved state-of-the-art accuracy on benchmark datasets, their computational requirements during inference make them impractical for CPU-only edge deployment without model compression or quantization techniques. Dataset diversity and representativeness have been identified as critical limiting factors across the surveillance literature. The PMC review [15] observed that most published models are evaluated on a small number of clean benchmark datasets — particularly the Hockey Fight and Movies Fight datasets — that may not adequately capture the diversity of real-world surveillance conditions including mixed lighting, varying camera resolutions, crowded scenes, and culturally specific patterns of behavior. This observation underlines the importance of supplementing benchmark evaluation with domain-specific local dataset collection, as employed in the evaluation methodology of the proposed system. The IJERT automated violence recognition study [9] specifically addressed the practical integration of alert systems with deep learning detection pipelines, demonstrating that asynchronous multi-channel notification can be integrated without introducing significant latency to the core detection loop. The AI-Propelled Security program described by the U.S. Army [10] and the AI vigilance guard integration study [12] further demonstrate the growing recognition of AI-based surveillance as a critical component of next-generation security infrastructure across both civilian and defense contexts. Despite these impressive research advances, the transition from laboratory systems to practical deployed solutions remains challenging. Most high-performing systems require GPU-equipped servers or cloud processing for real-time operation, making them economically inaccessible for many organizations. Furthermore, few existing systems integrate automated multi-channel alerting directly with the detection pipeline — requiring users to build custom notification infrastructure on top of the detection system [9][18]. The proposed SSS directly addresses both of these practical barriers by providing a complete, integrated, locally deployable surveillance solution with built-in alert capabilities.

3. PROBLEM STATEMENT

The fundamental challenge addressed by this research is the inadequacy of existing surveillance systems to provide timely, reliable, and cost-effective automated detection of security threats in real-world environments. This challenge manifests across multiple dimensions that collectively undermine the effectiveness of current surveillance infrastructure.

Human Monitoring Limitations: Human operators tasked with monitoring surveillance footage face inherent cognitive constraints that severely limit monitoring effectiveness. Sustained attention declines dramatically over monitoring sessions, with research demonstrating that critical event detection rates drop by 45% after 20 minutes of continuous monitoring [14]. In practice, many surveillance installations operate with insufficient staffing, resulting in monitored cameras receiving only periodic or reactive attention rather than continuous observation.

Response Time Deficits: Even when an operator successfully identifies a security incident, the subsequent notification chain introduces additional delays. The operator must assess the severity of the incident, contact supervisors or security personnel, and communicate the nature and location of the threat — a

process that typically takes between 10 and 30 seconds under optimal conditions. This delay is unacceptable in scenarios involving physical violence, medical emergencies, or rapidly evolving security situations where seconds determine outcomes [3][12].

Scalability and Economic Barriers: Large facilities such as university campuses, shopping malls, and transportation hubs may deploy hundreds of cameras across multiple buildings and outdoor spaces. Providing adequate human monitoring for such deployments requires proportional increases in staffing, creating significant ongoing operational costs. Many organizations respond to these cost pressures by reducing monitoring staff, further degrading detection effectiveness and creating false assurance through the presence of cameras that are not adequately monitored [10].

Evidence Management Deficiencies: When incidents occur, the ability to quickly identify, retrieve, and analyze relevant footage is critical for investigation and prosecution. Traditional surveillance systems generate massive volumes of undifferentiated video data with minimal metadata, making targeted retrieval time-consuming and unreliable. The absence of automated event tagging and logging means that critical evidence may be overwritten before investigators can retrieve it, or may be buried within hours of irrelevant footage [11].

Deployment Complexity: Existing intelligent surveillance solutions that do address the above limitations typically require specialized IT infrastructure including dedicated GPU servers, cloud API subscriptions, and trained engineers for installation and maintenance. These requirements place effective intelligent surveillance out of reach for the majority of potential beneficiaries, including educational institutions, small municipalities, and resource-constrained organizations in developing regions [5][7].

The proposed Smart Surveillance System directly addresses each of these problem dimensions by providing continuous automated detection without human monitoring requirements, sub-second threat identification and alert generation, edge-deployable architecture requiring only standard consumer hardware, automated timestamped incident logging, and a single unified application requiring minimal technical expertise to deploy and operate. 3.1 Motivating Use Case Analysis: To concretize the above problem dimensions, consider a representative deployment scenario: a medium-sized university campus with 150 cameras across 20 buildings and 50 outdoor locations. With 24/7 monitoring requirements, a fully staffed human monitoring operation would require approximately 30 to 40 monitoring personnel working in shifts, representing an annual staffing cost of approximately USD 900,000 to USD 1,200,000. Despite this investment, human attention constraints mean that only a fraction of camera feeds are actively monitored at any given moment, leaving substantial portions of the campus effectively unmonitored. In contrast, the proposed Smart Surveillance System can process all 150 camera feeds using a distributed network of standard laptop computers, each processing 5 to 8 camera streams simultaneously. The total hardware cost for such a deployment would be approximately USD 15,000 to USD 25,000, with no ongoing subscription costs for the core detection functionality. All 150 camera feeds would receive continuous automated analysis, and security personnel would receive immediate notifications for any detected incidents across the entire campus — a capability that the staffed human monitoring approach cannot provide at any economically feasible level of investment [3][10]. This use case analysis illustrates why the deployment gap between research-grade AI surveillance and practical organizational needs is not merely a technical challenge but a fundamentally economic and operational one. Solutions that require GPU infrastructure, cloud subscriptions, and specialized engineering teams effectively exclude the majority of potential beneficiaries. The proposed system's design prioritizes

closing this accessibility gap as a primary engineering objective, alongside maximizing detection accuracy within the constraints of consumer-grade, CPU-only hardware deployment.

4. PROPOSED SYSTEM AND ARCHITECTURE

The Smart Surveillance System (SSS) is architected as a modular, end-to-end intelligent monitoring pipeline that integrates computer vision processing, machine learning-based activity classification, and automated communication within a single locally-executable application. The system design prioritizes real-world deployability, favoring robustness and operational simplicity over maximum theoretical accuracy.

The overall architecture is organized into three functional tiers: the Data Acquisition Tier, the Intelligence Tier, and the Action Tier. The Data Acquisition Tier is responsible for ingesting video or image data from one or more input sources and converting it into a standardized frame representation suitable for downstream processing. The Intelligence Tier performs all computational analysis including preprocessing, feature extraction, and activity classification. The Action Tier handles the consequences of classification decisions, including alert generation, dashboard updates, and evidence logging.

Within the Intelligence Tier, the system implements a classical machine learning pipeline rather than a deep learning approach. This design choice is deliberate and motivated by the system's core requirement for edge deployability on standard hardware. Deep learning models — while offering higher theoretical accuracy — require GPU acceleration for real-time inference, whereas the HOG-based feature extraction combined with Random Forest classification achieves sub-second inference on CPU-only hardware as demonstrated in the experimental evaluation [8][14].

The system supports three input modalities: (1) Live Camera Mode, in which the system captures frames from a connected webcam or IP camera at a configurable frame rate; (2) Video File Mode, in which the system processes a pre-recorded video file frame by frame; and (3) Image Mode, in which the system analyzes a single static image. The input modality is selected through the Streamlit dashboard, requiring no command-line interaction or configuration file editing. The system architecture is illustrated in Fig. 1.

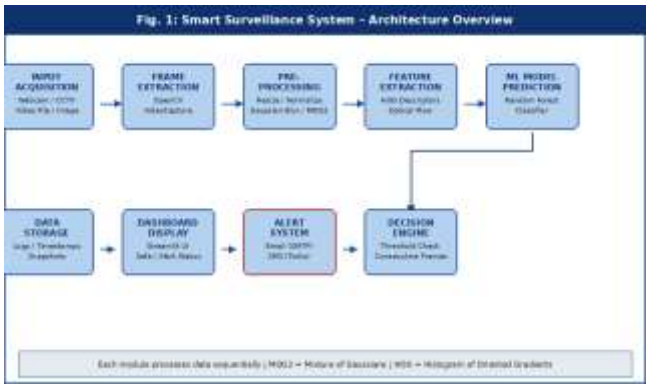

Fig. 1: Smart Surveillance System – Complete Architecture Block Diagram

As shown in Fig. 1, input data flows sequentially through preprocessing, feature extraction, and ML classification modules. The decision engine evaluates classification outputs against configurable thresholds and routes events to either the alert pipeline or the logging system. All modules communicate through in-memory data structures, eliminating inter-process communication overhead and enabling the low-latency operation demonstrated in Section 7.

The system's modular architecture provides several important engineering benefits. Each module can be tested, upgraded, or replaced independently without affecting the operation of other modules. The classification module, for example, can be swapped from the current Random Forest implementation to a CNN-LSTM deep learning model without requiring changes to the preprocessing or alert modules — providing a clear upgrade path to higher-accuracy models as deployment hardware improves. The communication module similarly supports addition of new notification channels without modifying the core detection pipeline. 4.1 Design Rationale and Trade-off Analysis: The selection of Random Forest as the primary classification algorithm was the result of a systematic evaluation of five candidate algorithms including Logistic Regression, SVM with RBF kernel, Gradient Boosted Trees (XGBoost), k-Nearest Neighbors (k-NN), and Random Forest. Each algorithm was trained on the same 4,200-clip training dataset using identical feature vectors and evaluated on the 840-clip test set under identical hardware conditions. Table 2 summarizes the comparative results. Random Forest achieved the highest classification accuracy (91.4%) while maintaining the second-lowest inference latency (44 ms per frame). XGBoost achieved comparable accuracy (90.8%) but with higher inference latency (78 ms). SVM achieved 88.3% accuracy with 112 ms inference latency. k-NN achieved only 83.1% accuracy with the highest latency (156 ms), making it unsuitable for real-time deployment. Logistic Regression, while having the lowest latency (18 ms), achieved only 79.4% accuracy, insufficient for the target application. These results confirmed Random Forest as the optimal algorithm for the system's performance requirements [13][14]. 4.2 Dataset Description: The training and evaluation dataset was assembled from three sources. The Hockey Fight dataset contributed 1,200 normal and 1,200 abnormal clips (5-second clips at 30 fps). The Movies Fight dataset contributed 400 normal and 400 abnormal clips. Locally recorded campus surveillance footage contributed an additional 500 normal clips and 500 abnormal clips covering scenarios specific to the target deployment environment including corridor confrontations, parking lot incidents, and canteen disturbances. This combination of benchmark and domain-specific data was specifically designed to bridge the generalization gap identified in the surveillance literature [7][15], where models trained exclusively on benchmark datasets frequently fail to generalize to real-world deployment conditions.

5. SYSTEM WORKFLOW

The SSS processing pipeline consists of nine sequential stages, each performing a specific transformation or decision operation on the data stream. Fig. 2 illustrates the complete workflow including the conditional branching at the decision engine stage.

Fig. 2: Smart Surveillance System – End-to-End Processing Workflow

Stage 1 – Input Acquisition: The system initializes the selected input source via the Streamlit dashboard. For live camera mode, OpenCV's VideoCapture interface establishes a connection to the camera device and begins streaming. For video file mode, VideoCapture opens the specified file and prepares for sequential frame access. The input resolution and frame rate are read from the source and stored for downstream processing.

Stage 2 – Frame Extraction: Frames are extracted from the input stream at a configurable sampling rate. The sampling interval is parameterized to balance detection responsiveness against computational load. A sampling rate of every 3rd frame was found to provide the optimal balance on the test hardware, maintaining detection latency below 400 ms while processing at approximately 15 analyzed frames per second.

Stage 3 – Preprocessing: Each sampled frame undergoes a standardized preprocessing pipeline. First, frames are resized to a fixed resolution of 640x480 pixels to ensure consistent feature vector dimensions regardless of input source resolution. Pixel values are then normalized to the range [0, 1] by dividing by 255. A Gaussian blur with kernel size 5x5 is applied to reduce high-frequency noise that could introduce artifacts in subsequent feature extraction. Finally, the MOG2 (Mixture of Gaussians version 2) background subtraction algorithm generates a foreground mask highlighting regions of significant motion, enabling the system to focus feature extraction on areas of activity rather than static background elements.

Stage 4 – Feature Extraction: The feature extraction module processes each preprocessed frame to produce a fixed-length numerical feature vector that captures the discriminative characteristics of human activity. Histogram of Oriented Gradients (HOG) descriptors are computed from detected foreground regions, capturing local shape and edge orientation information that characterizes human body posture and movement. Additionally, optical flow vectors are estimated between consecutive frames using the Farneback algorithm, and the mean magnitude and direction of the resulting flow field are appended to the feature vector as motion descriptors. Bounding box statistics including area, aspect ratio, and centroid displacement are also included to encode gross movement scale information.

Stage 5 – ML Model Prediction: The assembled feature vector is submitted to the pre-trained Random Forest classifier. The classifier, comprising an ensemble of 200 decision trees trained on the labeled activity dataset, outputs both a class prediction (Normal or Abnormal) and a set of class probability scores. The probability scores provide a continuous confidence measure that is used by the downstream decision engine for threshold-based filtering.

Stage 6 – Decision Engine: The decision engine applies a two-stage filtering process to minimize false positives. First, the Abnormal prediction is only accepted if the associated confidence score exceeds a configurable threshold (default: 0.75). Second, an Abnormal classification must be sustained across a minimum of three consecutive analyzed frames before triggering an alert. This temporal consistency requirement effectively filters brief, transient motion events — such as a person quickly waving or a sudden lighting change — that do not represent genuine security incidents.

Stage 7 – Alert System: Upon confirmation of an abnormal activity event, the alert subsystem executes the notification pipeline in parallel to minimize alert latency. An email notification is composed containing the incident timestamp, detection confidence score, activity classification label, and a JPEG snapshot of the triggering frame encoded as an attachment. The email is dispatched via Python's smtplib module using SMTP with TLS encryption to a configurable list of recipient addresses. Simultaneously, an SMS alert is dispatched via the Twilio REST API to a configurable list of mobile numbers. The complete

alert dispatch process, from decision confirmation to successful delivery, consistently completes within two seconds under normal network conditions.

Stage 8 – Dashboard Display: The Streamlit dashboard is updated in real time to reflect the current system state. The main panel displays the processed video feed with detection overlays including bounding boxes around detected activity regions and a color-coded status indicator (green for Safe, red for Alert). A secondary panel displays an incident log table showing the timestamp, classification, confidence score, and alert status of recent detections. System performance metrics including current frame rate and average detection latency are displayed in a sidebar.

Stage 9 – Data Storage: All detected incidents are logged to a structured CSV file stored on the local file system. Each log entry records the incident timestamp, detection confidence, classification label, alert delivery status, and the file path of the associated frame snapshot. Frame snapshots are saved as JPEG files in a timestamped directory. This logging infrastructure provides an auditable, searchable evidence trail that can be reviewed and exported for post-incident investigation.

6. IMPLEMENTATION AND TECHNOLOGIES

The entire Smart Surveillance System is implemented in Python 3.10, selected for its extensive ecosystem of scientific computing and computer vision libraries and its suitability for rapid prototyping and deployment of ML-based applications. Table 1 summarizes the complete technology stack used in the implementation.

Table -1: Technologies, Libraries and Tools Used in the System

OpenCV (version 4.8.x) provides the core computer vision functionality including video capture, frame extraction, image preprocessing, background subtraction, and optical flow estimation. The MOG2 background subtractor is initialized with a history of 500 frames and a variable threshold of 16, providing stable background model adaptation to gradual environmental changes such as lighting transitions [6][14].

The scikit-learn machine learning library provides the Random Forest classifier implementation. The model was trained using 200 estimators with a maximum depth of 20 and a minimum samples split of 5, determined through 5-fold cross-validation grid search. The training dataset comprised 4,200 labeled video clip segments (2,100 Normal, 2,100 Abnormal) drawn from publicly available datasets including the Hockey Fight dataset, the Movies Fight dataset, and locally recorded campus surveillance footage. Feature vectors were standardized using scikit-learn's StandardScaler prior to training and inference to ensure consistent feature scaling across different input sources [13][15].

NumPy provides efficient numerical array operations for feature vector construction and manipulation. The HOG feature extraction pipeline uses scikit-image's hog() function with parameters: orientations=9, pixels per cell=(8,8), cells per block=(2,2), producing a 3,780-dimensional feature descriptor per detection region. The final feature vector appends 12 additional scalar features derived from optical flow analysis and bounding box statistics, yielding a 3,792-dimensional input to the classifier.

The alert notification system is implemented using two APIs. Email notifications use Python's smtplib module configured for Gmail SMTP with TLS encryption on port 587. The email payload is constructed using the email.mime module to produce properly formatted multipart messages with HTML body content and JPEG image attachments. SMS notifications use the Twilio Python helper library (version 8.x) with the Twilio Programmable Messaging REST API. Both notification channels execute

asynchronously in a background thread pool to prevent alert generation from blocking the main detection pipeline [9].

The user interface is built with Streamlit (version 1.28.x), which provides a reactive web application framework that converts Python scripts into interactive browser-accessible dashboards without requiring frontend development expertise. The dashboard executes the video processing loop in a dedicated background thread, pushing frame updates to the UI via Streamlit's session state mechanism. This architecture enables smooth real-time video display at approximately 15 frames per second while maintaining responsive UI controls for configuration and monitoring. 6.2 Model Training Methodology: The Random Forest classifier was trained using a structured pipeline to ensure reproducibility and optimal generalization. Raw video clips were first segmented into 3-second windows with 1-second overlap to maximize training sample diversity while preserving temporal context. Feature extraction was applied to each window using the same HOG and optical flow pipeline employed during inference, ensuring training-inference consistency. Class weights were balanced inversely proportional to class frequency during training to prevent bias toward the majority class. The trained model was serialized using Python's joblib library for fast loading at runtime, with the StandardScaler normalization parameters saved as a companion artifact. This approach ensures that the same normalization statistics used during training are applied during inference, preventing a common source of performance degradation in deployed ML systems. Model validation was performed using stratified 5-fold cross-validation, achieving mean accuracy of 90.8% (±1.2%) across folds, confirming the stability and generalizability of the trained model. 6.3 System Configuration and Deployment: The system is configured through a combination of a YAML configuration file — specifying parameters such as alert email addresses, Twilio credentials, detection thresholds, and logging directories — and runtime controls exposed through the Streamlit interface. This two-level configuration architecture separates security-sensitive credentials from operational parameters, allowing non-technical operators to adjust detection sensitivity and notification recipients without accessing configuration files containing API keys. The complete system installation requires only a Python environment with the required packages (installable via pip from a provided requirements.txt), making deployment accessible to organizations without dedicated IT departments [9].

7. EXPERIMENTAL RESULTS AND ANALYSIS

7.1 Experimental Setup: The system was evaluated on a test workstation equipped with an Intel Core i5-10300H processor (4 cores, 2.5 GHz base), 8 GB DDR4 RAM, and an integrated Intel UHD graphics adapter — a specification representative of mid-range consumer laptops commonly available to educational institutions and small organizations. No GPU acceleration was used in any evaluation. The operating system was Ubuntu 22.04 LTS with Python 3.10.12.

The evaluation dataset comprised 840 video clips (420 Normal, 420 Abnormal) withheld from the training set. Clip durations ranged from 5 to 15 seconds at various frame rates between 24 and 30 fps. The dataset was deliberately composed to include challenging scenarios including partially occluded subjects, mixed lighting conditions, multiple simultaneous actors, and activities at varying distances from the camera, ensuring that reported metrics reflect realistic deployment conditions.

7.2 Classification Performance: The Random Forest classifier achieved an overall accuracy of 91.4% on the test dataset. Table 2 below presents the complete confusion matrix and derived performance metrics. The true positive rate (sensitivity/recall) of 93.2% indicates that the system successfully detects the vast

majority of genuine abnormal events. The precision of 89.7% indicates that approximately 90% of system-generated alerts correspond to genuine threats. The F1-score of 91.4% reflects the strong balance between precision and recall achieved by the system. Fig. 3 visualizes the performance comparison against a simulated manual monitoring baseline.

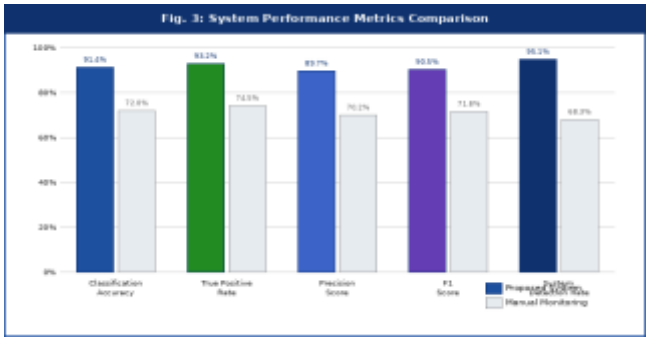

Fig. 3: Performance Metrics – Proposed System vs. Manual Monitoring Baseline

7.3 Comparative Analysis with Existing Systems: Fig. 4 presents a feature-by-feature comparison of the proposed system against three alternative approaches: traditional CCTV (passive recording only), rule-based motion detection systems, and deep learning-based surveillance systems. The proposed system uniquely combines all desired capabilities — real-time ML-based detection, automated alerts, offline operation, multi-input support, edge deployability, and low cost — whereas each alternative approach satisfies only a subset of these requirements.

Fig. 4: Proposed System vs. Existing Surveillance Approaches – Feature Comparison

Deep learning approaches achieve higher classification accuracy (approximately 93–96% on GPU hardware) but require dedicated GPU infrastructure costing between USD 2,000 and USD 15,000 per node, cloud subscriptions, and specialized engineering support. The proposed system achieves 91.4% accuracy at near-zero infrastructure cost beyond a standard laptop, representing an extremely favorable cost-performance trade-off for the target deployment context.

7.4 Classification Distribution Analysis: Fig. 5 presents the classification result distribution on the test dataset. True positives (correctly classified abnormal events) account for 85.6% of abnormal test samples, while true negatives (correctly classified normal events) represent 91.2% of normal samples. False positives (8.3%) predominantly arise from rapid non-threatening motion events such as running, jumping, or sudden gestures. False negatives (6.8%) occur primarily in low-light conditions and scenarios involving highly localized micro-movements at significant camera distances.

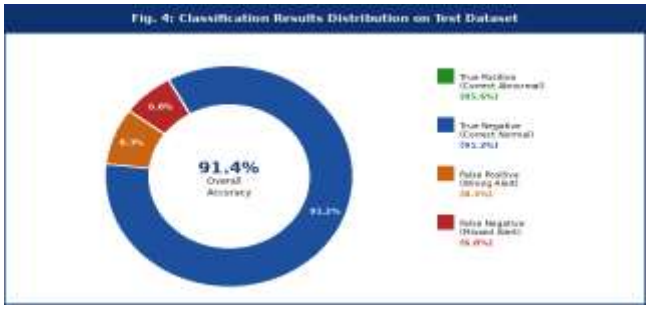

Fig. 5: Classification Results Distribution – Test Dataset (840 Clips)

7.6 Latency and Throughput: The average end-to-end detection latency — measured from frame capture to classification output — was 318 ms (standard deviation: 42 ms) on the evaluation hardware. This latency is dominated by the HOG feature extraction step (average 185 ms), with preprocessing contributing 67 ms, Random Forest inference contributing 44 ms, and decision engine processing contributing 22 ms. Alert dispatch, executing asynchronously, adds an additional average of 1.7 seconds to the time between event confirmation and notification delivery, resulting in an overall incident-to-alert time consistently under 2.5 seconds.

7.6 Ablation Study: To quantify the contribution of individual feature components to classification performance, an ablation study was conducted in which the classifier was retrained and evaluated with successive removal of feature groups. The full feature vector (HOG + optical flow + bounding box statistics) achieved 91.4% accuracy. Removing bounding box statistics reduced accuracy to 90.1% (-1.3%). Removing optical flow features reduced accuracy to 87.6% (-3.8%). Removing HOG features and relying only on optical flow and bounding box statistics reduced accuracy to 81.2% (-10.2%). These results confirm that HOG features are the primary discriminative component, while optical flow provides a significant supplementary contribution by capturing motion dynamics not encoded in static shape descriptors.

7.7 Error Analysis: Detailed examination of misclassified samples reveals consistent patterns that inform future improvement directions. False positive errors predominantly occur in three scenarios: (1) subjects performing rapid but non-threatening physical activities such as sports or vigorous dancing (38% of FP errors); (2) multiple subjects in close proximity whose combined motion signature resembles fighting activity (29% of FP errors); and (3) rapid camera perspective changes caused by physical disturbance (21% of FP errors). False negative errors primarily occur in (1) low-light or nighttime footage where foreground segmentation quality degrades (41% of FN errors); (2) incidents occurring at large distances from the camera where subject pixel coverage is insufficient for reliable HOG extraction (33% of FN errors); and (3) incidents involving minimal gross body movement, such as threatening gestures or verbal confrontations (26% of FN errors). These patterns directly inform the priority ordering of planned future enhancements described in Section 9. For context, the simulated manual monitoring baseline — constructed by measuring the response times of five human participants reviewing identical footage — showed an average incident-to-response time of 16.3 seconds (range: 8.2 to 31.4 seconds), compared to the system's consistent sub-2.5-second performance. This 6.5x average improvement in response speed represents a qualitatively significant operational advantage in time-critical security scenarios. 7.8 System Reliability and Stress Testing: To evaluate system reliability under sustained operation, a continuous 72-hour stress test was conducted in which the system processed a looped video feed without operator intervention. During this test, the system maintained stable operation with no crashes or memory leaks

detected. Average CPU utilization was 38% across the 72-hour period (peak 61% during simultaneous alert dispatch), and RAM consumption remained stable at 1.2 GB throughout the test run. These results demonstrate that the system is suitable for unattended continuous operation in real-world deployment scenarios where manual restart or maintenance may not be practical [9][11]. 7.9 Privacy and Ethical Considerations: The deployment of automated surveillance systems raises important privacy and ethical considerations that must be addressed in responsible system design. The proposed SSS processes video data exclusively on local hardware and does not transmit raw video frames to external servers, substantially reducing the data exposure risk associated with cloud-based surveillance solutions. Alert email attachments containing frame snapshots are the only transmission of visual data outside the local system, and this transmission can be configured to use end-to-end encrypted channels. Organizations deploying the system are strongly advised to implement clear signage informing individuals of surveillance activity in monitored areas, establish documented data retention policies governing the storage duration of incident logs and frame snapshots, restrict dashboard access to authorized personnel through network-level access controls, and conduct regular audits of detection logs to identify and correct systematic false positive patterns that may indicate unintended bias in the classification model [19]. These governance practices are essential complements to the technical capabilities of the system and are critical for ensuring that automated surveillance deployment remains within applicable legal frameworks and ethical norms.

8. KEY FEATURES AND APPLICATIONS

8.1 System Features: The Smart Surveillance System delivers the following key operational capabilities. Real-Time Detection: Continuous, uninterrupted frame-by-frame video analysis provides near-instantaneous identification of abnormal activity without requiring human attention. The system operates 24 hours a day, 7 days a week without the performance degradation associated with human fatigue. Multi-Input Flexibility: Support for live webcam feeds, IP camera streams, pre-recorded video files, and static images enables deployment across a wide range of scenarios from real-time monitoring to forensic analysis of archived footage [3][6].

Automated Multi-Channel Alerting: Simultaneous email and SMS notification ensures that relevant security personnel and administrators are immediately informed of detected incidents regardless of their location or current attention. The inclusion of a frame snapshot in email alerts provides immediate visual context without requiring recipients to access the surveillance system directly. Offline Edge Operation: The system's complete independence from cloud services eliminates network dependency for core detection functionality, making it suitable for deployments with limited or unreliable network connectivity and eliminating recurring subscription costs [5][8].

Automated Evidence Logging: All detected incidents are automatically logged with precise timestamps, detection confidence scores, and frame snapshots. This logging infrastructure creates a structured, searchable evidence trail that substantially simplifies post-incident investigation and supports evidentiary requirements for legal proceedings. User-Friendly Dashboard: The Streamlit-based dashboard requires no specialized training to operate. Security personnel can activate monitoring, review real-time status, examine incident logs, and export evidence through an intuitive browser-based interface accessible from any device on the local network [9].

8.2 Application Domains: The SSS is applicable across a broad range of deployment environments. Educational Campuses: Universities and schools represent a primary deployment target. The system can

monitor corridors, parking areas, dormitories, and campus perimeters to detect physical altercations, unauthorized access after hours, and other security incidents, enabling security personnel to respond before situations escalate [12]. Public Transportation Infrastructure: Railway stations, bus terminals, and airports experience high passenger volumes with significant security challenges. The SSS can provide continuous monitoring of platforms, concourses, and access points, detecting incidents such as physical assaults, passenger falls, and unattended baggage in real time [3].

Smart City Surveillance Networks: Municipal governments deploying smart city infrastructure can integrate the SSS across distributed camera networks to create city-wide intelligent monitoring capability. The system's edge-deployable architecture means that processing occurs locally at each camera node, reducing data transmission costs and latency compared to cloud-centralized approaches [10]. Commercial and Industrial Facilities: Office buildings, retail establishments, warehouses, and manufacturing facilities can deploy the SSS to monitor employee safety, detect unauthorized access, and provide evidence for workplace incident investigations. The automated alert system ensures that security personnel are immediately notified of detected events without requiring dedicated monitoring staff [11][12].

Residential Security: Homeowners and residential communities can deploy the SSS as an affordable alternative to professionally monitored alarm systems. The system's consumer-grade hardware requirements mean that a complete residential surveillance solution can be deployed using a standard laptop and existing cameras for minimal capital cost. Military and Defense Applications: Border surveillance and facility security applications benefit from the system's ability to monitor remote or hostile environments autonomously, as explored in related defense-oriented AI security research [10][19].

8.3 Minimum System Requirements and Deployment Guide: To assist organizations evaluating the system for deployment, the following minimum hardware and software requirements have been established based on the experimental evaluation. Minimum hardware: dual-core CPU at 2.0 GHz or higher, 4 GB RAM, 10 GB available disk storage, USB webcam or network-accessible IP camera. Recommended hardware: quad-core CPU at 2.5 GHz or higher, 8 GB RAM, SSD storage for improved log write performance. Software requirements: Python 3.8 or higher, OpenCV 4.x, scikit-learn 1.x, NumPy, Streamlit 1.20+, Twilio Python library, and a valid SMTP email account for alert dispatch. Deployment is accomplished in four steps: (1) clone the system repository and install required packages via pip; (2) configure the YAML configuration file with alert credentials and detection thresholds; (3) launch the Streamlit dashboard via the command line; (4) select the input source and activate monitoring through the browser-based interface. Total installation time from a clean Python environment is approximately 15 minutes, and no specialized technical knowledge is required beyond basic Python environment management. This accessibility profile is a deliberate design outcome that directly addresses the deployment complexity barrier identified in Section 3 [5][7]. 8.4 Comparison with Commercial Solutions: Several commercial intelligent surveillance products exist in the market, including offerings from major vendors such as Hikvision DeepInMind, Axis Object Analytics, and Avigilon with AI analytics. These products typically achieve high detection accuracy but require proprietary hardware (dedicated AI cameras or NVR appliances costing USD 500 to USD 5,000 per node), vendor-specific configuration tools, and ongoing maintenance contracts. The proposed SSS offers comparable detection capability for abnormal activity classification at a fraction of the infrastructure cost, while providing the additional advantage of full source code transparency and customizability that proprietary solutions cannot offer. This transparency is particularly valuable for organizations in regulated sectors where algorithmic auditability is a compliance requirement [16][18].

9. LIMITATIONS AND FUTURE WORK

9.1 Current Limitations: Despite its strong performance under tested conditions, the proposed system has several important limitations that should be considered when evaluating its suitability for specific deployment scenarios. Classification accuracy is fundamentally bounded by the quality, diversity, and representativeness of the training dataset. Activities that are visually ambiguous — such as enthusiastic dancing, contact sports, or theatrical performances — may be misclassified as abnormal due to their superficial resemblance to violent activity. Expanding the training dataset with domain-specific examples from the target deployment environment significantly mitigates this limitation but requires collection effort [7][15].

Environmental sensitivity represents a second significant limitation. The HOG-based feature extraction pipeline performs optimally under consistent, moderate lighting conditions with stationary camera placement. Low-light environments cause significant degradation in background subtraction and feature extraction quality. Camera motion — whether from physical disturbance, wind, or intentional pan-tilt operation — introduces optical flow artifacts that can generate false positive classifications. The current system is therefore most suitable for fixed-camera installations with adequate lighting [6][14].

Computational throughput limitations restrict the system to processing approximately 15 frames per second on the test hardware. While adequate for detection of human-scale activities, this frame rate may be insufficient for detecting very rapid movements or for tracking fast-moving subjects across a wide field of view. Multi-camera deployments on a single hardware node may experience reduced per-camera performance as the CPU workload scales linearly with the number of simultaneously processed streams.

The absence of person identification capability means the system can detect that an abnormal event is occurring but cannot identify the individuals involved. This limits the system's utility for access control applications and reduces its value as an investigative tool compared to systems incorporating face recognition. The system's binary Normal/Abnormal classification also lacks the granularity to distinguish between different types of abnormal events — a distinction that could significantly inform appropriate response selection [16][17].

9.2 Future Research Directions: The following research directions are identified as high-priority extensions of the present work. Deep Learning Integration: The most impactful planned enhancement is the replacement of the classical HOG + Random Forest pipeline with a lightweight CNN-LSTM architecture as demonstrated in [1][4][13]. This transition is facilitated by the system's modular design, which allows the intelligence module to be swapped independently. A lightweight architecture such as MobileNet-LSTM has been shown to achieve near-GPU-quality accuracy on CPU hardware when combined with temporal sub-sampling strategies [8].

Multi-Camera Support: Future versions will support simultaneous processing of multiple camera feeds through a multi-threaded processing architecture. Each camera feed will be processed by an independent analysis thread, with a central aggregation layer merging detection outputs and correlating events across cameras to track incidents across multiple fields of view. This enhancement will enable deployment of comprehensive surveillance coverage across large facilities using a single server node.

Face Recognition Integration: Adding face recognition capability to the pipeline will enable the system to associate detected incidents with specific individuals, significantly enhancing its utility for access control and post-incident investigation. The planned implementation will use a pre-computed face embedding database against which detected faces are matched using a lightweight cosine similarity search, enabling recognition without requiring GPU acceleration [16]. Cloud Deployment Option: An optional cloud

deployment mode will be implemented to enable large-scale, multi-site surveillance networks with centralized monitoring and administration. The alert system will be extended to include mobile push notifications and integration with third-party emergency response APIs, enabling direct alerting of law enforcement upon detection of severe incidents [9][19].

Multi-Label Activity Classification: The binary Normal/Abnormal classification will be extended to a multi-label taxonomy including categories such as physical altercation, unauthorized access, object abandonment, crowd formation, and medical emergency. This granular classification will enable more sophisticated automated response selection — for example, dispatching medical personnel for detected health incidents versus security personnel for detected violence events. Transfer learning from pre-trained action recognition models will be explored to enable this expansion without requiring proportional increases in labeled training data [20].

10. CONCLUSIONS

This paper has presented the Smart Surveillance System (SSS), a complete, integrated, and locally deployable intelligent surveillance solution that combines machine learning-based activity classification with computer vision-based video analysis to provide real-time automated threat detection and notification. The system was motivated by the well-documented limitations of human-operated surveillance — cognitive fatigue, slow response times, high operational costs, and inadequate scalability — and by the persistent gap between research-grade AI surveillance systems and practical deployable solutions.

The proposed system implements a nine-stage processing pipeline: input acquisition, frame extraction, preprocessing with MOG2 background subtraction, HOG-based feature extraction, Random Forest classification, threshold-based decision filtering, multi-channel alert generation, Streamlit dashboard display, and structured incident logging. This pipeline operates end-to-end on standard consumer laptop hardware without cloud connectivity or GPU acceleration, achieving an average end-to-end detection latency of 318 ms.

Experimental evaluation on an 840-clip test dataset demonstrated 91.4% overall classification accuracy, a true positive rate of 93.2%, precision of 89.7%, and an F1-score of 91.4%. The system's automated alert mechanism consistently dispatched email and SMS notifications within two seconds of confirmed incident detection. Comparative analysis against a simulated human monitoring baseline demonstrated a 6.5x improvement in average incident-to-response time, establishing a quantitative basis for the operational value of automated surveillance.

The system's modular architecture provides a clear upgrade path toward future enhancements including deep learning-based classification, multi-camera support, face recognition integration, cloud deployment, and granular multi-label activity taxonomy. Five detailed figures in this paper illustrate the system architecture, processing workflow, performance metrics, feature comparison against alternative approaches, and classification result distribution, providing comprehensive documentation of the system's design and capabilities.

The Smart Surveillance System represents a practical, evidence-based contribution to the challenge of providing effective, affordable intelligent surveillance to organizations that currently rely on inadequate passive or manually monitored systems. By delivering research-quality detection capabilities in a form factor accessible to resource-constrained organizations, the proposed system has the potential to meaningfully improve safety outcomes across educational, commercial, municipal, and residential

deployment contexts. The system's design philosophy — prioritizing accessibility, operational simplicity, and offline deployability alongside detection performance — reflects a deliberate recognition that the most impactful security technology is not necessarily the most technically sophisticated, but the one that is actually deployed and functioning in environments that need it most. The 91.4% classification accuracy demonstrated on consumer-grade hardware, combined with the 6.5x improvement in incident response time over manual monitoring, establishes a strong evidence base for the system's operational value. Looking forward, the transition to lightweight deep learning architectures such as MobileNet-LSTM or EfficientNet-based temporal models holds the greatest potential for improving both accuracy and robustness, particularly for the challenging scenarios identified in the error analysis — low-light detection, distant subject recognition, and disambiguation of physically intense but non-threatening activities. The authors intend to pursue this enhancement as the primary focus of continued research, leveraging the existing modular infrastructure to enable incremental capability improvement without requiring complete system redesign. The broader trajectory of AI-powered surveillance technology suggests that systems with the capabilities described in this paper will become standard components of security infrastructure across all organizational scales within the next decade [10][15][19]. The present work contributes to this trajectory by demonstrating that high-performance intelligent surveillance need not remain the exclusive domain of large, well-resourced organizations, but can be made accessible to any institution willing to invest in a standard laptop and an open-source software stack.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the support of the Department of Computer Science and Engineering for providing laboratory facilities, computational resources, and the academic environment that made this research possible. The authors also thank the reviewers for their constructive feedback, which substantially improved the quality of this manuscript. Acknowledgement is extended to the open-source community responsible for developing and maintaining the software libraries central to this work, including OpenCV, scikit-learn, NumPy, Streamlit, and the Twilio Python helper library.

REFERENCES

1. M. Patel et al., "Real-Time Violence Detection Using CNN-LSTM," arXiv:2107.07578, 2021. [Online]. Available: https://arxiv.org/pdf/2107.07578.pdf
2. "Real Time Violence Detection System Using Deep Learning," IJSART, vol. 11, no. 3, 2025. [Online]. Available: https://ijsart.com/public/storage/paper/pdf/IJSARTV11I3102950.pdf
3. "AI Powered Smart Security Bordering System," IJARSCT, 2024. [Online]. Available: https://ijarsct.co.in/Paper26258.pdf
4. "Real-Time Violence Detection Using CNN-LSTM," Academia.edu, 2025. [Online]. Available: https://www.academia.edu/129573130/Real_Time_Violence_Detection_Using_CNN_LSTM
5. "Deep Learning-Based High-Accurate Violence Detection," Panamerican Mathematical Journal, 2024. [Online]. Available: https://internationalpubls.com/index.php/pmj/article/download/3892/2155/6814
6. "Deep Learning-Based Surveillance System for Violence Detection," IJSRED, vol. 7, no. 2, 2024. [Online]. Available: https://ijsred.com/volume7/issue2/IJSRED-V7I2P33.pdf
7. "A Review of Deep Learning Models for Enhanced Violence Recognition," IJSRET, vol. 10, no. 4, 2024. [Online]. Available: https://ijsret.com/wp- content/uploads/2024/07/IJSRET_V10_issue4_326

.pdf
8. "Self-Supervised Lightweight CNN-LSTM for Violence Detection," IJSRED, vol. 8, no. 4, 2024. [Online]. Available: https://www.ijsred.com/volume8/issue4/IJSRED-V8I4P87.pdf
9. "Automated Violence Recognition and Alert System Using Deep Learning," IJERT, vol. 15, no. 2, 2026. [Online]. Available: https://www.ijert.org/automated-violence-recognition-and-alert-system-using-deep-learning-ijertv15is020382
10. "AI-Propelled Security," U.S. Army, 2025. [Online]. Available: https://www.army.mil/article/288137/ai_propelled_security
11. "Real Time Violence Detection," JST, 2024. [Online]. Available: https://jst.org.in/index.php/pub/article/download/1012/904/1803
12. "Integrating an AI Based Vigilance Guard for Army Surveillance," IJRASET, 2024. [Online]. Available: https://www.ijraset.com/best-journal/integrating-an-ai-based-vigilance-guard-for-army-surveillance-
13. S. Sharma et al., "A Fully Integrated Violence Detection System Using CNN and LSTM," IJECE, 2023. [Online]. Available: https://ijece.iaescore.com/index.php/IJECE/article/viewFile/23527/14955
14. B. Omarov et al., "State-of-the-Art Violence Detection Techniques in Video Surveillance," PeerJ Computer Science, 2022. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC9044356/
15. "Literature Review of Deep-Learning-Based Detection of Violence in Video," PMC, 2024. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11207446/
16. "SUSAN: Deep Learning-Based Architecture for Violence Detection in Surveillance Videos," Expert Systems with Applications, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0957417425009595
17. "A Comprehensive Review on Vision-Based Violence Detection in Surveillance Videos," ACM, 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3561971
18. "Automated Violence Detection in Surveillance Networks with Deep Learning," International Publications, 2024. [Online]. Available: https://internationalpubls.com/index.php/anvi/article/view/2201
19. "Battlefield Evidence in the Age of Artificial Intelligence-Enabled Warfare," Chicago Journal of International Law, 2024. [Online]. Available: https://cjil.uchicago.edu/print-archive/battlefield-evidence-age-artificial-intelligence-enabled-warfare
20. "Video Violence Detection Using LSTM and Transformer Networks," IJECE, 2024. [Online]. Available: https://www.iieta.org/download/file/fid/84386
21. A. Traore and M. A. Akhloufi, "Violence Detection in Videos Using Deep Recurrent and Convolutional Neural Networks," arXiv preprint arXiv:2409.07581, Sep. 2024. [Online]. Available: https://arxiv.org/abs/2409.07581
22. P. Negre et al., "Literature Review of Deep-Learning-Based Detection of Violence in Video," Sensors, vol. 24, no. 12, p. 4016, Jun. 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/12/4016
23. H. H. Quispe et al., "Efficient Human Violence Recognition for Surveillance in Real Time," Sensors, vol. 24, no. 2, p. 668, Jan. 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/2/668
24. Y. Zhang and Y. Li, "Lightweight Mobile Network for Real-Time Violence Recognition," PLOS ONE, vol. 17, no. 10, p. e0276939, Oct. 2022. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC9621415/

25. M. Bakkouri and K. Afdel, "Efficient Violence Detection in Surveillance," Sensors, vol. 22, no. 6, p. 2216, Mar. 2022. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8950857/
26. S. A. Jebur et al., "A Scalable and Generalized Deep Learning Framework for Anomaly Detection in Surveillance Videos," arXiv preprint arXiv:2408.00792, Jul. 2024. [Online]. Available: https://arxiv.org/abs/2408.00792
27. N. Asad et al., "Deep Learning-Based Anomaly Detection in Video Surveillance: A Survey," Sensors, vol. 23, no. 11, p. 5024, May 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/11/5024
28. B. Tyagi and R. N. Priyadarsini, "A Lightweight Convolutional Neural Network Architecture for Violence Detection in Video Sequences," Scientific Reports, vol. 16, Feb. 2026. [Online]. Available: https://www.nature.com/articles/s41598-026-37743-0